

[REDACTED]

**IN THE UNITED STATES DISTRICT COURT  
FOR THE EASTERN DISTRICT OF TEXAS  
MARSHALL DIVISION**

NETWORK SYSTEM TECHNOLOGIES, LLC

Plaintiff,

v.

SAMSUNG ELECTRONICS CO., LTD.,  
SAMSUNG ELECTRONICS AMERICA, INC.,  
SAMSUNG AUSTIN SEMICONDUCTOR,  
LLC, SAMSUNG SEMICONDUCTOR, INC.,  
LENOVO GROUP LTD., ONEPLUS  
TECHNOLOGY (SHENZHEN) CO., LTD.,

Defendants.

CIVIL ACTION NO. 2:22-CV-481

**JURY TRIAL DEMANDED**

[REDACTED]

**SAMSUNG'S FIRST AMENDED ANSWER AND  
COUNTERCLAIMS TO PLAINTIFF'S COMPLAINT**

Defendants Samsung Electronics Co., Ltd. ("SEC"), Samsung Electronics America, Inc. ("SEA"), Samsung Austin Semiconductor, LLC ("SAS"), and Samsung Semiconductor, Inc. ("SSI") (collectively "Samsung") hereby submits its First Amended Answer to the Complaint filed by Plaintiff Network System Technologies, LLC ("NST" or "Plaintiff") and submits its Affirmative Defenses. To the extent not specifically admitted below, Samsung denies the allegations of the Complaint. Any admissions herein are for purposes of this matter only. Samsung also reserves the right to take further positions and raise additional defenses that may become apparent as a result of additional information discovered subsequent to filing the First Amended Answer and Counterclaims.

**NATURE OF THE ACTION**

1. Samsung admits that Plaintiff purports to set forth an action for patent infringement under the Patent Laws of the United States, 35 U.S.C. § 1 *et seq*, involving six patents purported to be Plaintiff's: U.S. Patent Nos. 7,366,818 ("818 patent"); 7,373,449 ("449 patent"); 7,594,052 ("052 patent"); 7,769,893 ("9893 patent"); 8,072,893 ("2893 patent"); and 8,086,800 ("800 patent") (collectively, the "Asserted Patents"). Samsung denies that it has infringed or continues to infringe, directly or indirectly, any of the Asserted Patents.

**PARTIES**

2. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in Paragraph 2 of the Complaint, and therefore denies the same.

3. Samsung admits that SEC is a company duly organized and existing under the laws of the Republic of Korea with a principal place of business at 129 Samsung-ro, Maetan-3dong, Yeongtong-gu, Suwon-si, Gyeonggi-do, 16677, Republic of Korea.

4. Samsung admits that SEA is a corporation duly organized and existing under the laws of the State of New York with a principal place of business at 85 Challenger Road, Ridgefield Park, New Jersey 07600. Samsung admits that SEA is a wholly-owned subsidiary of SEC. Samsung admits that SEA has offices in the Eastern District of Texas, including at 6625 Excellence Way, Plano, Texas 75023; 2601 Preston Road, Frisco, Texas 75034; and 3580 Preston Road, Suite 100, Frisco, Texas 75034. Samsung further admits that SEA has a registered agent, CT Corporation System located at 1999 Bryan St., Ste. 900, Dallas, TX 75201. The remaining allegations in paragraph 4 of the Complaint are legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the remaining allegations in paragraph 4 of the Complaint.

[REDACTED]

5. Samsung admits that SAS is a limited liability company organized and existing under the laws of the State of Delaware, with a principal place of business at 12100 Samsung Boulevard, Austin, Texas 75754. Samsung admits that SAS is a wholly-owned subsidiary of SSI, which is a wholly-owned subsidiary of SEA, which is a wholly-owned subsidiary of SEC. Samsung admits that SAS has a registered agent CT Corporation System, located at 1999 Bryan St., Ste. 900, Dallas, TX 75201. The remaining allegations in paragraph 5 of the Complaint are legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the remaining allegations in paragraph 5 of the Complaint.

6. Samsung admits that SSI is a corporation organized and existing under the laws of the State of California, with a principal place of business at 3655 North First Street, San Jose, California 95134. Samsung admits that SSI is a wholly-owned subsidiary of SEA. Samsung admits that SEA has offices in the Eastern District of Texas, but denies that SSI has any business locations in this Judicial District. Samsung denies that SSI has a business location at 6625 Excellence Way, Plano, Texas 75023. Samsung admits that SSI has a registered agent National Registered Agents, Inc., located at 1999 Bryan St., Ste. 900, Dallas, TX 75201. The remaining allegations in paragraph 4 of the Complaint are legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the remaining allegations in paragraph 6 of the Complaint.

7. Samsung admits that the Complaint collectively refers to SEC, SEA, SAS, and SSI as "Samsung."

8. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in Paragraph 8 of the Complaint, and therefore denies the same.

[REDACTED]

9. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in Paragraph 9 of the Complaint, and therefore denies the same.

10. Samsung denies that its products infringe any valid and enforceable claim of any of the Asserted Patents. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in Paragraph 10 of the Complaint, and therefore denies the same.

### **JURISDICTION AND VENUE**

11. The allegations contained in paragraph 11 of the Complaint are legal conclusions as to which no responsive pleading is necessary. To the extent that a response is required, Samsung admits that, for the purposes of this action only, this Court has subject matter jurisdiction pursuant to 28 U.S.C. §§ 1331 and 1338(a).

12. The allegations contained in paragraph 12 of the Complaint are legal conclusions as to which no responsive pleading is necessary. To the extent that a response is required, for the purposes of this action only, Samsung does not challenge that this Court has personal jurisdiction over Samsung.

13. The allegations contained in paragraph 13 of the Complaint are legal conclusions as to which no responsive pleading is necessary. To the extent that a response is required, Samsung denies venue is proper in this Court for SSI and SAS. Samsung will not contest, for purposes of this case only, that venue is proper in this Court for SEC and SEA. Samsung further reserves the right to move to dismiss, and/or seek a transfer pursuant to 28 U.S.C. §§ 1404 and 1406.

14. The allegations contained in paragraph 14 of the Complaint are legal conclusions as to which no responsive pleading is necessary. To the extent that a response is required, for the purposes of this action only, Samsung does not challenge that this Court has personal jurisdiction over Samsung and Samsung denies the remaining allegations in paragraph 14 of the Complaint.

[REDACTED]

15. Samsung admits that SEA has authorized retailers and distributors in the State of Texas and this Judicial District. Samsung denies that its products infringe any of the Asserted Patents. Samsung denies the remaining allegations in paragraph 15 of the Complaint.

16. The allegations contained in paragraph 16 of the Complaint are legal conclusions as to which no responsive pleading is necessary. To the extent that a response is required, for the purposes of this action only, Samsung does not challenge that this Court has personal jurisdiction over Samsung and Samsung denies the remaining allegations in paragraph 16 of the Complaint.

17. The allegations contained in paragraph 17 of the Complaint are legal conclusions as to which no responsive pleading is necessary. To the extent that a response is required, Samsung denies venue is proper in this Court for SSI and SAS. Samsung denies that SSI and SAS have a regular and established place of business in this District. Samsung will not contest, for purposes of this case only, that venue is proper in this Court for SEC and SEA. Samsung denies that SEC, SSI, SEA, or SAS has committed any acts of infringement with respect to the Asserted Patents anywhere, including in this District. Samsung further reserves the right to move to dismiss, and/or seek a transfer pursuant to 28 U.S.C. §§ 1404 and 1406. To the extent that a further response is required, Samsung denies the remaining allegations in paragraph 17 of the Complaint.

18. The allegations contained in paragraph 18 of the Complaint are legal conclusions as to which no responsive pleading is necessary or Samsung lacks knowledge or information sufficient to form a belief as to the truth of those remaining allegations. To the extent that a response is required, for the limited purpose of this action only, Samsung does not deny that venue is proper in this Court for SEC and Samsung denies the remaining allegations in paragraph 18 of the Complaint.

[REDACTED]

19. The allegations contained in paragraph 19 of the Complaint are legal conclusions as to which no responsive pleading is necessary. To the extent that a response is required, Samsung denies that venue is proper in this Court for SSI and SAS. Samsung denies that SSI and SAS have a regular and established place of business in this District. Samsung further denies that SEC, SSI, SEA, or SAS has committed any acts of infringement with respect to the Asserted Patents, including in this District. For the purposes of this action only, Samsung admits that venue is proper for SEA. Samsung further reserves the right to move to dismiss, and/or seek a transfer pursuant to 28 U.S.C. §§ 1404 and 1406.

### **JOINDER**

20. The allegations in paragraph 20 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the allegations in paragraph 20 of the Complaint.

21. Samsung admits that the Complaint purports to set forth the accused products into three categories: 1) products containing Qualcomm “Snapdragon” SoCs; 2) Samsung “Exynos” SoCs; and 3) products containing Exynos SoCs. Samsung admits that it makes, uses, imports, offers for sale, and/or sells accused products that fall within these categories. Samsung denies that its products infringe any valid and enforceable claim of any of the Asserted Patents. The remaining allegations in paragraph 21 of the Complaint express legal conclusions and thus no response is required. To the extent a response is required, Samsung lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 21 of the Complaint, and therefore denies the same.

22. Samsung admits that it purchases or otherwise obtains Snapdragon SoCs from Qualcomm and that there are agreements between Qualcomm and Samsung. Samsung lacks

[REDACTED]

knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 22 of the Complaint, and therefore denies the same.

**THE ASSERTED PATENTS**

23. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 23 of the Complaint and, on that basis, denies them.

**U.S. Patent No. 7,366,818**

24. Samsung admits that the '818 patent is entitled "INTEGRATED CIRCUIT COMPRISING A PLURALITY OF PROCESSING MODULES AND A NETWORK AND METHOD FOR EXCHANGING DATA USING SAME." Samsung admits that, on its face, the '818 patent issued on April 29, 2008 and lists Andrei Radulescu and Kees Gerard Willem Goossens as the named inventors. Samsung admits that what is purported to be a copy of the '818 patent is attached to the Complaint as Exhibit 1. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 24 of the Complaint and, on that basis, denies them.

25. Samsung admits that the '818 patent has 7 claims: 1 independent claim and 6 dependent claims.

26. The allegations in paragraph 26 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the allegations in paragraph 26 of the Complaint.

27. Samsung admits that Claim 1 of the '818 patent is reproduced in paragraph 27 of the Complaint. The remaining allegations in paragraph 27 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the remainder of the allegations in paragraph 27 of the Complaint.

[REDACTED]

28. Samsung admits that it had knowledge of the '818 patent as of the date of service of the Complaint. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 28 of the Complaint and, on that basis, denies them.

29. Samsung denies that it monitored patenting activity by Koninklijke Philips N.V. ("Philips") and further denies that it reviewed or was aware of the U.S. published patent applications and patents obtained by Philips shortly after their publication. Samsung denies that it considered Philips' U.S. published patent applications and patents in the semiconductor space or their applicability to its own products. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 29 of the Complaint and, on that basis, denies them.

30. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 30 of the Complaint and, on that basis, denies them.

31. Samsung admits that it respects the intellectual property rights of others, and that it seeks to develop innovative and groundbreaking products, including the accused products. The remainder of the allegations pertaining to Samsung in paragraph 31 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the remainder of the allegations pertaining to Samsung in paragraph 31 of the Complaint. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 31 of the Complaint and, on that basis, denies them.

32. Samsung denies the allegations in paragraph 32 of the Complaint with respect to Samsung. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 32 of the Complaint and, on that basis, denies them.

[REDACTED]

33. Samsung denies the allegations in paragraph 33 of the Complaint with respect to Samsung. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 33 of the Complaint and, on that basis, denies them.

34. The allegations in paragraph 34 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the allegations in paragraph 34 of the Complaint.

35. Samsung incorporates by reference its responses to the allegations in paragraphs 28 through 34 of the Complaint as set forth above. Samsung denies the remaining allegations in paragraph 35 of the Complaint with respect to Samsung. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 35 of the Complaint and, on that basis, denies them.

**U.S. Patent No. 7,373,449**

36. Samsung admits that the '449 patent is entitled "APPARATUS AND METHOD FOR COMMUNICATING USING AN INTEGRATED CIRCUIT." Samsung admits that, on its face, the '449 patent issued on May 13, 2008 and lists Andrei Radulescu and Kees Gerard Willem Goossens as the named inventors. Samsung admits that what is purported to be a copy of the '449 patent is attached to the Complaint as Exhibit 2. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 36 of the Complaint and, on that basis, denies them.

37. Samsung admits that the '449 patent has 18 claims: 2 independent claims and 16 dependent claims.

38. The allegations in paragraph 38 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the allegations in paragraph 38 of the Complaint.

[REDACTED]

39. Samsung admits that Claim 10 of the '449 patent is reproduced in paragraph 39 of the Complaint. The remaining allegations in paragraph 39 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the remainder of the allegations in paragraph 39 of the Complaint.

40. Samsung admits that it had knowledge of the '449 patent as of the date of service of the Complaint.

41. Samsung denies that it monitored patenting activity by Philips and further denies that it reviewed or was aware of the U.S. published patent applications and patents obtained by Philips shortly after their publication. Samsung denies that it considered Philips' U.S. published patent applications and patents in the semiconductor space or their applicability to its own products. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 41 of the Complaint and, on that basis, denies them.

42. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 42 of the Complaint and, on that basis, denies them.

43. Samsung incorporates by reference its responses to the allegations in paragraphs 44 through 45 of the Complaint as set forth below.

44. Samsung admits that U.S. Patent Application No. 11/513,021, entitled "NOC SYSTEM EMPLOYING AXI PROTOCOL AXI INTERLEAVING METHOD THEREOF," with a filing date of August 31, 2006, is assigned to Samsung and that it claims priority to Republic of Korea Application 10-2005-0097582 filed October 17, 2005. Samsung further admits that during prosecution of U.S. Application No. 11/513,021, the examiner included the '449 patent in a list of references that the examiner described as "prior art made of record and not relied upon [and]

considered pertinent to applicant's disclosure." Samsung denies the remaining allegations in paragraph 44 of the Complaint.

45. Samsung admits that U.S. Patent Application No. 11/652,010, entitled "NETWORK ON CHIP (NOC) RESPONSE SIGNAL CONTROL APPARATUS AND NOC RESPONSE SIGNAL CONTROL METHOD USING THE APPARATUS," with a filing date of January 11, 2007, is assigned to Samsung and that it claims priority to Republic of Korea Application 10-2006-0088652 filed September 13, 2006. Samsung further admits that, during prosecution of U.S. Application No. 11/652,010, U.S. Patent Publication No. 2006/0041889 was listed in an information disclosure statement filed by the applicant. Samsung further admits that, during prosecution of U.S. Application No. 11/652,010, the examiner included the '449 patent in a "Notice of References cited." Samsung denies the remaining allegations in paragraph 45 of the Complaint.

46. Samsung admits that it seeks to obtain valid and enforceable patents.

47. Samsung denies that it was aware of, and further denies that it considered and analyzed the inventions described and claimed in, all patents and patent applications cited during the prosecution of its own patent applications, and during the prosecution of patents and patent applications that it acquired.

48. Samsung admits that it respects the intellectual property rights of others, and that it seeks to develop innovative and groundbreaking products, including the accused products. The remainder of the allegations in paragraph 48 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the remainder of the allegations in paragraph 48 of the Complaint.

49. Samsung denies the allegations in paragraph 49 of the Complaint.

[REDACTED]

50. Samsung denies the allegations in paragraph 50 of the Complaint.

51. The allegations in paragraph 51 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the allegations in paragraph 51 of the Complaint.

52. Samsung incorporates by reference its responses to the allegations in paragraphs 36 through 51 of the Complaint as set forth above. Samsung denies the remaining allegations in paragraph 52 of the Complaint.

53. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 53 of the Complaint and, on that basis, denies them.

54. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 54 of the Complaint and, on that basis, denies them.

55. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 55 of the Complaint and, on that basis, denies them.

56. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 56 of the Complaint and, on that basis, denies them.

57. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 57 of the Complaint and, on that basis, denies them.

**U.S. Patent No. 7,594,052**

58. Samsung admits that the '052 patent is entitled "INTEGRATED CIRCUIT AND METHOD OF COMMUNICATION SERVICE MAPPING." Samsung admits that, on its face, the '052 patent issued on September 22, 2009 and lists Andrei Radulescu and Kees Gerard Willem Goossens as the named inventors. Samsung admits that what is purported to be a copy of the '052 patent is attached to the Complaint as Exhibit 3. Samsung lacks knowledge or information

[REDACTED]

sufficient to form a belief as to the truth of the remaining allegations in paragraph 58 of the Complaint and, on that basis, denies them.

59. Samsung admits that the '052 patent has 7 claims: 3 independent claims and 4 dependent claims.

60. The allegations in paragraph 60 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the allegations in paragraph 60 of the Complaint.

61. Samsung admits that Claim 6 of the '052 patent is reproduced in paragraph 61 of the Complaint. The remaining allegations in paragraph 61 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the remainder of the allegations in paragraph 61 of the Complaint.

62. Samsung admits that it had knowledge of the '052 patent as of the date of service of the Complaint.

63. Samsung denies that it monitored patenting activity by Philips and further denies that it reviewed or was aware of the U.S. published patent applications and patents obtained by Philips shortly after their publication. Samsung denies that it considered Philips' U.S. published patent applications and patents in the semiconductor space or their applicability to its own products. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 63 of the Complaint and, on that basis, denies them.

64. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 64 of the Complaint and, on that basis, denies them.

65. Samsung incorporates by reference its responses to the allegations in paragraphs 66 through 68 of the Complaint as set forth below.

[REDACTED]

66. Samsung admits that U.S. Patent Application No. 11/434,188, entitled “NETWORK ON CHIP SYSTEM EMPLOYING AN ADVANCED EXTENSIBLE INTERFACE PROTOCOL,” with a filing date of May 16, 2006, is assigned to Samsung and that it claims priority to Republic of Korea Application 10-2005-0096105 filed October 12, 2005. Samsung further admits that U.S. Patent Publication No. 2007/0186018 was cited by the examiner during the prosecution of U.S. Patent Application No. 11/434,188. Samsung further admits that, during prosecution of U.S. Patent Application No. 11/434,188, the examiner included the ’052 patent in a “Notice of References cited.” Samsung denies that it filed a Notice of Abandonment with respect to U.S. Patent Application No. 11/434,188. Samsung denies the remaining allegations in paragraph 66 of the Complaint.

67. Samsung admits that U.S. Patent Application No. 11/513,021, entitled “NOC SYSTEM EMPLOYING AXI PROTOCOL AXI INTERLEAVING METHOD THEREOF,” with a filing date of August 31, 2006, is assigned to SEC and that it claims priority to Republic of Korea Application 10-2005-0097582 filed October 17, 2005. Samsung further admits that U.S. Patent Publication No. 2007/0186018, was cited by the examiner during the prosecution of U.S. Patent Application No. 11/513,021. Samsung denies the remaining allegations in paragraph 67 of the Complaint.

68. Samsung admits that U.S. Patent Application No. 11/652,010, entitled “NETWORK ON CHIP (NOC) RESPONSE SIGNAL CONTROL APPARATUS AND NOC RESPONSE SIGNAL CONTROL METHOD USING THE APPARATUS,” with a filing date of January 11, 2007, is assigned to SEC, and that it claims priority to Republic of Korea Application 10-2006-0088652 filed September 13, 2006. Samsung further admits that during the prosecution of U.S. Patent Application No. 11/513,021, the examiner included U.S. Patent Publication No.

[REDACTED]

2007/0186018 in a list of references that the examiner described as “prior art made of record and not relied upon [and] considered pertinent to applicant’s disclosure.” Samsung denies the remaining allegations in paragraph 68 of the Complaint.

69. Samsung admits that it seeks to obtain valid and enforceable patents.

70. Samsung denies that it was aware of, and further denies that it considered and analyzed the inventions described and claimed in, all patents and patent applications cited during the prosecution of its own patent applications, or during the prosecution of patents and patent applications that it acquired.

71. Samsung admits that it respects the intellectual property rights of others, and that it seeks to develop innovative and groundbreaking products, including the accused products. The remainder of the allegations in paragraph 71 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the remainder of the allegations in paragraph 71 of the Complaint.

72. Samsung denies the allegations in paragraph 72 of the Complaint.

73. Samsung denies the allegations in paragraph 73 of the Complaint.

74. The allegations in paragraph 74 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the allegations in paragraph 74 of the Complaint.

75. Samsung incorporates by reference its responses to the allegations in paragraphs 58 through 74 of the Complaint as set forth above. The remainder of the allegations in paragraph 75 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the remainder of the allegations in paragraph 75 of the Complaint.

76. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 76 of the Complaint and, on that basis, denies them.

77. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 77 of the Complaint and, on that basis, denies them.

78. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 78 of the Complaint and, on that basis, denies them.

79. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 79 of the Complaint and, on that basis, denies them.

80. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 80 of the Complaint and, on that basis, denies them.

**U.S. Patent No. 7,769,893**

81. Samsung admits that the '9893 patent is entitled "INTEGRATED CIRCUIT AND METHOD FOR ESTABLISHING TRANSACTIONS." Samsung admits that, on its face, the '9893 patent issued on August 3, 2010 and lists Kees Gerard Willem Goossens as the named inventor. Samsung admits that what is purported to be a copy of the '9893 patent is attached to the Complaint as Exhibit 4. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 81 of the Complaint and, on that basis, denies them.

82. Samsung admits that the '9893 patent has 11 claims: 2 independent claims and 9 dependent claims.

83. The allegations in paragraph 83 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the allegations in paragraph 83 of the Complaint.

[REDACTED]

84. Samsung admits that Claim 4 of the '9893 patent is reproduced in paragraph 84 of the Complaint. The remaining allegations in paragraph 84 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the remainder of the allegations in paragraph 84 of the Complaint.

85. Samsung admits that it had knowledge of the '9893 patent as of the date of service of the Complaint.

86. Samsung denies that it monitored patenting activity by Philips and further denies that it reviewed or was aware of the U.S. published patent applications and patents obtained by Philips shortly after their publication. Samsung denies that it considered Philips' U.S. published patent applications and patents in the semiconductor space or their applicability to its own products. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 86 of the Complaint and, on that basis, denies them.

87. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 87 of the Complaint and, on that basis, denies them.

88. Samsung incorporates by reference its response to the allegations in paragraphs 89 of the Complaint as set forth below.

89. Samsung admits that U.S. Patent Application No. 12/659,218, entitled "HOST DEVICE AND SLAVE DEVICE CONTROLLING METHOD," with a filing date of March 1, 2010, is assigned to Samsung and that it claims priority to Republic of Korea Application 2009-76287 filed August 18, 2009. Samsung further admits that during the prosecution of U.S. Patent Application No. 12/659,218, the examiner included the '9893 patent in a "Notice of References cited." Samsung further admits the examiner stated that "The prior art made of record and not relied upon is considered pertinent to applicant's disclosure."

90. Samsung admits that it seeks to obtain valid and enforceable patents.

91. Samsung denies that it was aware of, and further denies that it considered and analyzed the inventions described and claimed in, all patents and patent applications cited during the prosecution of its own patent applications, or during the prosecution of patents and patent applications that it acquired.

92. Samsung admits that it respects the intellectual property rights of others, and that it seeks to develop innovative and groundbreaking products, including the accused products. The remainder of the allegations in paragraph 92 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the remainder of the allegations in paragraph 92 of the Complaint.

93. Samsung denies the allegations in paragraph 93 of the Complaint.

94. Samsung denies the allegations in paragraph 94 of the Complaint.

95. The allegations in paragraph 95 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the allegations in paragraph 95 of the Complaint.

96. Samsung incorporates by reference its responses to the allegations in paragraphs 81 through 95 of the Complaint as set forth above. The remainder of the allegations in paragraph 96 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the remainder of the allegations in paragraph 96 of the Complaint.

97. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 97 of the Complaint and, on that basis, denies them.

[REDACTED]

98. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 98 of the Complaint and, on that basis, denies them.

99. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 99 of the Complaint and, on that basis, denies them.

100. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 100 of the Complaint and, on that basis, denies them.

101. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 101 of the Complaint and, on that basis, denies them.

**U.S. Patent No. 8,072,893**

102. Samsung admits that the '2893 patent is entitled "INTEGRATED CIRCUIT WITH DATA COMMUNICATION NETWORK AND IC DESIGN METHOD." Samsung admits that, on its face, the '2893 patent issued on December 6, 2011 and lists John Dielissen and Edwin Rijpkema as the named inventors. Samsung admits that what is purported to be a copy of the '2893 patent is attached to the Complaint as Exhibit 5. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 102 of the Complaint and, on that basis, denies them.

103. Samsung admits that the '2893 patent has 12 claims: 5 independent claims and 7 dependent claims.

104. The allegations in paragraph 104 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the allegations in paragraph 104 of the Complaint.

105. Samsung admits that Claims 1 and 10 of the '2893 patent is reproduced in paragraph 105 of the Complaint. The remaining allegations in paragraph 105 of the Complaint

[REDACTED]

express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the remainder of the allegations in paragraph 105 of the Complaint.

106. Samsung admits that it had knowledge of the '2893 patent as of the date of service of the Complaint.

107. Samsung denies that it monitored patenting activity by Philips and further denies that it reviewed or was aware of the U.S. published patent applications and patents obtained by Philips shortly after their publication. Samsung denies that it considered Philips' U.S. published patent applications and patents in the semiconductor space or their applicability to its own products. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 107 of the Complaint and, on that basis, denies them.

108. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 108 of the Complaint and, on that basis, denies them.

109. Samsung incorporates by reference its response to the allegations in paragraph 110 of the Complaint as set forth below.

110. Samsung admits that U.S. Patent Application No. 11/513,021, entitled "NOC SYSTEM EMPLOYING AXI PROTOCOL AXI INTERLEAVING METHOD THEREOF," with a filing date of August 31, 2006, is assigned to SEC, and that it claims priority to Republic of Korea Application 10-2005-0097582 filed October 17, 2005. Samsung further admits that, during prosecution of 11/513,021, the examiner included U.S. Patent Publication No. 2008/0186983 in a list of references that the examiner described as "prior art made of record and not relied upon [and] considered pertinent to applicant's disclosure." Samsung denies the remaining allegations in paragraph 110 of the Complaint.

111. Samsung admits that it seeks to obtain valid and enforceable patents.

[REDACTED]

112. Samsung denies that it was aware of, and further denies that it considered and analyzed the inventions described and claimed in, all patents and patent applications cited during the prosecution of its own patent applications, or during the prosecution of patents and patent applications that it acquired.

113. Samsung admits that it respects the intellectual property rights of others, and that it seeks to develop innovative and groundbreaking products, including the accused products. The remainder of the allegations in paragraph 113 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the remainder of the allegations in paragraph 113 of the Complaint.

114. Samsung denies the allegations in paragraph 114 of the Complaint.

115. Samsung denies the allegations in paragraph 115 of the Complaint.

116. The allegations in paragraph 116 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the allegations in paragraph 116 of the Complaint.

117. Samsung incorporates by reference its responses to the allegations in paragraphs 102 through 116 of the Complaint as set forth above. The remainder of the allegations in paragraph 117 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the remainder of the allegations in paragraph 117 of the Complaint.

118. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 118 of the Complaint and, on that basis, denies them.

119. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 119 of the Complaint and, on that basis, denies them.

[REDACTED]

120. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 120 of the Complaint and, on that basis, denies them.

121. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 121 of the Complaint and, on that basis, denies them.

122. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 122 of the Complaint and, on that basis, denies them.

**U.S. Patent No. 8,086,800**

123. Samsung admits that the '800 patent is entitled "INTEGRATED CIRCUIT AND METHOD FOR BUFFERING TO OPTIMIZE BURST LENGTH IN NETWORKS ON CHIPS." Samsung admits, on its face, that the '800 patent issued on December 27, 2011 and lists Andrei Radulescu and Kees Gerard Willem Goossens as the named inventors. Samsung admits that what is purported to be a copy of the '800 patent is attached to the Complaint as Exhibit 6. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 123 of the Complaint and, on that basis, denies them.

124. Samsung admits that the '800 patent has 21 claims: 4 independent claims and 17 dependent claims.

125. The allegations in paragraph 125 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the allegations in paragraph 125 of the Complaint.

126. Samsung admits that Claim 10 of the '800 patent is reproduced in paragraph 126 of the Complaint. The remaining allegations in paragraph 126 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the remainder of the allegations in paragraph 126 of the Complaint.

[REDACTED]

127. Samsung admits that it had knowledge of the '800 patent as of the date of service of the Complaint.

128. Samsung denies that it monitored patenting activity by Philips and further denies that it reviewed or was aware of the U.S. published patent applications and patents obtained by Philips shortly after their publication. Samsung denies that it considered Philips' U.S. published patent applications and patents in the semiconductor space or their applicability to its own products. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 128 of the Complaint and, on that basis, denies them.

129. Samsung incorporates by reference its response to the allegations in paragraph 130 of the Complaint as set forth below.

130. Samsung admits that U.S. Patent Application No. 11/513,021, entitled "NOC SYSTEM EMPLOYING AXI PROTOCOL AXI INTERLEAVING METHOD THEREOF," with a filing date of August 31, 2006, is assigned to SEC, and that it claims priority to Republic of Korea Application 10-2005-0097582 filed October 17, 2005. Samsung further admits that, during the prosecution of U.S. Patent Application No. 11/513,021, the examiner included U.S. Patent Publication No. 2007/0226407 in a list of references that the examiner described as "prior art made of record and not relied upon [and] considered pertinent to applicant's disclosure."

131. Samsung admits that it seeks to obtain valid and enforceable patents.

132. Samsung denies that it was aware of, and further denies that it considered and analyzed the inventions described and claimed in, all patents and patent applications cited during the prosecution of its own patent applications, or during the prosecution of patents and patent applications that it acquired.

[REDACTED]

133. Samsung admits that it respects the intellectual property rights of others, and that it seeks to develop innovative and groundbreaking products, including the accused products. The remainder of the allegations in paragraph 133 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the remainder of the allegations in paragraph 133 of the Complaint.

134. Samsung denies the allegations in paragraph 134 of the Complaint.

135. Samsung denies the allegations in paragraph 135 of the Complaint.

136. The allegations in paragraph 136 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the allegations in paragraph 136 of the Complaint.

137. Samsung incorporates by reference its responses to the allegations in paragraphs 123 through 136 of the Complaint as set forth above. The remainder of the allegations in paragraph 137 of the Complaint express legal conclusions and thus no response is required. To the extent that a response is required, Samsung denies the remainder of the allegations in paragraph 137 of the Complaint.

138. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 138 of the Complaint and, on that basis, denies them.

139. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 139 of the Complaint and, on that basis, denies them.

140. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 140 of the Complaint and, on that basis, denies them.

141. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 141 of the Complaint and, on that basis, denies them.

[REDACTED]

142. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 142 of the Complaint and, on that basis, denies them.

**BACKGROUND OF DEFENDANTS' INFRINGING CONDUCT**

143. Samsung incorporates by reference its responses to the allegations set forth in paragraphs 1 through 142 as set forth above.

144. Samsung admits that SoCs are used in some consumer electronics or computing devices, including in some smartphones, laptops, tablets, and embedded systems. Samsung further admits SoCs are complex integrated circuits that may incorporate multiple processors, memory units, and interfaces onto a single chip.

145. Samsung admits that as SoCs have developed over time, additional processing cores and other IP blocks have been incorporated into SoCs. Samsung further admits that various interconnect technologies have been used in SoCs and that there are many factors that go into SoC system performance and size, one of which is interconnect design. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 145 of the Complaint and, on that basis, denies them.

146. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 146 of the Complaint and, on that basis, denies them.

147. Samsung admits that it is a leading semiconductor and electronic device company. Samsung incorporates by reference its responses to the allegations in paragraphs 21 through 22 as set forth above. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 147 of the Complaint and, on that basis, denies them.

148. Samsung incorporates its responses to the allegations in paragraph 147 of the Complaint as set forth above. Samsung lacks knowledge or information sufficient to form a belief

[REDACTED]

as to the truth of the remaining allegations in paragraph 148 of the Complaint and, on that basis, denies them.

149. Samsung denies the allegations in paragraph 149 of the Complaint.

150. Samsung admits that it makes, uses, sells, offers for sale, and/or imports, or has otherwise made, used, sold, offered for sale, and/or imported, Exynos SoCs and products containing Exynos SoCs, wherein some Exynos SoCs incorporate interconnect technology provided by Arteris.

151. Samsung denies the allegations in paragraph 151 of the Complaint.

152. Samsung denies the allegations in paragraph 152 of the Complaint.

153. Samsung denies the allegations in paragraph 153 of the Complaint.

154. Samsung admits that paragraph 154 of the Complaint purports to define “Samsung Accused Products” as those that constitute or include Samsung Exynos SoCs with Arteris interconnect technology and/or a derivative thereof, and/or include Qualcomm Snapdragon SoCs with Arteris interconnect technology and/or a derivative thereof. Samsung denies the remainder of the allegations in paragraph 154 of the Complaint.

155. Samsung denies the allegations in paragraph 155 of the Complaint.

156. Samsung admits that Plaintiff purports to set forth a comparison of claims of the Asserted Patents to certain Samsung products in Exhibits 7–25 to the Complaint. Samsung denies that its products infringe any valid and enforceable claim of any of the Asserted Patents.

157. Samsung denies the allegations in paragraph 157 of the Complaint.

158. Samsung admits that it markets, advertises, and/or offers to sell its products, including on its website. Samsung further admits that it publishes and distributes product

[REDACTED]

information for at least some of its products. Samsung denies the remaining allegations in paragraph 158 of the Complaint.

159. Samsung denies the allegations in paragraph 159 of the Complaint.

160. Samsung incorporates by reference its responses to the allegations in paragraphs 28, 40, 62, 85, 106, and 127 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 160 of the Complaint.

161. Samsung denies that any Samsung products incorporated into Lenovo products infringe any valid and enforceable claim of any of the Asserted Patents. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the remainder of the allegations in paragraph 161 of the Complaint and, on that basis, denies them.

162. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 162 of the Complaint and, on that basis, denies them.

163. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 163 of the Complaint and, on that basis, denies them.

164. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 164 of the Complaint and, on that basis, denies them.

165. Samsung admits that Plaintiff purports to set forth a comparison of claims of the Asserted Patents to certain Lenovo products in Exhibits 26–37 to the Complaint. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 165 of the Complaint and, on that basis, denies them.

166. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 166 of the Complaint and, on that basis, denies them.

[REDACTED]

167. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 167 of the Complaint and, on that basis, denies them.

168. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 168 of the Complaint and, on that basis, denies them.

169. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 169 of the Complaint and, on that basis, denies them.

170. Samsung denies that any Samsung products incorporated into OnePlus products infringe any valid and enforceable claim of any of the Asserted Patents. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the remainder of the allegations in paragraph 170 of the Complaint and, on that basis, denies them.

171. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 171 of the Complaint and, on that basis, denies them.

172. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 172 of the Complaint and, on that basis, denies them.

173. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 173 of the Complaint and, on that basis, denies them.

174. Samsung admits that Plaintiff purports to set forth a comparison of claims of the Asserted Patents to certain OnePlus products in Exhibits 38–43 to the Complaint. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 174 of the Complaint and, on that basis, denies them.

175. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 175 of the Complaint and, on that basis, denies them.

[REDACTED]

176. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 176 of the Complaint and, on that basis, denies them.

177. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 177 of the Complaint and, on that basis, denies them.

178. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 178 of the Complaint and, on that basis, denies them.

179. Samsung denies that its products infringe any valid and enforceable claim of any of the Asserted Patents. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 179 of the Complaint and, on that basis, denies them.

**COUNT I - INFRINGEMENT OF U.S. PATENT NO. 7,366,818**

180. Samsung incorporates by reference its responses to the allegations in paragraphs 1 through 179 of the Complaint as set forth above.

181. Samsung admits that Plaintiff purports to set forth comparisons of claim 1 of the '818 patent to certain Samsung Accused Products in Exhibits 7, 13, and 19 to the Complaint. Samsung denies the remaining allegations of paragraph 181 of the Complaint.

182. Samsung denies the allegations in paragraph 182 of the Complaint.

183. Samsung incorporates by reference its responses to the allegations in paragraph 158 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 183 of the Complaint.

184. Samsung incorporates by reference its responses to the allegations in paragraph 158 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 184 of the Complaint.

[REDACTED]

185. Samsung incorporates by reference its responses to the allegations in paragraph 158 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 185 of the Complaint.

186. Samsung incorporates by reference its responses to the allegations in paragraphs 24 through 35 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 186 of the Complaint.

187. Samsung denies the allegations in paragraph 187 of the Complaint.

188. Samsung denies the allegations in paragraph 188 of the Complaint.

189. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 189 of the Complaint and, on that basis, denies them.

190. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 190 of the Complaint and, on that basis, denies them.

191. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 191 of the Complaint and, on that basis, denies them.

192. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 192 of the Complaint and, on that basis, denies them.

193. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 193 of the Complaint and, on that basis, denies them.

194. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 194 of the Complaint and, on that basis, denies them.

195. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 195 of the Complaint and, on that basis, denies them.

[REDACTED]

196. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 196 of the Complaint and, on that basis, denies them.

197. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 197 of the Complaint and, on that basis, denies them.

198. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 198 of the Complaint and, on that basis, denies them.

199. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 199 of the Complaint and, on that basis, denies them.

200. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 200 of the Complaint and, on that basis, denies them.

201. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 201 of the Complaint and, on that basis, denies them.

202. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 202 of the Complaint and, on that basis, denies them.

203. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 203 of the Complaint and, on that basis, denies them.

204. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 204 of the Complaint and, on that basis, denies them.

**COUNT II - INFRINGEMENT OF U.S. PATENT NO. 7,373,449**

205. Samsung incorporates by reference its responses to the allegations in paragraphs 1 through 204 of the Complaint as set forth above.

206. Samsung admits that Plaintiff purports to set forth comparisons of claim 10 of the '449 patent to certain Samsung Accused Products in Exhibits 8, 14, and 20 to the Complaint. Samsung denies the remaining allegations of paragraph 206 of the Complaint.

[REDACTED]

207. Samsung denies the allegations in paragraph 207 of the Complaint.

208. Samsung incorporates by reference its responses to the allegations in paragraph 158 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 208 of the Complaint.

209. Samsung incorporates by reference its responses to the allegations in paragraph 158 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 209 of the Complaint.

210. Samsung incorporates by reference its responses to the allegations in paragraph 158 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 210 of the Complaint.

211. Samsung incorporates by reference its responses to the allegations in paragraphs 36 through 57 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 211 of the Complaint.

212. Samsung denies the allegations in paragraph 212 of the Complaint.

213. Samsung denies the allegations in paragraph 213 of the Complaint.

214. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 214 of the Complaint and, on that basis, denies them.

215. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 215 of the Complaint and, on that basis, denies them.

216. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 216 of the Complaint and, on that basis, denies them.

217. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 217 of the Complaint and, on that basis, denies them.

[REDACTED]

218. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 218 of the Complaint and, on that basis, denies them.

219. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 219 of the Complaint and, on that basis, denies them.

220. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 220 of the Complaint and, on that basis, denies them.

221. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 221 of the Complaint and, on that basis, denies them.

222. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 222 of the Complaint and, on that basis, denies them.

223. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 223 of the Complaint and, on that basis, denies them.

224. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 224 of the Complaint and, on that basis, denies them.

225. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 225 of the Complaint and, on that basis, denies them.

226. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 226 of the Complaint and, on that basis, denies them.

227. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 227 of the Complaint and, on that basis, denies them.

228. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 228 of the Complaint and, on that basis, denies them.

[REDACTED]

229. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 229 of the Complaint and, on that basis, denies them.

**COUNT III - INFRINGEMENT OF U.S. PATENT NO. 7,594,052**

230. Samsung incorporates by reference its responses to the allegations in paragraphs 1 through 229 of the Complaint as set forth above.

231. Samsung admits that Plaintiff purports to set forth comparisons of claim 6 of the '052 patent to certain Samsung Accused Products in Exhibits 9, 15, and 21 to the Complaint. Samsung denies the remaining allegations of paragraph 231 of the Complaint.

232. Samsung denies the allegations in paragraph 232 of the Complaint.

233. Samsung incorporates by reference its responses to the allegations in paragraph 158 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 233 of the Complaint.

234. Samsung incorporates by reference its responses to the allegations in paragraph 158 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 234 of the Complaint.

235. Samsung incorporates by reference its responses to the allegations in paragraph 158 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 235 of the Complaint.

236. Samsung incorporates by reference its responses to the allegations in paragraphs 58 through 80 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 236 of the Complaint.

237. Samsung denies the allegations in paragraph 237 of the Complaint.

238. Samsung denies the allegations in paragraph 238 of the Complaint.

[REDACTED]

239. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 239 of the Complaint and, on that basis, denies them.

240. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 240 of the Complaint and, on that basis, denies them.

241. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 241 of the Complaint and, on that basis, denies them.

242. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 242 of the Complaint and, on that basis, denies them.

243. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 243 of the Complaint and, on that basis, denies them.

244. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 244 of the Complaint and, on that basis, denies them.

245. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 245 of the Complaint and, on that basis, denies them.

246. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 246 of the Complaint and, on that basis, denies them.

247. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 247 of the Complaint and, on that basis, denies them.

248. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 248 of the Complaint and, on that basis, denies them.

249. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 249 of the Complaint and, on that basis, denies them.

[REDACTED]

250. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 250 of the Complaint and, on that basis, denies them.

251. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 251 of the Complaint and, on that basis, denies them.

252. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 252 of the Complaint and, on that basis, denies them.

253. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 253 of the Complaint and, on that basis, denies them.

254. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 254 of the Complaint and, on that basis, denies them.

**COUNT IV - INFRINGEMENT OF U.S. PATENT NO. 7,769,893**

255. Samsung incorporates by reference its responses to the allegations in paragraphs 1 through 254 of the Complaint as set forth above.

256. Samsung admits that Plaintiff purports to set forth comparisons of claim 4 of the '9893 patent to certain Samsung Accused Products in Exhibits 10, 16, and 22 to the Complaint. Samsung denies the remaining allegations of paragraph 256 of the Complaint.

257. Samsung denies the allegations in paragraph 257 of the Complaint.

258. Samsung incorporates by reference its responses to the allegations in paragraph 158 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 258 of the Complaint.

259. Samsung incorporates by reference its responses to the allegations in paragraph 158 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 259 of the Complaint.

[REDACTED]

260. Samsung incorporates by reference its responses to the allegations in paragraph 158 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 260 of the Complaint.

261. Samsung incorporates by reference its responses to the allegations in paragraphs 81 through 101 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 261 of the Complaint.

262. Samsung denies the allegations in paragraph 262 of the Complaint.

263. Samsung denies the allegations in paragraph 263 of the Complaint.

264. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 264 of the Complaint and, on that basis, denies them.

265. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 265 of the Complaint and, on that basis, denies them.

266. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 266 of the Complaint and, on that basis, denies them.

267. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 267 of the Complaint and, on that basis, denies them.

268. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 268 of the Complaint and, on that basis, denies them.

269. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 269 of the Complaint and, on that basis, denies them.

270. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 270 of the Complaint and, on that basis, denies them.

[REDACTED]

271. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 271 of the Complaint and, on that basis, denies them.

272. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 272 of the Complaint and, on that basis, denies them.

273. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 273 of the Complaint and, on that basis, denies them.

274. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 274 of the Complaint and, on that basis, denies them.

275. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 275 of the Complaint and, on that basis, denies them.

276. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 276 of the Complaint and, on that basis, denies them.

277. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 277 of the Complaint and, on that basis, denies them.

278. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 278 of the Complaint and, on that basis, denies them.

279. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 279 of the Complaint and, on that basis, denies them.

**COUNT V - INFRINGEMENT OF U.S. PATENT NO. 8,072,893**

280. Samsung incorporates by reference its responses to the allegations in paragraphs 1 through 279 of the Complaint as set forth above.

281. Samsung admits that Plaintiff purports to set forth comparisons of claim 1 of the '2893 patent to certain Samsung Accused Products in Exhibits 11, 17, and 23 for claim 1 to the Complaint. Samsung denies the remaining allegations of paragraph 281 of the Complaint.

[REDACTED]

282. Samsung admits that Plaintiff purports to set forth comparisons of claims 1 and 10 of the '2893 patent to certain Samsung Accused Products in Exhibits 11, 17, 23, and 24 to the Complaint. Samsung denies the remaining allegations of paragraph 282 of the Complaint.

283. Samsung denies the allegations in paragraph 283 of the Complaint.

284. Samsung incorporates by reference its responses to the allegations in paragraph 158 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 284 of the Complaint.

285. Samsung incorporates by reference its responses to the allegations in paragraph 158 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 285 of the Complaint.

286. Samsung incorporates by reference its responses to the allegations in paragraph 158 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 286 of the Complaint.

287. Samsung incorporates by reference its responses to the allegations in paragraph 158 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 287 of the Complaint.

288. Samsung incorporates by reference its responses to the allegations in paragraph 158 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 288 of the Complaint.

289. Samsung incorporates by reference its responses to the allegations in paragraph 158 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 289 of the Complaint.

[REDACTED]

290. Samsung incorporates by reference its responses to the allegations in paragraph 158 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 290 of the Complaint.

291. Samsung incorporates by reference its responses to the allegations in paragraphs 102 through 122 of the Complaint, as set forth above. Samsung denies the allegations in paragraph 291 of the Complaint.

292. Samsung denies the allegations in paragraph 292 of the Complaint.

293. Samsung denies the allegations in paragraph 293 of the Complaint.

294. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 294 of the Complaint and, on that basis, denies them.

295. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 295 of the Complaint and, on that basis, denies them.

296. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 296 of the Complaint and, on that basis, denies them.

297. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 297 of the Complaint and, on that basis, denies them.

298. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 298 of the Complaint and, on that basis, denies them.

299. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 299 of the Complaint and, on that basis, denies them.

300. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 300 of the Complaint and, on that basis, denies them.

[REDACTED]

301. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 301 of the Complaint and, on that basis, denies them.

302. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 302 of the Complaint and, on that basis, denies them.

303. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 303 of the Complaint and, on that basis, denies them.

304. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 304 of the Complaint and, on that basis, denies them.

305. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 305 of the Complaint and, on that basis, denies them.

306. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 306 of the Complaint and, on that basis, denies them.

307. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 307 of the Complaint and, on that basis, denies them.

308. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 308 of the Complaint and, on that basis, denies them.

309. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 309 of the Complaint and, on that basis, denies them.

**COUNT VI - INFRINGEMENT OF U.S. PATENT NO. 8,086,800**

310. Samsung incorporates by reference its responses to the allegations in paragraphs 1 through 309 of the Complaint as set forth above.

311. Samsung admits that Plaintiff purports to set forth comparisons of claim 10 of the '800 patent to certain Samsung Accused Products in Exhibits 12, 18, and 25 to the Complaint. Samsung denies the remaining allegations of paragraph 311 of the Complaint.

[REDACTED]

312. Samsung denies the allegations in paragraph 312 of the Complaint.

313. Samsung incorporates by reference its responses to the allegations in paragraph 158 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 313 of the Complaint.

314. Samsung incorporates by reference its responses to the allegations in paragraph 158 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 314 of the Complaint.

315. Samsung incorporates by reference its responses to the allegations in paragraph 158 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 315 of the Complaint.

316. Samsung incorporates by reference its responses to the allegations in paragraphs 123 through 142 of the Complaint, as set forth above. Samsung denies the remaining allegations in paragraph 316 of the Complaint.

317. Samsung denies the allegations in paragraph 317 of the Complaint.

318. Samsung denies the allegations in paragraph 318 of the Complaint.

319. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 319 of the Complaint and, on that basis, denies them.

320. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 320 of the Complaint and, on that basis, denies them.

321. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 321 of the Complaint and, on that basis, denies them.

322. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 322 of the Complaint and, on that basis, denies them.

[REDACTED]

323. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 323 of the Complaint and, on that basis, denies them.

324. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 324 of the Complaint and, on that basis, denies them.

325. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 325 of the Complaint and, on that basis, denies them.

326. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 326 of the Complaint and, on that basis, denies them.

327. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 327 of the Complaint and, on that basis, denies them.

328. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 328 of the Complaint and, on that basis, denies them.

329. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 329 of the Complaint and, on that basis, denies them.

330. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 330 of the Complaint and, on that basis, denies them.

331. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 330 of the Complaint and, on that basis, denies them.

332. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 332 of the Complaint and, on that basis, denies them.

333. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 333 of the Complaint and, on that basis, denies them.

[REDACTED]

334. Samsung lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 334 of the Complaint and, on that basis, denies them.

**ANSWER TO PRAYER FOR RELIEF**

Samsung denies that Plaintiff is entitled to any relief whatsoever in this action, either as requested in the Complaint or otherwise.

**RESPONSE TO JURY TRIAL DEMAND**

This sentence contains a demand for jury trial to which no response is required.

**AFFIRMATIVE AND OTHER DEFENSES**

Samsung asserts the following defenses in response to the allegations of the Complaint, undertaking the burden of proof only as to those defenses deemed affirmative defenses by law, regardless of how such defenses are denominated herein. Samsung further reserves all rights to allege additional defenses that become known through the course of discovery in this action or further investigation.

**FIRST AFFIRMATIVE DEFENSE**  
**(Failure to State A Claim)**

The Complaint fails to state a claim upon which relief can be granted as Plaintiff has failed to plead its claims with sufficient specificity or support to place Samsung on notice of Plaintiff's theories of alleged infringement with respect to the Asserted Patents.

**SECOND AFFIRMATIVE DEFENSE**  
**(No Infringement)**

Samsung does not infringe and has not infringed under any theory any valid and enforceable claim of the Asserted Patents, either literally or under the doctrine of equivalents, directly or indirectly, willfully or otherwise. Plaintiff's claims are further barred, in whole or in part, to the extent they are based on activities occurring outside the territorial reach of the United States patent laws.

[REDACTED]

**THIRD AFFIRMATIVE DEFENSE**  
**(Invalidity)**

The Asserted Patents are invalid for failure to satisfy one or more of the conditions for patentability specified in Title 35 of the United States Code, 35 U.S.C. §§ 101, 102, 103, 112 and/or 116, and the rules, regulations, and laws pertaining thereto.

**FOURTH AFFIRMATIVE DEFENSE**  
**(License)**

Plaintiff's claims are barred by express and/or implied licenses. For example, Plaintiff alleges that it is the owner by assignment of the Asserted Patents. *See, e.g.*, Compl. ¶¶ 2, 23.

Before the alleged assignment, [REDACTED]

[REDACTED]

[REDACTED]

[REDACTED] Accordingly, Plaintiff's infringement claims are barred, at least in part, by the doctrine of express license.

**FIFTH AFFIRMATIVE DEFENSE**  
**(Lack of Standing)**

To the extent that Plaintiff lacks all substantive rights to bring suit and to exclude others from practicing the claims of any of the Asserted Patents, Plaintiff's claims are barred by a lack of standing.

**SIXTH AFFIRMATIVE DEFENSE**  
**(Exhaustion)**

To the extent that Plaintiff's allegations of infringement relate to products that were provided by or for any licensee of the Asserted Patents and/or provided to Samsung by or through a licensee of the Asserted Patents or under a covenant not to sue, whether explicitly or impliedly, Plaintiff's claims are barred.

**SEVENTH AFFIRMATIVE DEFENSE**  
**(Prosecution History Estoppel and Disclaimer)**

Plaintiff's claims are barred in whole or in part by the doctrines of prosecution history estoppel and/or prosecution disclaimer based on statements, representations, and admissions made during the prosecution before the United States Patent and Trademark Office ("USPTO") of the patent families resulting in the Asserted Patents.

Plaintiff is estopped from construing the claims of the Asserted Patents to cover or include, either literally or under the doctrine of equivalents, products or methods that were surrendered because of arguments, amendments, admissions, representations, and/or statements made during prosecution or *inter partes* review before the USPTO.

**EIGHTH AFFIRMATIVE DEFENSE**  
**(Ensnarement)**

On information and belief, Plaintiff's claims are barred by the doctrine of ensnarement. Plaintiff is foreclosed from asserting infringement under the doctrine of equivalents to the extent that the scope of such equivalent would ensnare prior art.

**NINTH AFFIRMATIVE DEFENSE**  
**(Limitation on Damages)**

Plaintiff's claims for damages are statutorily limited under 35 U.S.C. §§ 286, 287, and 288. Plaintiff's claims are further barred, in whole or in part, to the extent they are based on activities occurring outside the territorial reach of the United States patent laws.

**TENTH AFFIRMATIVE DEFENSE**  
**(No Willfulness or Enhancement)**

Plaintiff is precluded from recovering enhanced damages under 35 U.S.C. § 284 at least because Samsung has not willfully infringed any valid and enforceable claim of the Asserted Patents. Samsung does not, did not, and could not subjectively or objectively believe that it

[REDACTED]

infringes any valid and enforceable claim of the Asserted Patents either literally or under the doctrine of equivalents.

**ELEVENTH AFFIRMATIVE DEFENSE**  
**(No Double Compensation)**

The relief that Plaintiff seeks is limited by any compensation it receives or seeks for its alleged patent rights from others whose products incorporate any accused Samsung products, or from others who supply products, to which the alleged patent rights relate, to Samsung.

For example, the relief that Plaintiff seeks is limited by any compensation received or sought by Plaintiff for its alleged patent rights from Qualcomm and/or Arteris.

**TWELFTH AFFIRMATIVE DEFENSE**  
**(No Injunctive Relief)**

No injunctive relief should be awarded because Plaintiff has not and cannot establish entitlement to such relief, and no factor supporting issuance of such relief is present in this case.

**THIRTEENTH AFFIRMATIVE DEFENSE**  
**(No Exceptional Case)**

Samsung has not engaged in any conduct that would make this an exceptional case or that would entitle Plaintiff to an award of attorneys' fees.

**FOURTEENTH AFFIRMATIVE DEFENSE**  
**(Improper Venue)**

Venue is not proper in the Eastern District of Texas for SSI and SAS under 28 U.S.C. § 1400(b). SSI and SAS do not reside in the Eastern District of Texas. SSI and SAS do not have a regular and established place of business in the Eastern District of Texas.

**FIFTEENTH AFFIRMATIVE DEFENSE**  
**(28 U.S.C. § 1498)**

Samsung is also not liable to the extent the Samsung Accused Products were provided, used or manufactured by or for the United States, or to the extent accused activities were undertaken on behalf of the United States, pursuant to 28 U.S.C. § 1498.

**SIXTEENTH AFFIRMATIVE DEFENSE**  
**(Unenforceability Due To Inequitable Conduct)**

For at least the reasons set forth below, and upon information and belief, the '818, '449, '052, '800, and '2,893 patents are each unenforceable as a result of inequitable conduct before the USPTO by the applicants and others associated with and substantively involved in the prosecution of those patents. The individuals substantively involved with the filing and prosecution of one or more of the '818, '449, '052, '800, and '2,893 patents that owed duties of disclosure under 37 C.F.R. § 1.56 include, for example, at least the applicants—*i.e.*, named inventors Kees Gerard Willem Goossens, Andrei Rădulescu, John Dielissen, and Edwin Rijpkema—as well as others substantively involved with the prosecution—*e.g.*, David L. Barnes, Michael E. Belk, Edward M. Blocker, Carrie Anne Colby, Yan Glickberg, Edward W. Goodman, Andrew C. Gust, Dicran Halajian, Jack E. Haken, Frank Keegan, Michael E. Marion, and Michael A. Scaturro. On information and belief, these applicants and others substantively involved with the prosecution of the '818, '449, '052, '800, and '2,893 patents had knowledge of prior art—including printed publications written by the applicants themselves—material to the patentability of the claimed inventions in those patents, failed to disclose that prior art during prosecution by intentionally withholding and concealing it with the specific intent to mislead or deceive the USPTO, and thereby wrongly secured the '818, '449, '052, '800, and '2,893 patents that otherwise would not have issued but for their misconduct.

[REDACTED]

### **The Applicants Authored and Published Material Prior Art References**

The applicants of the '818, '449, '052, '800, and '2,893 patents authored and published many material prior art references, most of which they failed to disclose to the USPTO during prosecution. Some of the applicant-authored prior art references are identified below.

**Goossens I.** No later than March 8, 2002, named inventor Kees Gerard Willem Goossens and several co-authors published a paper in the IEEE journal *Proceedings of the 2002 Design, Automation and Test in Europe Conference and Exhibition* titled “Networks on Silicon: Combining Best-Effort and Guaranteed Services” (hereinafter “Goossens I”), attached as Exhibit A.

**Goossens II.** No later than January 31, 2003, named inventors Kees Gerard Willem Goossens and Andrei Rădulescu, and several additional co-authors, published a book chapter in NETWORKS ON CHIP (eds. A. Jantsch and H. Tenhunen) titled “Guaranteeing the Quality of Services in Networks on Chip” (hereinafter “Goossens II”), attached as Exhibit B.

**Rijpkema I.** No later than September 2003, named inventors Kees Gerard Willem Goossens and Andrei Rădulescu, and several co-authors, including Edwin Rijpkema, published a paper in Volume 150, Issue No. 5 of the journal *IEE Proceedings – Computer and Digital Techniques* titled “Trade Offs in the Design of a Router with Both Guaranteed and Best-Effort Services for Networks on Chip” (hereinafter “Rijpkema I”), attached as Exhibit C.

**Rijpkema II.** No later than September 12, 2003, named inventors Kees Gerard Willem Goossens and Andrei Rădulescu, and several co-authors, including Edwin Rijpkema, published a paper titled “Trade Offs in the Design of a Router with Both Guaranteed and Best-Effort Services for Networks on Chip (Extended Version)” (hereinafter “Rijpkema II”), attached as Exhibit D.

[REDACTED]

**Rijpkema III.** No later than March 3, 2003, named inventors Kees Gerard Willem Goossens and Andrei Rădulescu, and several co-authors, including Edwin Rijpkema, published a paper with *IEEE* titled “Trade Offs in the Design of a Router with Both Guaranteed and Best-Effort Services for Networks on Chip” (hereinafter “Rijpkema III”), attached as Exhibit E.

**Rijpkema IV.** No later than November 1, 2001, named inventor Kees Gerard Willem Goossens and several co-authors, including Edwin Rijpkema, published a paper with *Proceedings of Progress 2001, 2nd Workshop on Embedded Systems* titled “A Router Architecture for Networks on Silicon” (hereinafter “Rijpkema IV”), attached as Exhibit F.

**Rădulescu I.** No later than February 20, 2004, named inventors Andrei Rădulescu and Kees Gerard Willem Goossens, and several co-authors, published a paper in the IEEE journal *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition* titled “An Efficient On-Chip Network Interface Offering Guaranteed Services, Shared-Memory Abstraction, and Flexible Network Configuration” (hereinafter “Rădulescu I”), attached as Exhibit G.

**Rădulescu II.** No later than November 11, 2003, named inventors Andrei Rădulescu and Kees Gerard Willem Goossens published a book chapter in *DOMAIN-SPECIFIC PROCESSORS: SYSTEMS, ARCHITECTURES, MODELING, AND SIMULATION* (eds. S. Bhattacharyya, E. Deprettere, J. Teich) titled “Communication Services for Networks on Chip” (hereinafter “Rădulescu II”), attached as Exhibit H.

**Rădulescu WO.** On April 22, 2004, International Patent Application No. PCT/IB2003/004414 filed by named inventors Andrei Rădulescu and Kees Gerard Willem Goossens on October 7, 2003 was published as WO 2004/034176 A2 (hereinafter “Rădulescu WO”), attached as Exhibit I.

[REDACTED]

**Wielage.** No later than September 6, 2002, named inventor Kees Gerard Willem Goossens and co-author Paul Wielage published a paper in *Proceedings Euromicro Symposium on Digital System Design. Architectures, Methods and Tools* titled “Networks on Silicon: Blessing or Nightmare?” (hereinafter “Wielage”), attached as Exhibit J.

**Vermeulen.** No later than September 15, 2003, named inventor Kees Gerard Willem Goossens and several co-authors, including Bart Vermeulen, published a paper in Volume 41, Issue No. 9 of the journal *IEEE Communications Magazine* titled “Bringing Communication Networks on a Chip: Test and Verification Implications” (hereinafter “Vermeulen”), attached as Exhibit K.

**Dielissen.** No later than January 11, 2004, named inventors Andrei Rădulescu and Kees Gerard Willem Goossens, and several co-authors, including John Dielissen, published a paper titled “Concepts and Implementation of the Philips Network-on-Chip” (hereinafter “Dielissen”), attached as Exhibit L.

**Pestana.** No later than February 20, 2004, named inventors Andrei Rădulescu and Kees Gerard Willem Goossens, and several co-authors, including Santiago Gonzalez Pestana, published a paper in the IEEE journal *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition* titled “Cost-Performance Trade-Offs in Networks on Chip: A Simulation-Based Approach” (hereinafter “Pestana”), attached as Exhibit M.

#### **'818 Patent**

According to the face of U.S. Patent No. 7,366,818, it issued on April 29, 2008 from U.S. Application No. 10/530,266 (“the '266 Application”), which was filed by Kees Gerard Willem Goossens and Andrei Rădulescu on April 5, 2005. The '266 Application purports to be a national stage application, pursuant to 35 U.S.C. § 371, of International Application No. PCT/IB03/03035,

[REDACTED]

which had an international filing date of July 4, 2003. The '266 Application claimed priority to European Patent Application No. 02079196, which was filed on October 8, 2002. The transmittal letter filed with the '266 Application was signed by attorney David Barnes. Submitted with the '266 Application was an executed power of attorney form signed by Kees Gerard Willem Goossens and Andrei Rădulescu, which was dated May 6, 2004 and appointed Jack E. Haken, Michael E. Marion, and Edward M. Blocker as the attorneys and/or agents to prosecute the '266 Application and transact all business in the USPTO connected with that patent application. Kees Gerard Willem Goossens and Andrei Rădulescu further declared in the signed May 6, 2004 document that they “acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, § 1.56(a).”

During the prosecution of the '266 Application, the Examiner issued several rejections over various prior art references. The applicants through their prosecuting attorneys and/or agents—including Carrie Anne Colby “for David Barnes”—responded to the rejections by amending the independent claim and arguing that the cited prior art failed to disclose elements of the claims, including a “network on chip” and an “interface means” comprising a “dropping means (DM) for dropping data.” In a Notice of Allowance issued on September 28, 2007, the Examiner stated the claims were in his view “allowable over the prior art of record because” it did not “teach the interface means comprises a first dropping means for dropping data and wherein the dropping of data and therefore the transaction completion can be controlled by the interface means.”

On information and belief, despite having and being aware of a duty to disclose prior art material to patentability, the applicants, prosecuting attorneys, and/or their agents failed to disclose to the USPTO during prosecution of the '266 Application prior art bearing on the exact same

[REDACTED]

subject matter as the patent application and material to patentability. This includes certain references authored by the applicants themselves and references authored by others of which they were aware. These include publications particularly relevant to, among other things, the claimed “network on chip” and “network interface” comprising a “dropping means (DM) for dropping data” recited in every issued claim of the ’266 Application (the “dropping means limitation”).

For example, Goossens I was published no later than March 8, 2002 and thus constitutes prior art under at least 35 U.S.C. § 102(b) because it was published more than one year before the date of the filing of the ’266 Application in the United States. Goossens I discloses “systems on silicon (SOSs)” comprising hundreds of IPs and a “network on silicon (NOS) to implement the communication between these IPs,” as well as “network interfaces” that “implement multipath and multi-channel ordering, and channel flow control.” Ex. A (Goossens I) at 1, 3.

As a further example, Rijpkema IV was published no later than November 1, 2001 and thus constitutes prior art under at least 35 U.S.C. § 102(b) because it was published more than one year before the date of the filing of the ’266 Application in the United States. Rijpkema IV discloses “a cost-effective router and network suitable for on-chip integration” that enables “communication between [] IP blocks and between IP blocks and memory,” as well as “network interfaces that connect the network to the sources and sinks.” Ex. F (Rijpkema IV) at 1, 5.

As another example, and without limitation, Raoul A.F. Bhoedjang and several co-authors published a paper in Volume 31, Issue No. 11 of the journal *Computer* titled “User-Level Network Interface Protocols” (“Bhoedjang”), attached as Exhibit O. Bhoedjang was published no later than November 1998 and thus constitutes prior art under at least 35 U.S.C. § 102(b) because it was published more than one year before the date of the filing of the ’266 Application in the United

[REDACTED]

States. Bhoedjang discloses “receiving network interface[s]” that “drop incoming packets” in the case of “buffer overflow.” Ex. O (Bhoedjang) at 55–56, 59.

Goossens I, Rijpkema IV, and Bhoedjang are material to the patentability of the asserted claims as they disclose, among other things, the claimed “network on chip” and “interface means” comprising a “dropping means (DM) for dropping data,” which the applicants alleged were missing from the prior art and purportedly made the claims of the ’266 Application patentable. The disclosures in Goossens I, Rijpkema IV, and Bhoedjang are not cumulative of the disclosure in the prior art disclosed by the applicants, cited by the Examiner, or considered during prosecution of the ’266 Application. Further, when combined with one another and/or other secondary prior art references, such as but not limited to “PM: An Operating System Coordinated High Performance Communication Library” by Hiroshi Tezuka et al., the claims of the ’266 Application are unpatentable.

Kees Gerard Willem Goossens and Andrei Rădulescu had knowledge of Goossens I and Rijpkema IV before filing and during prosecution of the ’266 Application at least because they co-authored those publications and would have known they were published before the filing date of their own ’266 Application. Kees Gerard Willem Goossens and Andrei Rădulescu also had knowledge of Bhoedjang at least during prosecution of the ’266 Application at least because they co-authored a January 1, 2005 publication titled “An Efficient On-Chip NI Offering Guaranteed Services, Shared-Memory Abstraction, and Flexible Network Configuration,” which itself cites Bhoedjang, and would have known that Bhoedjang was published before the filing date of their own ’266 Application. On information and belief, the prosecuting attorneys and/or their agents, including at least those identified above, also would have had knowledge of Goossens I, Rijpkema IV, and Bhoedjang before filing and/or during prosecution of the ’266 Application at least through

[REDACTED]

the applicants and/or through their own independent research on known prior art relating to the subject matter of the '266 Application. Despite this knowledge, Goossens I, Rijpkema IV, and Bhoedjang were withheld from all USPTO filings associated with the '266 Application.

On information and belief, the withholding by the applicants, prosecuting attorneys, and/or their agents of information material to patentability, including at least Goossens I, Rijpkema IV, and Bhoedjang, during prosecution of the '266 Application was done with specific intent to mislead and/or deceive the USPTO. All three papers, two of which were written by the applicants themselves, are directed to the same subject matter as the claimed invention and discuss a “network on chip” and/or an “interface means” comprising a “dropping means (DM) for dropping data.” The withholding of these publications left the USPTO without any prior art references before the Examiner during prosecution of the '266 Application that disclosed the dropping means limitation, the sole basis identified for the alleged patentability of the issued claims. On information and belief, the applicants, prosecuting attorneys, and/or their agents knew that the withheld prior art, including at least Goossens I, Rijpkema IV, and Bhoedjang, were material to the patentability of the claims of the '266 Application and made this choice to present the misleading and deceptive argument that the claimed “network on chip” and the dropping means limitation were supposedly missing from the prior art and purportedly made the claims of the '266 Application patentable. On information and belief, the applicants, prosecuting attorneys, and/or their agents made the deliberate decision to withhold these publications during prosecution of the '266 Application because disclosing them would have been adverse to issuance of the desired patent claims in that application.

On information and belief, but for the applicants, prosecuting attorneys, and/or their agents intentionally and improperly withholding from the USPTO during prosecution of the '266

[REDACTED]

Application prior art references material to patentability (including at least Goossens I, Rijkema IV, and Bhoedjang), the '818 patent would not have issued. As a result, the '818 patent is unenforceable due to inequitable conduct.

#### **'449 Patent**

According to the face of U.S. Patent No. 7,373,449, it issued on May 13, 2008 from U.S. Application No. 10/530,267 (“the '267 Application”), which was filed by Kees Gerard Willem Goossens and Andrei Rădulescu on April 5, 2005. The '267 Application purports to be a national stage application, pursuant to 35 U.S.C. § 371, of International Application No. PCT/IB03/004414, which had an international filing date of October 7, 2003. The '267 Application claimed priority to European Patent Application No. 02079196, which was filed on October 8, 2002. The transmittal letter and preliminary amendment filed with the '267 Application were each signed by attorney David Barnes. Submitted with the '267 Application was an executed power of attorney form signed by Kees Gerard Willem Goossens and Andrei Rădulescu, which was dated May 6, 2004 and appointed Jack E. Haken, Michael E. Marion, and Edward M. Blocker as the attorneys and/or agents to prosecute the '267 Application and transact all business in the USPTO connected with that patent application. Kees Gerard Willem Goossens and Andrei Rădulescu further declared in the signed May 6, 2004 document that they “acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, § 1.56(a).”

During the prosecution of the '267 Application, the Examiner issued several rejections over various prior art references. The applicants through their prosecuting attorneys and/or agents—including Andrew C. Gust “for David Barnes”—responded to the rejections by amending the independent claims, adding new dependent claims, and arguing that the cited prior art failed to

[REDACTED]

disclose elements of the claims, including an “integrated circuit” and a “resource manager.” In a Notice of Allowance issued on January 9, 2008, the Examiner stated the claims were in his view allowable “over the prior art of record because the prior arts, cited in its entirety, or in combination, do not teach” the claimed “resource manager,” which “determines whether the at least one connection with the communication channels is available based on the connection properties being available.”

On information and belief, despite having and being aware of a duty to disclose prior art material to patentability, at no point during prosecution of the '267 Application did the applicants, prosecuting attorneys, and/or their agents, disclose to the USPTO *any* of the applicants' publications as prior art even though they bear on the exact same subject matter as the patent application and are material to patentability. These include publications particularly relevant to, among other things, the claimed “resource manager” recited in every issued claim of the '267 Application.

For example, Goossens I was published no later than March 8, 2002 and thus constitutes prior art at least under 35 U.S.C. § 102(b) because it was published more than one year before the date of the filing of the '267 Application in the United States. Goossens I discloses “systems on silicon (SOSs)” comprising IPs that perform “computation” and a “network on silicon (NOS)” for “implement[ing] the communication between these IPs” and “end-to-end connection management,” that includes a resource manager identified as a “quality-of-service manager.” Ex. A (Goossens I) at 1–3.

As another example, and without limitation, Goossens II was published no later than January 31, 2003 and thus constitutes prior art at least under 35 U.S.C. § 102(a) because it was published before the alleged invention of the '267 Application, as reflected in that application's

[REDACTED]

October 7, 2003 filing date. While the '267 Application claims priority to European Patent Application No. 02079196.8, which was filed on October 8, 2002, that application fails to disclose the claimed “resource manager” (among other elements) that was alleged by the applicants to be missing from the prior art and purportedly made the claims of the '267 Application patentable and thus cannot support an earlier priority date. Goossens II discloses a “system on a chip (SOC)” comprising IPs and a “network[] on chip (NOC)” for implementing “communication between one master [IP] and one or more slaves [IPs]” and “resource management.” Ex. B (Goossens II) at Abstract, 61, 71–72, 74; *see also id.* at 65 (disclosing “[r]esource management” to facilitate “predictable QoS”); *id.* at 67 (“Effective steering of the limited resources in a NOC, and hence effective QoS, is contingent on a reliable reaction of resources to instructions.” (citation omitted)); *id.* at 68 (“Moreover, SOC’s can be more robust, because rather than relying on cooperation to share resources, resource management enforces the contracts between service providers and users.”).

As a further example, Rijpkema IV was published no later than November 1, 2001 and thus constitutes prior art at least under 35 U.S.C. § 102(b) because it was published more than one year before the date of the filing of the '267 Application in the United States. Rijpkema IV discloses “a cost-effective router and network suitable for on-chip integration.” Ex. F (Rijpkema IV) at 1. Rijpkema IV discloses a process of managing resources—such as reserving slots to allow guaranteed throughput and using virtual-output-queuing to permit logical queues for buffering data—to facilitate communication over its on-chip network in “a cost-effective” manner. *Id.* at 1, 5–7.

Goossens I, Goossens II, and Rijpkema IV are material to patentability of claims of the '267 Application because the teachings in those withheld references are relevant to the claimed

[REDACTED]

“resource manager,” which the applicants alleged was missing from the prior art and purportedly made the claims of the ’267 Application patentable. The disclosures in Goossens I, Goossens II, and Rijpkema IV are not cumulative of the disclosure in the prior art disclosed by the applicants, cited by the Examiner, or considered during prosecution of the ’267 Application. Further, as explained in IPR petitions, when combined with secondary prior art references such as but not limited to U.S. Patent No. 5,461,611 to Drake, Jr. *et al.*, the claims of the ’276 Application are unpatentable. For example, see Petition for *Inter Partes* Review, *Samsung Elecs. Co., Ltd. v. Network Sys. Techs., LLC*, IPR2024-00338, at 13–14, 48–68 (P.T.A.B. Dec. 18, 2023), which is incorporated by reference herein.

Kees Gerard Willem Goossens had knowledge of Goossens I, Goossens II, and Rijpkema IV before filing and during prosecution of the ’267 Application at least because he co-authored those publications and would have known they were published before the filing date of his own ’267 Application. Andrei Rădulescu also had knowledge of Goossens I, Goossens II, and Rijpkema IV before filing and during prosecution of the ’267 Application at least because he co-authored Goossens II, which itself cites Goossens I and Rijpkema IV, and would have known they were published before the filing date of his own ’267 Application. On information and belief, the prosecuting attorneys and/or their agents, including at least those identified above, also would have had knowledge of Goossens I, Goossens II, and Rijpkema IV before filing and during prosecution of the ’267 Application at least through the applicants and/or through their own independent research on known prior art relating to the subject matter of the ’267 Application. Despite this knowledge, Goossens I, Goossens II, and Rijpkema IV were withheld from all USPTO filings associated with the ’267 Application.

[REDACTED]

On information and belief, the withholding by the applicants, prosecuting attorneys, and/or their agents of information material to patentability, including at least Goossens I, Goossens II, and Rijpkema IV, during prosecution of the '267 Application was done with specific intent to mislead and/or deceive the USPTO. All three papers, written by the applicants themselves, are directed to the same subject matter as the claimed invention and discuss resource management in the context of networks on chip. The withholding of these publications left the USPTO without any prior art references before the Examiner during prosecution of the '267 Application that disclosed the claimed resource manager, the sole basis identified for the alleged patentability of the issued claims. On information and belief, the applicants, prosecuting attorneys, and/or their agents knew that the withheld prior art, including at least Goossens I, Goossens II, and Rijpkema IV, were material to the patentability of the claims of the '267 Application and made this choice to present the misleading and deceptive argument that the claimed resource manager was supposedly missing from the prior art and purportedly made the claims of the '267 Application patentable. On information and belief, the applicants, prosecuting attorneys, and/or their agents made the deliberate decision to withhold the applicants' prior art publications, including at least Goossens I, Goossens II, and Rijpkema IV, during prosecution of the '267 Application because it would have been adverse to issuance of the desired patent claims in that application.

On information and belief, but for the applicants, prosecuting attorneys, and/or their agents intentionally and improperly withholding from the USPTO during prosecution of the '267 Application prior art references material to patentability (including at least Goossens I, Goossens II, and Rijpkema IV), the '449 patent would not have issued. As a result, the '449 patent is unenforceable due to inequitable conduct.

### **'052 Patent**

According to the face of U.S. Patent No. 7,594,052, it issued on September 22, 2009 from U.S. Application No. 10/598,795 (“the ’795 Application”), which was filed by Kees Gerard Willem Goossens and Andrei Rădulescu on September 12, 2006. The ’795 Application purports to be a national stage application, pursuant to 35 U.S.C. § 371, of International Application No. PCT/IB2005/050833, which had an international filing date of March 7, 2005. The ’795 Application claimed priority to European Patent Application No. 04101095, which was filed on March 17, 2004. The application data sheet filed with the ’795 Application was signed by Michael E. Marion, and the preliminary amendment filed with that application was signed by Edward W. Goodman. Submitted with the ’795 Application was an executed power of attorney form signed by Kees Gerard Willem Goossens and Andrei Rădulescu, which was dated October 17, 2005 and appointed Jack E. Haken, Michael E. Marion, and Edward M. Blocker as the attorneys and/or agents to prosecute the ’795 Application and transact all business in the USPTO connected with that patent application. Kees Gerard Willem Goossens and Andrei Rădulescu further declared in the signed October 17, 2005 document that they “acknowledge the duty to disclose information which is material to patentability as defined in 37 CFR 1.56.”

During prosecution of the ’795 Application, the Examiner issued several rejections over the Æthereal NoC disclosed in “Concepts and Implementation of the Philips Network-on-Chip” by Dielissen et al. (Nov. 2003) and “Communication Services for Networks on Chip” by Rădulescu et al. (2000). The applicants through their prosecuting attorneys and/or agents—including Michael A. Scaturro—responded to the rejections by amending the independent claims and arguing that the cited prior art failed to disclose elements of the claims, including “mapping the requested at least one communication service based on said specific communication properties

[REDACTED]

to a connection based on a set of connection properties according to said at least one communication service identification,” where the communication service identification comprises “at least one communication thread or at least one address range” (the “mapping limitation”). Notably, pursuant to 37 C.F.R. § 1.105, the Examiner also required the applicants and the assignee (Koninklijke Philips Electronics, N.V.) to “provide the names of any products or services that have incorporated the claimed subject matter.” The applicants through their prosecuting attorneys and/or agents represented to the USPTO that “there are no products or services that have incorporated the claimed subject matter.” Despite this representation, on information and belief, the applicants knew and failed to disclose that they had at least developed and publicly presented on the Æthereal NoC before 2009.

On information and belief, despite having and being aware of a duty to disclose prior art material to patentability, at no point during prosecution of the '795 Application did the applicants, prosecuting attorneys, and/or their agents, disclose to the USPTO *any* of the applicants' publications as prior art even though they bear on the exact same subject matter as the patent application and are material to patentability. These include publications particularly relevant to, among other things, the claimed mapping limitation, which is recited in every issued claim of the '795 Application.

For example, Rădulescu I was published no later than February 20, 2004 and thus constitutes prior art at least under 35 U.S.C. § 102(b) because it was published more than one year before the date of the filing of the '795 Application in the United States. Rădulescu I discloses an on-chip “network interface” that “is intended for systems on chip (SoC).” Ex. G (Rădulescu I) at Abstract, 1. Specifically, Rădulescu I discloses the Æthereal NoC, including the mapping limitation. *See, e.g., id.* (“Communication is performed using a transaction-based protocol” that

[REDACTED]

allows “master IP modules” to “issue *request messages* (e.g., read and write commands at an address, possibly carrying data) that are executed by the addressed slave modules, which may respond with a *response message* (i.e., status of the command execution, and possibly data).”); *id.* at Abstract, 1–2 (disclosing network interface that “provides both guaranteed and best-effort services via connections,” “[c]onnections are composed of ... *channels*” to which “*properties* are attached” such as “guaranteed message delivery or not, in order or un-ordered message delivery, and with or without timing guarantees,” and exemplary “guarantees are lower bounds on throughput” and “upper bounds on latency”); *id.* at 2 (“Support for real-time communication is achieved by providing throughput, latency and jitter *guarantees*.”); *id.* at 4 (explaining transactions over connections may be “executed by a single slave selected based on the address provided in the transaction,” where “[t]he address range assigned to a slave is configurable” and “the slave for which the transaction is destined is selected based on the address (Conn block)”).

Additional examples of withheld prior art references material to patentability include Goossens I, Goossens II, Rijpkema IV, Wielage, and Pestana. Each of these references constitutes prior art at least under 35 U.S.C. § 102(b) because they were all published more than one year before the date of the filing of the ’795 Application in the United States. Moreover, each discloses NoCs that allow connection-based communication and properties of those communications, among other things. *See, e.g.*, Ex. A (Goossens I) at 1, 2 (disclosing NoC that provides “*end-to-end services over connections*”); Ex. B (Goossens II) at Abstract, 61, 71–72, Fig. 4.5 (similar, including transactions performed “on an address space that is mapped on two memory modules”); Ex. F (Rijpkema IV) at 1–3 (disclosing an NoC in which data communication is facilitated using data packets with headers containing, among other things, a “destination address”); Ex. J (Wielage) at Abstract, 2–3 (disclosing an NoC with communication of data packets having “routing

[REDACTED]

information in their headers” over links optimized based on performance requirements); Ex. M (Pestana) at 2–4 (disclosing a network interface architecture in which data packets containing “route information” is transferred over the network and “[d]ifferent applications using different services can be mapped without changing the system core,” and disclosing a “mapping file” that describes “which [IP] blocks are connected to which network interface”). *Id.* at 2–4.

Rădulescu I, Goossens I, Goossens II, Rijpkema IV, Wielage, and Pestana are material to the patentability of the claims of the ’795 Application as the teachings in those withheld references are relevant to, among other things, the claimed mapping limitation, which the applicants alleged was missing from the prior art and purportedly made the claims of the ’795 Application patentable. The disclosures in Rădulescu I, Goossens I, Goossens II, Rijpkema IV, Wielage, and Pestana are not cumulative of the disclosure in the prior art disclosed by the applicants, cited by the Examiner, or considered during prosecution of the ’795 Application.<sup>1</sup> For example, Rădulescu I discloses every limitation of, anticipates, and thus renders unpatentable the claims of the ’795 Application. See Petition for *Inter Partes* Review, *Qualcomm Inc. v. Network Sys. Techs., LLC*, IPR2024-00343, at 20–47, 68–76 (P.T.A.B. Dec. 19, 2023), which is incorporated by reference herein. Further, as another example, Rădulescu I, Goossens I, Goossens II, Rijpkema IV, Wielage, and Pestana, when combined with secondary prior art references such as but not limited to Release 1.0 of the Open Core Protocol Specification by OCP-IP Association in 2001 (“OCP Specification”), would have also blocked issuance of the patent claims of the ’795 Application.

---

<sup>1</sup> Although the Examiner noted a January 1, 2005 publication by the applicants titled “An Efficient On-Chip NI offering Guaranteed Services, Shared-Memory Abstraction, and Flexible Network Configuration,” that reference was published after the filing date of European Patent Application No. 04101095.0, to which the ’795 Application claimed priority, and less than one year before the ’795 Application was filed, so it does not constitute prior art under 35 U.S.C. §§ 102(a) or (b), and thus would not have been considered by the Examiner as an invalidating (including anticipatory) reference on that ground.

[REDACTED]

Kees Gerard Willem Goossens had knowledge of Rădulescu I, Goossens I, Goossens II, Rijpkema IV, Wielage, and Pestana before filing and during prosecution of the '795 Application at least because he co-authored those publications and would have known they were published before the filing date of his own '795 Application. Andrei Rădulescu also had knowledge of Rădulescu I, Goossens I, Goossens II, Rijpkema IV, Wielage, and Pestana before filing and during prosecution of the '795 Application at least because he co-authored Rădulescu I, Goossens II, and Pestana, Goossens II itself cites Goossens I, Rijpkema IV, and Wielage, and thus he would have known they were published before the filing date of his own '795 Application. On information and belief, the prosecuting attorneys and/or agents for the '795 Application, including at least those identified above, also would have had knowledge of Rădulescu I, Goossens I, Goossens II, Rijpkema IV, Wielage, and Pestana before filing and during prosecution of the '795 Application at least through the applicants and/or through their own independent research on known prior art relating to the subject matter of the '795 Application. Despite this knowledge, Rădulescu I, Goossens I, Goossens II, Rijpkema IV, Wielage, and Pestana were withheld from all USPTO filings associated with the '795 Application.

On information and belief, the withholding by the applicants, prosecuting attorneys, and/or their agents of information material to patentability, including at least Rădulescu I, Goossens I, Goossens II, Rijpkema IV, Wielage, and Pestana, during prosecution of the '795 Application was done with specific intent to mislead and/or deceive the USPTO. The Rădulescu I paper, written by the applicants themselves, is directed to the same subject matter as the claimed invention and discusses—among all the other claim limitations—communication service identification in which an address range identifies a second processing module. Goossens I, Goossens II, Rijpkema IV, Wielage, and Pestana were also written by one or more of the applicants and are directed to the

[REDACTED]

same subject matter as the claimed invention. The withholding of these publications left the USPTO without any prior art references before the Examiner during prosecution of the '795 Application that disclosed the claimed communication service identifiers in the context of the claimed on-chip network having all the claimed limitations. On information and belief, the applicants, prosecuting attorneys, and/or their agents knew that the withheld prior art, including at least Rădulescu I, Goossens I, Goossens II, Rijpkema IV, Wielage, and Pestana, was material to the patentability of the claims of the '795 Application and made this choice to present the misleading and deceptive argument that the claimed invention—including but not limited to the mapping limitation—was supposedly missing from the prior art and purportedly made the claims of the '795 Application patentable. On information and belief, the applicants, prosecuting attorneys, and/or their agents made the deliberate decision to withhold the applicants' prior art publications, including at least Rădulescu I, Goossens I, Goossens II, Rijpkema IV, Wielage, and Pestana, during prosecution of the '795 Application because it would have been adverse to issuance of the desired patent claims in that application.

On information and belief, but for the applicants, prosecuting attorneys, and/or their agents intentionally and improperly withholding from the USPTO during prosecution of the '795 Application prior art references material to patentability (including at least Rădulescu I, Goossens I, Goossens II, Rijpkema IV, Wielage, and Pestana), the '052 patent would not have issued. As a result, the '052 patent is unenforceable due to inequitable conduct.

### **'800 Patent**

According to the face of U.S. Patent No. 8,086,800, it issued on December 27, 2011 from U.S. Application No. 11/569,083 ("the '083 Application"), which was filed by Kees Gerard Willem Goossens and Andrei Rădulescu on November 14, 2006. The '083 Application purports

[REDACTED]

to be a national stage application, pursuant to 35 U.S.C. § 371, of International Application No. PCT/IB2005/051580, which had an international filing date of May 13, 2005. The '083 Application claimed priority to European Patent Application No. 04102189, which was filed on May 18, 2004. The application data sheet filed with the '083 Application was signed by Michael E. Marion and the preliminary amendment filed with that application was signed by Edward W. Goodman. Submitted with the '083 Application was an executed power of attorney form signed by Kees Gerard Willem Goossens and Andrei Rădulescu, which was dated December 8, 2005 and appointed Jack E. Haken, Michael E. Marion, and Edward M. Blocker as the attorneys and/or agents to prosecute the '083 Application and transact all business in the USPTO connected with that patent application. Kees Gerard Willem Goossens and Andrei Rădulescu further declared in the signed December 8, 2005 document that they “acknowledge the duty to disclose information which is material to patentability as defined in 37 C.F.R. 1.56.”

During the prosecution of the '083 Application, the Examiner issued several rejections over various prior art references. The applicants through their prosecuting attorneys and/or agents—including Dicran Halajian—responded to the rejections by amending the claims, adding new claims, and arguing that the cited prior art failed to disclose elements of the claims, including a “determination unit for determining an optimal first and second amount of data to be buffered.” In a Notice of Allowance issued on May 11, 2011, the Examiner stated the claims were in his view allowable because “[t]he prior art found alone or in combination does not teach or suggest that the communication properties of the communication between the master and slave used to determine the optimal moment for sending data, incorporate the claimed list of communication properties to make that determination” (the “optimal moment” and “communication properties” limitations).

[REDACTED]

On information and belief, despite having and being aware of a duty to disclose prior art material to patentability, at no point before allowance of the '083 Application did the applicants, prosecuting attorneys, and/or their agents, disclose to the USPTO *any* of the applicants' publications as prior art even though they bear on the exact same subject matter as the patent application and are material to patentability. Nor did they disclose the vast majority of the applicants' publications after the '083 Application was already allowed. These include publications particularly relevant to, among other things, the claimed "communication properties" and "optimal moment" limitations as recited in every issued claim of the '083 Application. The disclosures in Rădulescu I, Rădulescu WO, Rijpkema I, Rijpkema II, Rijpkema III, Rijpkema IV, Goossens I, Goossens II, Wielage, Pestana, Dielissen, Rădulescu II, and Vermeulen are not cumulative of the disclosure in the prior art disclosed by the applicants, cited by the Examiner, or considered during prosecution of the '083 Application.

For example, Rădulescu I was published no later than February 20, 2004 and thus constitutes prior art at least under 35 U.S.C. § 102(b) because it was published more than one year before the date of the filing of the '083 Application in the United States. Rădulescu I discloses many of the claimed features that the applicants and their prosecuting attorneys and/or agents argued were missing from the cited prior art during prosecution of the '083 Application. *See, e.g.*, Ex. G (Rădulescu I) at 1 (disclosing "a network interface for an on-chip network" that "is intended for systems on chip (SoC)"—the Æthereal NoC in particular—to facilitate communication between "master" and "slave" "IP modules"); *id.* at 1, 3–4, (describing the Æthereal NoC as containing network interface units with queues for buffering data and a "scheduler" that "optimize[s] the NoC utilization" in terms of amount of data and determines the moment of transmission based on communication properties. In fact, Rădulescu I (at least in view of the prior

[REDACTED]

art) renders unpatentable the claims of the '083 Application. See Petition for *Inter Partes* Review, *Samsung Elecs. Co., Ltd. v. Network Sys. Techs., LLC*, IPR2024-00339, at 9–74 (P.T.A.B. Dec. 18, 2023), which is incorporated by reference herein.

As another example, Rădulescu WO was filed October 7, 2003 and published April 22, 2004 and thus constitutes prior art at least under 35 U.S.C. § 102(b) because it was published more than one year before the filing of the '083 Application in the United States. Rădulescu WO also discloses improving an NoC by adapting the data transmission moment to communication properties. *See, e.g.*, Ex. I (Rădulescu WO) at Abstract, Figs. 1–3 (disclosing an integrated circuit comprising processing modules, a router network interconnect, and network interface units between them); *id.* at Abstract, 9:5–7, 9:18–23, 11:5–22, 16:32–34 (improving use of NoC resources by adapting the moment for transmitting data based on properties of the communications themselves, including the exact same ten communication properties claimed by the applicants in the finally amended claims of the '083 Application). In fact, Rădulescu WO (at least in view of the prior art) also renders unpatentable the claims of the '083 Application. See Petition for *Inter Partes* Review, *Samsung Elecs. Co., Ltd. v. Network Sys. Techs., LLC*, IPR2024-00339, at 60–74 (P.T.A.B. Dec. 18, 2023), which is incorporated by reference herein.

As further examples, Rijpkema I was published no later than September 2003, Rijpkema II was published no later than September 12, 2003, and Rijpkema III was published no later than March 2003, and thus all three references constitute prior art at least under 35 U.S.C. § 102(b), because they were published more than one year before the filing of the '083 Application in the United States. Rijpkema I, Rijpkema II, and Rijpkema III all disclose NoCs in which data flow through the network is optimized based on communication properties and the size of the data “flits.” *See, e.g.*, Ex. C (Rijpkema I) at 297, 299, 302 (disclosing NoC in which “a data path and

[REDACTED]

a control path” are “designed and optimised independently,” such as by appropriately sizing “so-called flits (flow control digits)” to their optimal size  $F_{opt}$ ); Ex. D (Rijpkema II) at 7, 13, 18 (similar); Ex. E (Rijpkema III) at 3, 5, 6 (similar). Each of Rijpkema I, Rijpkema II, and Rijpkema III, in view of the prior art, also renders unpatentable the claims of the '083 Application.

Additional examples of withheld prior art references material to patentability include Goossens I, Goossens II, Rijpkema IV, Wielage, Pestana, Dielissen, Rădulescu II, and Vermeulen. Each of these references constitutes prior art at least under 35 U.S.C. § 102(b) because they were all published more than one year before the date of the filing of the '083 Application in the United States. Each of those prior art references discloses, among other things, use of claimed communication properties in the operation of a NoC, such as to optimize its performance. *See, e.g.*, Ex. A (Goossens I) at 1, 3 (disclosing a NoC wherein communication is based on connections that account for communication properties such as guaranteed throughput, latency guarantees, flow control, and “lossless in-order (multicast) transmission”); Ex. B (Goossens II) at Abstract, 61, 71–73 (similar); Ex. H (Rădulescu II) at 275; *see also id.* at 278–80, 286–96 (similar); Ex. K (Vermeulen) at 2–3 (similar); Ex. F (Rijpkema IV) at 1–3 (disclosing a NoC “that meets the set of all simultaneous communication requirements of the individual IP blocks in a certain application”); Ex. J (Wielage) at Abstract, 2 (disclosing a NoC in which communication links “can be optimized for bandwidth, latency, power, or a combination of these, depending on performance requirements”); Ex. M (Pestana) at 2, 4 (disclosing a NoC in which connections are established based on communication properties and for which “metrics should be optimize[d]” based on the use application); Ex. L (Dielissen) at 2–4, Fig. 6 (disclosing a NoC in which “the flit size is chosen ... to optimize the data clock frequency and control frequency” and data is packetized in a buffer

[REDACTED]

and scheduled for routing using “a scheduler” that “arbitrate[s]” between “multiple channels” for data communication, including based on whether a slot is reserved for guaranteed throughput).

Rădulescu I, Rădulescu II, Rădulescu WO, Rijpkema I, Rijpkema II, Rijpkema III, Rijpkema IV, Goossens I, Goossens II, Wielage, Pestana, Dielissen, and Vermeulen are material to the patentability of the claims of the '083 Application as the teachings in those withheld references are relevant to, among other things, the claimed communication properties and optimal moment limitations, which the applicants and/or Examiner alleged were missing from the prior art and purportedly made the claims of the '083 Application patentable. The disclosures in Rădulescu I, Rădulescu II, Rădulescu WO, Rijpkema I, Rijpkema II, Rijpkema III, Rijpkema IV, Goossens I, Goossens II, Wielage, Pestana, Dielissen, and Vermeulen are not cumulative of the disclosure in any of the prior art references disclosed by the applicants, cited by the Examiner, or considered during prosecution of the '083 Application.<sup>2</sup> For example, Rădulescu I, Rădulescu II, Rădulescu WO, Rijpkema I, Rijpkema II, Rijpkema III, Rijpkema IV, Goossens I, Goossens II, Wielage, Pestana, Dielissen, and Vermeulen, when combined with secondary prior art references such as but not limited to U.S. Patent Application Publication No. 2002/0186660 to Bahadiroglu et al. or U.S. Patent No. 5,434,872 to Petersen et al., would have blocked issuance of the patent claims of the '083 Application.

---

<sup>2</sup> Although the applicants disclosed U.S. Patent No. 7,769,893 (the “'893 patent”) and the WO 2004/034173 (the “'173 WO Reference”) months after the claims of the '083 Application had already been allowed, the '893 patent is not prior art under § 102(b)—and thus would not have been considered by the Examiner as an invalidating (including at least obviating) reference on that ground—and the '173 WO Reference fails to disclose the claim elements that were the subject of the applicants’ arguments’ and the Examiner’s allowance, including the “determination unit,” “optimal amount of data to be buffered,” “optimal moment for sending the data,” and the ten communication properties recited in the allowed claims of the '083 Application.

[REDACTED]

Kees Gerard Willem Goossens had knowledge of Rădulescu I, Rădulescu II, Rădulescu WO, Rijpkema I, Rijpkema II, Rijpkema III, Rijpkema IV, Goossens I, Goossens II, Wielage, Pestana, Dielissen, and Vermeulen before filing and during prosecution of the '083 Application at least because he co-authored those publications and would have known they were published before the filing date of his own '083 Application. Andrei Rădulescu also had knowledge of all those publications before filing and during prosecution of the '083 Application at least because (1) he co-authored Rădulescu I, Rădulescu II, Rădulescu WO, Rijpkema I, Rijpkema II, Rijpkema III, Goossens II, Pestana, and Dielissen, (2) his Goossens II publication cites to Rijpkema IV, Goossens I, and Wielage, and (3) he co-authored another 2005 publication that cites Vermeulen, and thus he would have known they were published before the filing date of his own '083 Application. On information and belief, the prosecuting attorneys and/or agents for the '083 Application, including at least those identified above, also would have had knowledge of Rădulescu I, Rădulescu II, Rădulescu WO, Rijpkema I, Rijpkema II, Rijpkema III, Rijpkema IV, Goossens I, Goossens II, Wielage, Pestana, Dielissen, and Vermeulen before filing and during prosecution of the '083 Application at least through the applicants and/or through their own independent research on known prior art relating to the subject matter of the '083 Application. Further, the applicants', prosecuting attorneys', and/or agents' knowledge of Rădulescu WO is also demonstrated by the fact that both Rădulescu WO and the '173 WO Reference claim priority to EP 02079196.8, but only the disparate '173 WO Reference was provided to the USPTO. Despite this knowledge, Rădulescu I, Rădulescu II, Rădulescu WO, Rijpkema I, Rijpkema II, Rijpkema III, Rijpkema IV, Goossens I, Goossens II, Wielage, Pestana, Dielissen, and Vermeulen were withheld from all USPTO filings associated with the '083 Application.

[REDACTED]

On information and belief, the withholding by the applicants, prosecuting attorneys, and/or their agents of information material to patentability, including at least Rădulescu I, Rădulescu II, Rădulescu WO, Rijpkema I, Rijpkema II, Rijpkema III, Rijpkema IV, Goossens I, Goossens II, Wielage, Pestana, Dielissen, and Vermeulen during prosecution of the '083 Application was done with specific intent to mislead and/or deceive the USPTO. On information and belief, the applicants and prosecuting attorneys and/or agents knew that the withheld prior art, including at least Rădulescu I, Rădulescu II, Rădulescu WO, Rijpkema I, Rijpkema II, Rijpkema III, Rijpkema IV, Goossens I, Goossens II, Wielage, Pestana, Dielissen, and Vermeulen, were material to patentability of the claims of the '083 Application. All of these withheld references are directed to the same subject matter as the claimed invention and provide teachings relating to at least the communication properties and/or optimal moment limitations recited in every claim of the '083 Application, which at least the applicants would have known as co-authors of these publications. On information and belief, the applicants, prosecuting attorneys, and/or their agents made the deliberate decision to withhold the applicants' prior art publications, including at least Rădulescu I, Rădulescu II, Rădulescu WO, Rijpkema I, Rijpkema II, Rijpkema III, Rijpkema IV, Goossens I, Goossens II, Wielage, Pestana, Dielissen, and Vermeulen, during prosecution of the '083 Application because it would have been adverse to issuance of the desired patent claims in that application.

On information and belief, but for the applicants, prosecuting attorneys, and/or their agents intentionally and improperly withholding from the USPTO during prosecution of the '083 Application prior art references material to patentability (including at least Rădulescu I, Rădulescu II, Rădulescu WO, Rijpkema I, Rijpkema II, Rijpkema III, Rijpkema IV, Goossens I, Goossens II,

[REDACTED]

Wielage, Pestana, Dielissen, and Vermeulen), the '800 patent would not have issued. As a result, the '800 patent is unenforceable due to inequitable conduct.

**'2,893 Patent**

According to the face of U.S. Patent No. 8,072,893, it issued on December 6, 2011 from U.S. Application No. 11/912,175 (“the '175 Application”), which was filed by John Dielissen and Edwin Rijpkema on October 22, 2007. The '175 Application purports to be a national stage application, pursuant to 35 U.S.C. § 371, of International Application No. PCT/IB2006/051218, which had an international filing date of April 20, 2006. The '175 Application claimed priority to European Patent Application No. 05103232, which was filed on April 21, 2005. The application data sheet filed with the '175 Application was signed by Yan Glickberg and dated September 19, 2007, and the preliminary amendment filed with that application was also signed by Yan Glickberg. Submitted with the '175 Application was an executed power of attorney form signed by John Dielissen and Edwin Rijpkema, which was dated December 21, 2006 and appointed Jack E. Haken, Michael E. Marion, and Edward M. Blocker as the attorneys and/or agents to prosecute the '175 Application and transact all business in the USPTO connected with that patent application. John Dielissen and Edwin Rijpkema further declared in the signed December 21, 2006 document that they “acknowledge the duty to disclose information which is material to patentability as defined in 37 CFR 1.56.”

During the prosecution of the '175 Application, the Examiner issued several rejections over various prior art references. The applicants through their prosecuting attorneys and/or agents—including Dicran Halajian—responded to the rejections by amending the claims, adding a new claim, and arguing that the cited prior art failed to disclose elements of the claims, including a network comprising M\*N data storage elements for “introducing the particular delay of M\*N

cycles on the first communication channel, when the data communication network identifies the first communication channel as having a data transfer delay exceeding a predefined delay threshold,” as well as “inserting M\*N data storage elements” “in response to identifying that the first communication channel has a data transfer delay exceeding a predefined delay threshold” (the “M\*N data storage elements limitations”). In a Notice of Allowance issued on August 8, 2011, the Examiner stated the claims were, in the Examiner’s view, allowable because, among other things, “[t]he prior arts of record do not disclose” the claimed “data communication network” comprising those limitations.

On information and belief, despite having and being aware of a duty to disclose prior art material to patentability, the applicants, prosecuting attorneys, and/or their agents, failed to disclose to the USPTO during prosecution of the ’175 Application prior art bearing on the exact same subject matter as the patent application and material to patentability. This includes certain references authored by the applicants themselves and colleagues of the applicants at Philips. These withheld publications are particularly relevant to, among other things, the claimed “M\*N data storage elements” and “introducing a delay of M\*N cycles on the first communication channel,” which are recited in every issued claim of the ’175 Application. Although the applicants submitted certain of their own prior art publications as well as certain prior art publications co-authored by the applicants’ colleagues at Philips, several other applicant-authored and colleague-authored prior art publications (which are not cumulative of the disclosed references) were withheld.

For example, the article “*Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chip*” by Davide Bertozzi and Luca Benini was published in *IEEE Circuits and Systems Magazine* no later than October 2004 (hereinafter “Bertozzi”), and thus constitutes prior art at least under 35 U.S.C. §§ 102(a) and (b) because it was published before the alleged invention of the

[REDACTED]

'175 Application and was published more than one year before the date of the filing of the '175 Application in the United States. Bertozzi is attached as Exhibit N. Bertozzi discloses “an embodiment of the concept of NoC represented by the *Xpipes* NoC architecture,” which solves “the interconnect-delay problem” by “use of pipelined links” whose length of data storage elements  $N$  “can be tailored to the desired clock frequency that needs to be achieved in the design” and which can accommodate  $M$  flits of a data package in a clock cycle. Ex. N (Bertozzi) at 20, 24–25, Fig. 4.

Additional examples of withheld prior art references material to patentability include Goossens I, Goossens II, Rădulescu WO, and Vermeulen. Each of these references constitutes prior art at least under 35 U.S.C. § 102(b) because they were all published more than one year before the date of the filing of the '175 Application in the United States. Moreover, each discloses a NoC that permits communication of data packets over pipelined connections, among other things. *See, e.g.*, Ex. K (Vermeulen) at 3–4 (disclosing a NoC that uses “pipelined circuits”); Ex. B (Goossens II) at 72, 76 (similar); Ex. I (Rădulescu WO) at 19:22–29 (similar); Ex. A (Goossens I) at 2 (“When scheduling statically, a packet need not incur more than a pipeline delay per hop.”).

Bertozzi, Goossens I, Goossens II, Rădulescu WO, and Vermeulen are material to the patentability of the claims of the '175 Application as the teachings in those withheld references are relevant to, among other things, the  $M \times N$  data storage element limitations, which the applicants alleged was missing from the prior art and purportedly made the claims of the '175 Application patentable. The disclosures in Bertozzi, Goossens I, Goossens II, Rădulescu WO, and Vermeulen are not cumulative of the disclosure in the prior art disclosed by the applicants, cited by the Examiner, or considered during prosecution of the '175 Application. For example, Bertozzi discloses every limitation of, anticipates, and thus renders unpatentable the claims of the '175

[REDACTED]

Application. See Petition for *Inter Partes* Review, *Qualcomm Inc. v. Network Sys. Techs., LLC*, IPR2024-00355, at 10–47 (P.T.A.B. Dec. 20, 2023), which is incorporated by reference herein. Further, as another example, Bertozzi, Goossens I, Goossens II, Rădulescu WO, and Vermeulen, when combined with secondary prior art references such as but not limited to the “Networks on Chip” book edited by Axel Jantsch et al. and published no later than May 12, 2003 (“Jantsch”), “Interconnection Networks: An Engineering Approach” authored by José Duato et al. and published by *IEEE Computer Society Press* in 1997 (“Duato”), or “Optimal Buffered Routing Path Constructions for Single and Multiple Clock Domain Systems” authored by Soha Hassoun et al. and published by *IEEE* in 2002 (“Hassoun”), would have also blocked issuance of the patent claims of the ’175 Application. See *id.* at 37–81, which is incorporated by reference herein.

On information and belief, John Dielissen and Edwin Rijpkema had knowledge of Bertozzi at least during prosecution of the ’175 Application, if not sooner. First, they knew of the Xpipes technology disclosed in Bertozzi. For example, the applicants, prosecuting attorneys, and agents were aware of Xpipes and had disclosed other Xpipes prior art references during prosecution of the ’175 Application, such as the “XpipesCompiler: A Tool for Instantiating Application Specific Networks on Chip” reference by Antoine Jalabert et al. published no later than 2004. Second, the applicants’ colleagues at Philips working on the same NoC technology published numerous papers citing Bertozzi. These include at least “Congestion-Controlled Best-Effort Communication for Networks-on-Chip” by J.W. van den Brand et al. published by IEEE no later than April 2007, “Transaction-Based Communication-Centric Debug” by K. Goossens et al. published by IEEE no later than May 2007, and “On-Chip Network Interfaces Supporting Automatic Burst Write Creation, Posted Writes and Read Prefetch” by R. Stefan published by IEEE no later than July 2010. Third, Bertozzi itself indicates that one of the co-authors worked at the same place of

employment as the applicants. *See* Ex. N (Bertozzi) at 31 (“[Bertozzi] has been a visiting research at ... Philips Research Labs.”).

On information and belief, John Dielissen and Edwin Rijpkema also had knowledge of Goossens I, Goossens II, Rădulescu WO, and Vermeulen at least during prosecution of the '175 Application, if not sooner. They co-authored Goossens II (which cites Goossens I) and would have known they were published before the filing date of their own '175 Application. John Dielissen co-authored Vermeulen. And both Rădulescu WO and Vermeulen were authored by colleagues of the applicants at the same place of employment—Philips.

On information and belief, the prosecuting attorneys and/or agents for the '175 Application, including at least those identified above, also would have had knowledge of Bertozzi, Goossens I, Goossens II, Rădulescu WO, and Vermeulen during prosecution of the '175 Application at least through the applicants and/or through their own independent research on known prior art relating to the subject matter of the '175 Application.

Despite this knowledge, Bertozzi, Goossens I, Goossens II, Rădulescu WO, and Vermeulen were withheld from all USPTO filings associated with the '175 Application.

On information and belief, the withholding by the applicants, prosecuting attorneys, and/or their agents of information material to patentability, including at least Bertozzi, Goossens I, Goossens II, Rădulescu WO, and Vermeulen, during prosecution of the '175 Application was done with specific intent to mislead and/or deceive the USPTO. The Bertozzi paper, cited by the applicants' colleagues working in the same research group at Philips and regarding a subject matter of which the applicants were well aware, is directed to the same subject matter as the claimed invention and discusses—among all the other claim limitations—the M\*N data storage element limitations. Goossens I, Goossens II, Rădulescu WO, and Vermeulen were also written by one or

[REDACTED]

more of the applicants or their colleagues working in the same research group at Philips, and are directed to the same subject matter as the claimed invention. The withholding of these publications left the USPTO without any prior art references before the Examiner during prosecution of the '175 Application that disclosed the M\*N data storage elements limitations in the context of the claimed on-chip network. On information and belief, the applicants, prosecuting attorneys, and/or their agents knew that the withheld prior art, including at least Bertozzi, Goossens I, Goossens II, Rădulescu WO, and Vermeulen, was material to the patentability of the claims of the '175 Application and made this choice to present the misleading and deceptive argument that the claimed invention—including but not limited to the M\*N data storage elements limitations—was supposedly missing from the prior art and purportedly made the claims of the '175 Application patentable. On information and belief, the applicants, prosecuting attorneys, and/or their agents made the deliberate decision to withhold the applicants' prior art publications, including at least Bertozzi, Goossens I, Goossens II, Rădulescu WO, and Vermeulen, during prosecution of the '175 Application because it would have been adverse to issuance of the desired patent claims in that application.

On information and belief, but for the applicants, prosecuting attorneys, and/or their agents intentionally and improperly withholding from the USPTO during prosecution of the '175 Application prior art references material to patentability (including at least Bertozzi, Goossens I, Goossens II, Rădulescu WO, and Vermeulen), the '2,893 patent would not have issued. As a result, the '2,893 patent is unenforceable due to inequitable conduct.

**RESERVATION OF RIGHTS TO ASSERT  
ADDITIONAL DEFENSES OR COUNTERCLAIMS**

Samsung reserves the right to amend or supplement its First Amended Answer and Counterclaims. Samsung has not knowingly or intentionally waived any applicable defenses or

[REDACTED]

counterclaims, and it reserves the right to assert and rely upon other applicable defenses and counterclaims that may become available or apparent throughout the course of this action. Samsung continues to investigate this matter and reserves the right to amend or seek to amend its First Amended Answer and Counterclaims to assert any additional defenses and counterclaims that come to light upon further investigation and discovery.

### **COUNTERCLAIMS**

Pursuant to Federal Rule of Civil Procedure 13, Samsung Electronics Co., Ltd. (“SEC”) and Samsung Electronics America, Inc. (“SEA”) (collectively “Counterclaim-Plaintiffs”) counterclaim against Network System Technologies, LLC (“NST”) and, in support of thereof, allege the following:

### **PARTIES**

1. Counterclaim-Plaintiff SEC is a corporation organized and existing under the laws of the Republic of Korea with a principal place of business at 129 Samsung-ro, Maetan-3dong, Yeongtong-gu, Suwon-si, Gyeonggi-do, 16677, Republic of Korea.

2. Counterclaim-Plaintiff SEA is incorporated under the laws of New York with a principal place of business at 85 Challenger Road, Ridgefield Park, New Jersey, 07660.

3. Counterclaim-Defendant NST has represented that it is a limited liability company organized and existing under the laws of Delaware with a principal place of business at 533 Congress Street, Portland, ME 04101.

### **JURISDICTION AND VENUE**

4. This Court has jurisdiction over Counterclaim-Plaintiffs’ Declaratory Judgment claims pursuant to 28 U.S.C. §§ 2201–2202 and subject matter jurisdiction over patent infringement and validity pursuant to 28 U.S.C. §§ 1331 and 1338(a). An actual, substantial, and continuing justiciable controversy exists between Counterclaim-Plaintiffs and NST based on NST

[REDACTED]

having filed a Complaint against Counterclaim-Plaintiffs alleging infringement of the U.S. Patent Nos. 7,366,818 (“’818 patent”); 7,373,449 (“’449 patent”); 7,594,052 (“’052 patent”); 7,769,893 (“’9893 patent”); 8,072,893 (“’2893 patent”); and 8,086,800 (“’800 patent”) (collectively, the “Asserted Patents”), with respect to which Counterclaim-Plaintiffs require a declaration of their rights by this Court. Specifically, the controversy concerns the invalidity and non-infringement of the Asserted Patents and the right of NST to maintain suit for alleged infringement of the Asserted Patents.

5. The Court has personal jurisdiction over NST, *inter alia*, at least because NST has submitted to the personal jurisdiction of this Court by filing the Complaint.

6. Venue is proper in this District as to these Counterclaims pursuant to 28 U.S.C. §§ 1391(b)-(c) and 1400(b) *inter alia*, at least because NST has submitted to the venue of this Court by filing its Complaint here.

**FIRST COUNTERCLAIM**  
**(DECLARATORY JUDGMENT OF NON-INFRINGEMENT OF THE ’818 PATENT)**

7. Counterclaim-Plaintiffs incorporate paragraphs 1–6 of their Counterclaims as if fully set forth herein.

8. In its Complaint, NST avers that Counterclaim-Plaintiffs infringe the ’818 patent.

9. Counterclaim-Plaintiffs have not directly or indirectly infringed, contributed to or induced infringement of any valid or enforceable claim of the ’818 patent and have not otherwise committed any acts in violation of 35 U.S.C. § 271.

10. Accordingly, an actual controversy exists between Counterclaim-Plaintiffs and NST based on NST having filed its Complaint against Counterclaim-Plaintiffs alleging infringement of the ’818 patent.

11. Counterclaim-Plaintiffs therefore seek a declaration that they have not infringed,

[REDACTED]

and do not infringe, literally or under the doctrine of equivalents, directly or indirectly, any valid and enforceable claim of the '818 patent.

12. This is an exceptional case entitling Counterclaim-Plaintiffs to an award of their attorneys' fees incurred in connection with this action pursuant to 35 U.S.C. § 285.

**SECOND COUNTERCLAIM**  
**(DECLARATORY JUDGMENT OF PATENT INVALIDITY OF THE '818 PATENT)**

13. Counterclaim-Plaintiffs incorporate paragraphs 1–12 of their counterclaims as if fully set forth herein.

14. One or more of the claims of the '818 patent are invalid for failing to meet the conditions for patentability as set forth in 35 U.S.C. § 1 *et seq.*, including, but not limited to §§ 101, 102, 103, 112, and 116 and the rules, regulations, and laws pertaining thereto.

15. An actual controversy exists between Counterclaim-Plaintiffs and NST based on NST having filed its Complaint against Counterclaim-Plaintiffs alleging infringement of the '818 patent.

16. Counterclaim-Plaintiffs therefore seek a declaration that one or more of the claims of the '818 patent are invalid for failing to meet the conditions for patentability as set forth in 35 U.S.C. § 1 *et seq.*

17. This is an exceptional case entitling Counterclaim-Plaintiffs to an award of their attorneys' fees incurred in connection with this action pursuant to 35 U.S.C. § 285.

**THIRD COUNTERCLAIM**  
**(DECLARATORY JUDGMENT OF NON-INFRINGEMENT OF THE '449 PATENT)**

18. Counterclaim-Plaintiffs incorporate paragraphs 1–17 of their Counterclaims as if fully set forth herein.

19. In its Complaint, NST avers that Counterclaim-Plaintiffs infringe the '449 patent.

20. Counterclaim-Plaintiffs have not directly or indirectly infringed, contributed to or

[REDACTED]

induced infringement of any valid or enforceable claim of the '449 patent and have not otherwise committed any acts in violation of 35 U.S.C. § 271.

21. Accordingly, an actual controversy exists between Counterclaim-Plaintiffs and NST based on NST having filed its Complaint against Counterclaim-Plaintiffs alleging infringement of the '449 patent.

22. Counterclaim-Plaintiffs therefore seek a declaration that they have not infringed, and do not infringe, literally or under the doctrine of equivalents, directly or indirectly, any valid and enforceable claim of the '449 patent.

23. This is an exceptional case entitling Counterclaim-Plaintiffs to an award of their attorneys' fees incurred in connection with this action pursuant to 35 U.S.C. § 285.

**FOURTH COUNTERCLAIM**  
**(DECLARATORY JUDGMENT OF PATENT INVALIDITY OF THE '449 PATENT)**

24. Counterclaim-Plaintiffs incorporate paragraphs 1–23 of their counterclaims as if fully set forth herein.

25. One or more of the claims of the '449 patent are invalid for failing to meet the conditions for patentability as set forth in 35 U.S.C. § 1 *et seq.*, including, but not limited to §§ 101, 102, 103, 112, and 116 and the rules, regulations, and laws pertaining thereto.

26. An actual controversy exists between Counterclaim-Plaintiffs and NST based on NST having filed its Complaint against Counterclaim-Plaintiffs alleging infringement of the '449 patent.

27. Counterclaim-Plaintiffs therefore seek a declaration that one or more of the claims of the '449 patent are invalid for failing to meet the conditions for patentability as set forth in 35 U.S.C. § 1 *et seq.*

28. This is an exceptional case entitling Counterclaim-Plaintiffs to an award of their

attorneys' fees incurred in connection with this action pursuant to 35 U.S.C. § 285.

**FIFTH COUNTERCLAIM**  
**(DECLARATORY JUDGMENT OF NON-INFRINGEMENT OF THE '052 PATENT)**

29. Counterclaim-Plaintiffs incorporate paragraphs 1–28 of their Counterclaims as if fully set forth herein.

30. In its Complaint, NST avers that Counterclaim-Plaintiffs infringe the '052 patent.

31. Counterclaim-Plaintiffs have not directly or indirectly infringed, contributed to or induced infringement of any valid or enforceable claim of the '052 patent and have not otherwise committed any acts in violation of 35 U.S.C. § 271.

32. Accordingly, an actual controversy exists between Counterclaim-Plaintiffs and NST based on NST having filed its Complaint against Counterclaim-Plaintiffs alleging infringement of the '052 patent.

33. Counterclaim-Plaintiffs therefore seek a declaration that they have not infringed, and do not infringe, literally or under the doctrine of equivalents, directly or indirectly, any valid and enforceable claim of the '052 patent.

34. This is an exceptional case entitling Counterclaim-Plaintiffs to an award of their attorneys' fees incurred in connection with this action pursuant to 35 U.S.C. § 285.

**SIXTH COUNTERCLAIM**  
**(DECLARATORY JUDGMENT OF PATENT INVALIDITY OF THE '052 PATENT)**

35. Counterclaim-Plaintiffs incorporate paragraphs 1–34 of their counterclaims as if fully set forth herein.

36. One or more of the claims of the '052 patent are invalid for failing to meet the conditions for patentability as set forth in 35 U.S.C. § 1 *et seq.*, including, but not limited to §§ 101, 102, 103, 112, and 116 and the rules, regulations, and laws pertaining thereto.

37. An actual controversy exists between Counterclaim-Plaintiffs and NST based on

[REDACTED]

NST having filed its Complaint against Counterclaim-Plaintiffs alleging infringement of the '052 patent.

38. Counterclaim-Plaintiffs therefore seek a declaration that one or more of the claims of the '052 patent are invalid for failing to meet the conditions for patentability as set forth in 35 U.S.C. § 1 *et seq.*

39. This is an exceptional case entitling Counterclaim-Plaintiffs to an award of their attorneys' fees incurred in connection with this action pursuant to 35 U.S.C. § 285.

**SEVENTH COUNTERCLAIM**  
**(DECLARATORY JUDGMENT OF NON-INFRINGEMENT OF THE '9893 PATENT)**

40. Counterclaim-Plaintiffs incorporate paragraphs 1–39 of their Counterclaims as if fully set forth herein.

41. In its Complaint, NST avers that Counterclaim-Plaintiffs infringe the '9893 patent.

42. Counterclaim-Plaintiffs have not directly or indirectly infringed, contributed to or induced infringement of any valid or enforceable claim of the '9893 patent and have not otherwise committed any acts in violation of 35 U.S.C. § 271.

43. Accordingly, an actual controversy exists between Counterclaim-Plaintiffs and NST based on NST having filed its Complaint against Counterclaim-Plaintiffs alleging infringement of the '9893 patent.

44. Counterclaim-Plaintiffs therefore seek a declaration that they have not infringed, and do not infringe, literally or under the doctrine of equivalents, directly or indirectly, any valid and enforceable claim of the '9893 patent.

45. This is an exceptional case entitling Counterclaim-Plaintiffs to an award of their attorneys' fees incurred in connection with this action pursuant to 35 U.S.C. § 285.

**EIGHTH COUNTERCLAIM**  
**(DECLARATORY JUDGMENT OF PATENT INVALIDITY OF THE '9893 PATENT)**

46. Counterclaim-Plaintiffs incorporate paragraphs 1–45 of their counterclaims as if fully set forth herein.

47. One or more of the claims of the '9893 patent are invalid for failing to meet the conditions for patentability as set forth in 35 U.S.C. § 1 *et seq.*, including, but not limited to §§ 101, 102, 103, 112, and 116 and the rules, regulations, and laws pertaining thereto.

48. An actual controversy exists between Counterclaim-Plaintiffs and NST based on NST having filed its Complaint against Counterclaim-Plaintiffs alleging infringement of the '9893 patent.

49. Counterclaim-Plaintiffs therefore seek a declaration that one or more of the claims of the '9893 patent are invalid for failing to meet the conditions for patentability as set forth in 35 U.S.C. § 1 *et seq.*

50. This is an exceptional case entitling Counterclaim-Plaintiffs to an award of their attorneys' fees incurred in connection with this action pursuant to 35 U.S.C. § 285.

**NINTH COUNTERCLAIM**  
**(DECLARATORY JUDGMENT OF NON-INFRINGEMENT OF THE '2893 PATENT)**

51. Counterclaim-Plaintiffs incorporate paragraphs 1–50 of their Counterclaims as if fully set forth herein.

52. In its Complaint, NST avers that Counterclaim-Plaintiffs infringe the '2893 patent.

53. Counterclaim-Plaintiffs have not directly or indirectly infringed, contributed to or induced infringement of any valid or enforceable claim of the '2893 patent and have not otherwise committed any acts in violation of 35 U.S.C. § 271.

54. Accordingly, an actual controversy exists between Counterclaim-Plaintiffs and

[REDACTED]

NST based on NST having filed its Complaint against Counterclaim-Plaintiffs alleging infringement of the '2893 patent.

55. Counterclaim-Plaintiffs therefore seek a declaration that they have not infringed, and do not infringe, literally or under the doctrine of equivalents, directly or indirectly, any valid and enforceable claim of the '2893 patent.

56. This is an exceptional case entitling Counterclaim-Plaintiffs to an award of their attorneys' fees incurred in connection with this action pursuant to 35 U.S.C. § 285.

**TENTH COUNTERCLAIM**  
**(DECLARATORY JUDGMENT OF PATENT INVALIDITY OF THE '2893 PATENT)**

57. Counterclaim-Plaintiffs incorporate paragraphs 1–56 of their counterclaims as if fully set forth herein.

58. One or more of the claims of the '2893 patent are invalid for failing to meet the conditions for patentability as set forth in 35 U.S.C. § 1 *et seq.*, including, but not limited to §§ 101, 102, 103, 112, and 116 and the rules, regulations, and laws pertaining thereto.

59. An actual controversy exists between Counterclaim-Plaintiffs and NST based on NST having filed its Complaint against Counterclaim-Plaintiffs alleging infringement of the '2893 patent.

60. Counterclaim-Plaintiffs therefore seek a declaration that one or more of the claims of the '2893 patent are invalid for failing to meet the conditions for patentability as set forth in 35 U.S.C. § 1 *et seq.*

61. This is an exceptional case entitling Counterclaim-Plaintiffs to an award of their attorneys' fees incurred in connection with this action pursuant to 35 U.S.C. § 285.

**ELEVENTH COUNTERCLAIM**  
**(DECLARATORY JUDGMENT OF NON-INFRINGEMENT OF THE '800 PATENT)**

62. Counterclaim-Plaintiffs incorporate paragraphs 1–61 of their Counterclaims as if

[REDACTED]

fully set forth herein.

63. In its Complaint, NST avers that Counterclaim-Plaintiffs infringe the '800 patent.

64. Counterclaim-Plaintiffs have not directly or indirectly infringed, contributed to or induced infringement of any valid or enforceable claim of the '800 patent and have not otherwise committed any acts in violation of 35 U.S.C. § 271.

65. Accordingly, an actual controversy exists between Counterclaim-Plaintiffs and NST based on NST having filed its Complaint against Counterclaim-Plaintiffs alleging infringement of the '800 patent.

66. Counterclaim-Plaintiffs therefore seek a declaration that they have not infringed, and do not infringe, literally or under the doctrine of equivalents, directly or indirectly, any valid and enforceable claim of the '800 patent.

67. This is an exceptional case entitling Counterclaim-Plaintiffs to an award of their attorneys' fees incurred in connection with this action pursuant to 35 U.S.C. § 285.

**TWELFTH COUNTERCLAIM**  
**(DECLARATORY JUDGMENT OF PATENT INVALIDITY OF THE '800 PATENT)**

68. Counterclaim-Plaintiffs incorporate paragraphs 1–67 of their counterclaims as if fully set forth herein.

69. One or more of the claims of the '800 patent are invalid for failing to meet the conditions for patentability as set forth in 35 U.S.C. § 1 *et seq.*, including, but not limited to §§ 101, 102, 103, 112, and 116 and the rules, regulations, and laws pertaining thereto.

70. An actual controversy exists between Counterclaim-Plaintiffs and NST based on NST having filed its Complaint against Counterclaim-Plaintiffs alleging infringement of the '800 patent.

71. Counterclaim-Plaintiffs therefore seek a declaration that one or more of the claims

[REDACTED]

of the '800 patent are invalid for failing to meet the conditions for patentability as set forth in 35 U.S.C. § 1 *et seq.*

72. This is an exceptional case entitling Counterclaim-Plaintiffs to an award of their attorneys' fees incurred in connection with this action pursuant to 35 U.S.C. § 285.

**RESERVATION OF RIGHTS TO ASSERT  
ADDITIONAL DEFENSES OR COUNTERCLAIMS**

Counterclaim-Plaintiffs reserve the right to supplement their Counterclaims. Counterclaim-Plaintiffs have not knowingly or intentionally waived any counterclaims, and reserve the right to assert and rely upon other counterclaims that may become available or apparent throughout the course of this action. Counterclaim-Plaintiffs continue to investigate this matter and reserve the right to amend or seek to amend their Counterclaims to assert any counterclaims that come to light upon further investigation and discovery.

**DEMAND FOR TRIAL BY JURY**

Pursuant to Rule 38(b) of the Federal Rules of Civil Procedure, Samsung hereby demands a trial by jury on all issues so triable.

**PRAYER FOR RELIEF**

WHEREFORE, Samsung respectfully prays for the following relief:

- A. Plaintiff takes nothing by way of its Complaint;
- B. The Court enter judgment in favor of Samsung and against Plaintiff, thereby dismissing the Complaint in its entirety, with prejudice, and deny Plaintiff all requested relief;
- C. The Court enter judgment that each of the '818, '449, '052, '800, and '2,893 patents are unenforceable; and
- D. Judgment awarding Samsung such other relief as this Court deems just and proper.

WHEREFORE, Counterclaim-Plaintiffs respectfully pray for the following relief:

[REDACTED]

E. Declaratory judgment that Counterclaim-Plaintiffs have not, literally or under the doctrine of equivalents, directly or indirectly, willfully or otherwise, infringed any claim of the Asserted Patents;

F. Declaratory judgment that the claims of the Asserted Patents are invalid;

G. The Court declare that this case is exceptional, entitling Counterclaim-Plaintiffs to their attorneys' fees under 35 U.S.C. § 285 and their costs; and

H. Award Counterclaim-Plaintiffs such other and further relief as the Court deems just and proper.

Dated: January 23, 2024

Respectfully submitted,

/s/ Melissa R. Smith

Melissa R. Smith  
**GILLAM & SMITH, LLP**  
303 South Washington Avenue  
Marshall, TX 75670  
Telephone: (903) 934-8450  
Facsimile: (903) 934-9257  
Email: melissa@gillamsmithlaw.com

Gregory S. Arovas, P.C. (*pro hac vice*)  
Alan M. Rabinowitz (*pro hac vice*)  
**KIRKLAND & ELLIS LLP**  
601 Lexington Avenue  
New York, NY 10022  
Telephone: (212) 446-4800  
Facsimile: (212) 446-4900  
Email: garovas@kirkland.com  
alan.rabinowitz@kirkland.com

F. Christopher Mizzo, P.C. (*pro hac vice*)  
Karthik Ravishankar (*pro hac vice*)  
Stephen DeSalvo (*pro hac vice*)  
**KIRKLAND & ELLIS LLP**  
1301 Pennsylvania Avenue, N.W.  
Washington, D.C. 20004  
Telephone: (202) 389-5000  
Facsimile: (202) 389-5200  
Email: chris.mizzo@kirkland.com  
karthik.ravishankar@kirkland.com  
stephen.desalvo@kirkland.com

*Counsel for Defendants Samsung Electronics Co.,  
Ltd., Samsung Semiconductor, Inc., Samsung  
Electronics America, Inc., and Samsung Austin  
Semiconductor, LLC*

[REDACTED]

**CERTIFICATE OF SERVICE**

The undersigned certifies that the foregoing document was served on all counsel of record who have consented to electronic service via the Court's CM ECF system per Local Rule CV-5(a)(3) on January 23, 2024.

s. Melissa R. Smith  
Melissa R. Smith

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

# Exhibit A

# Networks on Silicon: Combining Best-Effort and Guaranteed Services

K. Goossens, J. van Meerbergen, A. Peeters, and P. Wielage  
Philips Research Laboratories, Eindhoven, The Netherlands  
{kees.goossens,jef.van.meerbergen,ad.peeters,paul.wielage}@philips.com

## Abstract

*We advocate a network on silicon (NOS) as a hardware architecture to implement communication between IP cores in future technologies, and as a software model in the form of a protocol stack to structure the programming of NOSs. We claim guaranteed services are essential. In the ÆTHEREAL NOS they pervade the NOS as a requirement for hardware design, and as foundation for software programming.*

## 1. Introduction

It will soon be possible to manufacture systems on silicon (SOSs) containing 10M gates and megabytes of embedded software. The complexity of these systems will be overwhelming, and two fundamental questions arise. First, from a hardware view point, how can these SOSs be designed? What are the required hardware architecture concepts? Second, from a software (application) perspective, how can the SOS hardware be programmed? What software concepts are needed? Our tenet is that only a consistent approach to both questions will enable us to harness the opportunities offered by semiconductor technology.

We believe in the intellectual-property block (IP) re-use model. This means that a SOS contains hundreds of IPs that are bought or premade. Assuming their correctness, their communication is then the problem. We propose a *network on silicon* (NOS) to implement the communication between these IPs. A NOS is a hardware architecture together with a programming model.

*Guaranteed services* are the focus of this paper: they serve as the basis for the programming model, and as a requirement for the NOS hardware architecture. A *protocol stack* is a layered approach to offering different services to different clients, based on a common network. It structures communication complexity from the physical implementation up to the application in a number of layers. The OSI reference model [4], used in this paper, comprises seven layers. In Section 2 we motivate why a packet-switched router network is a suitable hardware architecture, using the lower three layers as a backdrop. The top four layers are independent of the hardware architecture and are discussed in Section 3. In that context we show why guaranteed services offer many advantages over *best-effort* services when programming SOSs.

## 2. Packet-switched Router Networks

In this section we consider how a SOS containing hundreds of IPs can be designed, especially regarding layout and timing

closure. Consider that the communication patterns of the IPs are dynamic at two levels. First, multiple configurations exist, each corresponding to a required user functionality. Second, communication data rates are variable within a configuration (e.g. variable-bit-rate MPEG streams). Structuring the wires used for communication between IPs is the main problem. If dedicated point-to-point wires are used, IPs are connected to all possible communication partners. This leads to many global wires, with three problems. First, they must be dimensioned for worst-case conditions in terms of a) electrical parameters such as cross talk and wire spacing, and b) volume of traffic. Second, it causes wire congestion around each IP. Thin closely-spaced high-power wires solve congestion, but this conflicts with the requirements for global wires, which are fat and widely spaced. Third, layout and timing dependencies between all IPs make timing closure global, making it non-scalable. Worse, it is layout specific: it cannot be re-used over design iterations or different designs [2].

A router network based on packet switching helps on these accounts. Fundamentally, wires are shared, or made programmable, by introducing switches (routers). This is possible because dedicated wires are normally underutilised (as little as 10% [3]). The first benefit is that there are fewer programmable wires, which reduces wire congestion around IP blocks. Second, traffic of multiple IPs is combined over a single wire, which can be dimensioned for the average traffic over all blocks using that wire instead of worst-case traffic per IP. Third, inter-IP wires are now structured: either they are short to get on the network, or they are router to router. The network topology and router layout can both be fixed (e.g. a torus and mesh [3] or a fat tree and Maltese cross [5]), or only the network topology. This allows dedicated optimisations that are re-usable over SOSs. Finally, and most importantly, decoupling of computation (IPs) and communication (NOS) enables compositional layout and timing closure. IPs communicate only using the network so that their layout and timing closure are independent of other IPs.

Programmable wires also have drawbacks. First, different IPs will contend for the use of a wire, requiring arbitration or scheduling. The properties of a router network depend much on the kind of scheduling it uses. A schedule can be fixed for a configuration (static) or dynamic (within a configuration). Static scheduling must be worst case, while dynamic scheduling leads to dimensioning for average traffic volume. Scheduling can also be independent for each router

(local) or consider all routers (global). Global scheduling can avoid *contention* (several packets want to use the same resource at the same time) and *congestion* (packets waiting for a resource) but is not scalable. Global scheduling is usually static (circuit switching) and local scheduling dynamic (packet switching). Performance guarantees (i.e. absolute certainty instead of probabilistic behaviour) are more easily given for global static scheduling than for local dynamic scheduling. The way packets are buffered and scheduled in routers, and the effects on performance guarantees has been the subject of intense research. Fundamentally, sharing and guarantees are conflicting, and efficiently combining guaranteed traffic with best-effort traffic is hard [7].

Second, programmable wires may be slower than using dedicated IP-to-IP wires, which may contain some buffers. In a NOS a packet must traverse a number of routers. When scheduling statically, a packet need not incur more than a pipeline delay per hop. But for dynamic scheduling the header of each packet must be inspected before it can be sent to the appropriate output port. Contention and congestion may further delay a packet. Finally, a packet header uses some of the communication bandwidth.

Although a NOS based on packet-switching has its costs (packetisation, routers, buffering, harder to give guarantees) it is essential for an IP re-use strategy because it enables compositional and scalable integration of the IPs. Moreover, a NOS infrastructure (such as routers and protocol stack) can be re-used and its design cost amortised over multiple designs.

The issues raised in this section can be mapped to the OSI protocol stack [4]. Below we discuss the three lower layers that are specific to the hardware architecture of the network.

The *physical layer* defines how a single bit is sent over a physical medium (such as a wire, bus, optical link, radio waves). For a NOS it deals with signal voltages and slopes, wire sizing, etc. of inter-router wires. The preceding discussions motivate why a router-based NOS eases the implementation of the physical on-chip wires.

The *data-link layer* ensures reliable (lossless, fault-free) communication over a single data link. It considers cross-talk and fault-tolerance (both transient soft errors and hard errors). For shared and especially broad-cast media (such as busses), the medium access sublayer deals with contention for the link. Examples are bus arbitration, and router-to-router flow control.

The *network layer* has two main functions for a given network. First, delivering packets, more specifically, routing, congestion control, and scheduling over multiple links. Second, network management, including accounting, monitoring, management or steering, and internetworking.

### 3. Guaranteed Services are essential

In this section we discuss the remainder of the OSI protocol stack, and then show why the services offered by this stack

must be predictable.

In a re-use model IPs that are bought or re-used cannot be adapted to each SOS they are used in. They will interact in many different ways (event-driven, data streaming, message passing, shared memory, etc.) [8]. Thus the interconnecting network has to be flexible, in terms of the services that it offers. This is the task of the top four OSI layers. They are independent of the network hardware architecture, *end to end* (the network is transparent), and *peer to peer* (between entities at the same level of abstraction, such as task to task). We now list these layers, starting with the lowest.

The aim of the *transport layer* is to provide reliable *end-to-end services over connections*, through packetisation, end-to-end connection management (perhaps over multiple networks), (de)multiplexing multiple transport connections over network connections, and so on. Examples of services per connection are: uncorrupted transmission, lossless transmission, in-order delivery, throughput, jitter (delay variation), and latency. Flow control avoids data loss by aligning IP production and consumption rates.

In the *session layer* multiple connections are combined into services such as multicast, half duplex and full duplex connections. Synchronisation of multiple connections (of audio and video streams, for example), and token management for mutual exclusion and atomic (chains of) transactions (e.g. test and set, swap) may also be offered.

The *presentation layer* converts the application view on data to a view more appropriate for transport and vice versa. Examples are encryption and endian-ness conversions.

In the *application layer* communication is natural to applications, running on IPs. This could be bit streams obeying the MP3 standard, video frames, or Java byte code.

We now return to the question of programming. A SOS consists of a set of IPs and a NOS that must be (dynamically) *configured*. For a given user function, computation is mapped to functional IPs (e.g. tasks to processors) and communication to the NOS. The number of configurations can be large and they may be generated at run-time by an application manager. To avoid unnecessary design iterations, we need to be sure of correctness of the combination of static hardware (the IPs and their interconnect) and dynamic software (the configurations) as early as possible in the design flow. Programming must be predictable; this can be ensured by basing it on a NOS that offers guaranteed services.

A NOS with guaranteed services simplifies programming individual IPs in three ways. First, some IPs have inherent performance requirements, such as a minimum throughput (for real-time streaming data), or bounded latency (for interrupts). A NOS with guaranteed services eases their integration. Second, an IP communicates using the NOS. Requests for services make its assumptions and dependencies on the NOS explicit, structuring the design and programming of the IP. Third, an IP can also be simpler when using guaranteed

services because it has fewer possible interactions (a stricter contract) with its environment. A service request may be granted or denied by the network. Communication failures are therefore restricted to the configuration phase of the IP, instead of every communication action, so simplifying the IP's programming model.

Guaranteed services offer the following advantages for the composition of IPs. First, services that are guaranteed to an IP are not affected by other IPs in the network, making reasoning about the IP in isolation possible. This is essential for a *compositional construction* (design and programming) of SOSs. When IPs, assumed to be correct, are combined in a system it must be clear that the system will function as a whole. This holds both at the time of design and at run time. Current systems often fail this requirement. Consider a bus; adding an IP influences the electrical parameters (like load) of the bus, as well as the existing traffic patterns. The latter can seriously affect cache behaviour and hence embedded processor performance. Second, a NOS that offers guaranteed services must accurately model its resources to offer those services. This makes costs of the NOS explicit early in the design of a SOS. At run-time an application or quality-of-service manager reconfigures the SOS. This can only be effective when the NOS is observable and controllable, but above all, predictable in terms of performance and cost. Third, a NOS based on guaranteed services concentrates communication failures to (re)configuration points. When the NOS declines a service request from an IP, only that IP is affected. Fall-back strategies are much easier for local configuration-time failure than for global failure at any point caused by any of the communicating parties.

#### 4. The Æthereal Network-on-Silicon

The ÆTHEREAL NOS intends to support both guaranteed and best-effort services. We can only list some salient features here, to indicate our current direction.

We use a combination of circuit and packet switching, in the spirit of ATM [1]. A channel implements a simplex point-to-point communication between two network interfaces (possibly with multiple paths). A channel can have a combined throughput and latency guarantee (GT), or be best effort (BE), with an unspecified finite latency upper bound. Delivery order is not guaranteed over different paths or channels. A connection combines multiple channels and can include flow control and reordering for lossless in-order (multicast) transmission. The programming model is concurrent and distributed: IPs can set up and remove connections in a pipelined fashion, and at the same time. Configurations that have been computed off-line can be efficiently loaded and extended or changed at run time. GT channels are created and removed with BE packets that reserve a path through the network, subject to resource availability.

BE packets use worm-hole source routing, and GT streams

use time-division-multiplexed circuit switching. Routers use virtual output queuing for BE traffic. GT traffic is tightly scheduled to minimise buffering. Routers do not drop or reorder packets, the former due to fine-grain link-level flow control. A variant of the iSLIP switch-scheduling algorithm [6] is used for BE traffic. BE traffic use all spare capacity in the network, i.e. bandwidth that is not reserved or unused by GT traffic. Network interfaces implement multi-path and multi-channel ordering, and channel flow control.

#### 5. Conclusions

In the IP re-use model communication between the many IP cores is key. We have shown that a network on silicon (NOS) eases the design of systems on silicon (SOSs) at the levels of hardware design and software programming.

A NOS hardware architecture based on a packet-switched router network allows average-case wire dimensioning, and reduces wire congestion around IPs. It breaks the fatal global timing closure loop by separating inter-IP from intra-IP communication, and can so reduce global design iterations. The OSI protocol stack structures the communication complexity in the router network (the physical to network layers) and the programming model (the transport to application layers).

While a protocol stack offers diverse services on a single hardware architecture, we show that predictability, in the form of guaranteed services, greatly benefits programming a SOS at two levels. Individual IPs with real-time requirements are more easily integrated, and the NOS's actions are restricted by the contract, simplifying the IP's IO. At the NOS level, programming of IPs is compositional because their services are independent of each other. Quality-of-service managers can effectively observe and steer a NOS because it behaves predictably in terms of performance and cost.

We believe that a NOS must be a synthesis of hardware architecture and software programming model with guaranteed services at its core. In the ÆTHEREAL NOS we aim to do so efficiently.

#### References

- [1] ATM Forum. *ATM User-Network Interface Specification*. 1994
- [2] R. E. Bryant, et al. Limitations and challenges of computer-aided design technology for CMOS VLSI. *Proc. of the IEEE*, 89(3):341–365, March 2001.
- [3] W. J. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *DAC*, pages 684–689, June 2001.
- [4] J. D. Day and H. Zimmerman. The OSI reference model. In *Proceedings of the IEEE*, volume 71, pages 1334–1340, 1983.
- [5] P. Guerrier. *Un Réseau D'Interconnexion pour Systèmes Intégrés*. PhD thesis, Université Paris VI, Mar. 2000.
- [6] N. McKeown. iSLIP: A scheduling algorithm for input-queued switches. *IEEE Transactions on Networking*, 7(2), Apr. 1999.
- [7] J. Rexford and K. G. Shin. Support for multiple classes of traffic in multicomputer routers. In *LNCS 853*, 1994.
- [8] M. Sgroi, et al. Addressing the system-on-a-chip interconnect woes through communication-based design. In *DAC'2001*

# Exhibit B

## Chapter 4

# GUARANTEEING THE QUALITY OF SERVICES IN NETWORKS ON CHIP

Kees Goossens, John Dielissen, Jef van Meerbergen,  
Peter Poplavko<sup>†</sup>, Andrei Rădulescu, Edwin Rijpkema,  
Erwin Waterlander, and Paul Wielage

*Philips Research Laboratories, Eindhoven, The Netherlands*

<sup>†</sup> *Technical University of Eindhoven, Eindhoven, The Netherlands*

Kees.Goossens@philips.com

**Abstract** Users expect a *predictable quality of service* (QoS) of embedded systems, even for future, more dynamic, applications. System-on-chip designers use networks on chip (NOC) to solve deep submicron problems, and to divide global problems into local, decoupled problems. NOCs provide services through protocol stacks, and introducing *guaranteed* services enables IP re-use and platform-based design. It also provides globally predictable behaviour, as required by the user, when combining local, decoupled solutions. There are several levels of QoS commitment (correctness, completion, completion bounds), with increasing cost. A combination of guaranteed and best-effort (no commitment) services combines their respective attractive features: predictable behaviour, and good average resource utilisation. The ÆTHEREAL NOC is an example of this approach, and forms the basis of a QoS-based design style, as advocated in this chapter.

## 1. Future applications and systems on chip

In this section we raise the following question: *why and how must systems implementing future applications guarantee the quality of service (QoS)?* We look at each of the components in turn, in the context of embedded systems and systems on a chip (SOC). We then observe that new SOC architectures and design methods divide global problems into local ones, and rely on networks on chip (NOC) to compose local solutions. We answer the question, in Section 2, by presenting a synthesis of a QoS-based design style and networks on chip. Section 3 explains that QoS can be decomposed into several levels of

commitment, with different resource requirements. In Section 4 we introduce the *ÆTHEREAL* NOC, and compare it to other NOCs, in Section 5.

### 1.1 Characteristics of future applications

We are witnessing the *convergence* of previously unrelated application domains: computation (personal digital assistants, mobile computing), communication (telephone, videophone, networking), and multimedia (audio, photography, video, augmented and virtual reality). Convergence leads to increased functionality and heterogeneity, as previously unrelated functions are combined. Systems become more dynamic, or even unpredictable, as new algorithms seek to take advantage of the disparity between average and worst-case processing. Compressed audio and video data of MPEG2 is an example. Algorithms are also shifting to higher semantic levels, and the semantic complexity of data is less predictable than its volume. An example is the shift from constant pixel processing (scaling, edge enhancement, and so on), to MPEG4 object detection and synthesis, to face recognition and scenario detection. Finally, increased interaction with the environment also makes systems more dynamic. System functionality can be influenced by the current location (affecting communication or computation capabilities), environmental conditions (precipitation and multipath interference can affect mobile communication), and other systems in the proximity (e.g. car guidance, ad hoc networking).

Trends such as ambient intelligence and the networked home make future applications *embedded* and *pervasive*. Moreover, applications acquire a responsibility for the control of physical objects, such as home heating systems, cars, and so on. Real-time performance and safety are critical in many of these applications.

Below, we examine how users interact with these applications, and then what is required to design these systems.

### 1.2 Quality of service: a user view

*Users* expect a certain behaviour of applications; in other words, they must be *predictable*. While those expectations may be low, as is often the case for personal computers, a certain fitness for purpose is always assumed. Consumer electronics are subject to higher demands: a television must have a robust user interface and is not allowed to crash or be unresponsive. Expectations are stricter yet for real-time applications (e.g. involving audio and video, or control systems); a television must display at least 50 pictures of a constant quality per second, for example. The essence of quality of service (QoS) is therefore the offering of a predictable system behaviour to the user.

A central question is how to reconcile the dynamic, unpredictable nature of future applications with the requirements for predictable services.

### 1.3 Implementing future applications on chip

Having identified the characteristics of future applications and their QoS requirements, we now turn to the question of their implementation in SOCs. Moore's law describes the *exponential growth* over time of the number of transistors that can be integrated in an IC. It predicts that chips in 2010 will count over 4 billion transistors, operating in the multi-GHz range [1]. It is this abundance of computational power that has fuelled the convergence of application domains described above. However, Moore's law is only a prediction, and two major obstacles must be solved to make it a reality [2]. First, to use future VLSI technologies, several *deep-submicron problems* must be solved: the increasing disparity between transistor and wire speeds, power delivery and dissipation, and signal integrity. Second, the intrinsic computational power of an IC must not only be used efficiently and effectively, but the time and effort to design a system containing both hardware and software must also remain acceptable. The so-called *design productivity gap* states that the increase in our ability to design SOCs does not match Moore's law. To close the gap, system design methods to implement applications with the latest VLSI technology (including the definition of architectures, mapping applications to architectures, and programming) must harness exponential hardware resource growth in a scalable and modular manner.

The following two examples show that until recently, architectures and design methods at both the deep-submicron and system levels have been often been *global* in nature. This hampers scalability, and in the next section we show that approaches that compose local solutions are becoming popular.

The physical communication between intellectual property blocks (IP) has made use of a mix of local and global wires. When timing constraints of the design must be verified, IPs cannot be checked independently because they are interconnected; correcting a timing violation in one IP may invalidate the timing of another. This process does not necessarily converge to a solution, and is design specific (not re-usable). This global timing closure problem results from a tight (timing) coupling [3].

The second example considers the model of time. Until now, the dominant design style has been synchronous; a global notion of time has been implemented by globally synchronous clocking. Increasing processing variations on a single IC make this style untenable in the future [4]. This will impact clocking regimes, but also the programming model. When using shared memory, which has been the dominant method to communicate between tasks, if tasks are not tightly synchronised (requiring a common, global time frame) then task scheduling may lead to interference and unpredictable behaviour.

### 1.4 Composing local subsolutions

To avoid the exponential complexity of global methods and solutions, there is increasing interest in subdividing global problems into *local*, *decoupled* problems, and then composing the local solutions. This requires, foremost, *compositionality* to assemble a global solution from local ones (Figure 4.1). But solutions must also be *scalable* (and likely hierarchical, Figure 4.1(c)) because there will be an exponential number of local solutions. This applies at all

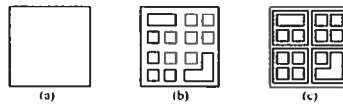


Figure 4.1. Divide global problem (a) into local problems that are composed flat (b) or hierarchically (c). Examples are GALS, local timing closure, local shared memory with global fifo communication or message passing, local busses or switches with global packet switching.

levels of SOC design: lay-out, timing verification, clocking, and programming. This trend commenced some time ago with IP-based re-use, platform-based design [5]. The interest in decoupling and composing local solutions can be observed in hierarchical lay-out and wiring, local clocking strategies such as GALS (globally asynchronous, locally synchronous) [6], software-programmable fixed IPs or tiles [7, 8], and synthesisable tiles for dedicated silicon [9] or FPGAs [10, 11]. At the task level, these scalable system architectures (including e.g. chip multiprocessing [12, 13, 14]) use tasks with local, independent address spaces and time frames that are composed by means of timing-independent fifo channels (for example, Kahn process networks for stream-based processing).

### 1.5 Networks on chip

All the approaches that advocate local solutions have a common reliance on a scalable and compositional communication medium to efficiently combine the large number of (hardware and software) IPs or subsystems in a working system. This challenge is addressed by networks on chip (NOC), which therefore play a pivotal role in future SOCs. NOCs help to solve both deep-submicron problems (timing closure, wiring, lay-out, etc.) and compositional design methods (services and protocol stacks). More detailed argumentation for the use of NOCs in future SOCs can be found in [15, 9, 16, 17, 2], and elsewhere in this volume.

In the following section we show that NOCs and QoS naturally combine to solve the two problems stated in this section: NOCs combine decoupled local solutions, and a service-based design style makes both the design process and the QoS of the resulting SOC more predictable.

## 2. NoCs and QoS: a synthesis

We advocate system design centred on NOCs and QoS for three reasons. First, services relieve the inherent tension between the *dynamic nature of future applications* and the user requirement for predictable services. This addresses concerns raised in Sections 1.1 and 1.2. Second, IP re-use and platform-based design aim to re-use applications and architectures by decoupling them. The use of NOCs and their associated network protocol stacks are a way to achieve this (cf. Section 1.5). Finally, it is our tenet that QoS, in particular *guaranteed services*, allow us to move towards *globally predictable, locally predictable* methods and solutions for SOC design. This, in combination with NOCs solves a need of recent architectures and design methods, introduced in Section 1.4. We discuss each point in turn.

### 2.1 Dynamic applications and predictable QoS

A QoS-based design style alleviates the demanding task for the system *designer*. He or she must plan systems that reliably provide users with the expected QoS for future applications. Moreover, systems must always be cost effective. Higher costs are acceptable when safety is critical, but they are always under pressure for consumer products, whether they be television sets or cars. Systems, especially SOCs, therefore have limited resources, and they must be shared and managed as the application unfolds its dynamic behaviour. Resource management depends on two phases: negotiation to obtain resources, followed by a steady state in which allocated resources are used. As applications become more dynamic, the first phase, *renegotiation*, will become more frequent. Thus, users can be given a predictable QoS by giving resource management and QoS a prominent place in system design.

### 2.2 Platform-based design, NoCs, and QoS

The aim of *platform*-based design [5, 18] is to reduce the cost of system design through re-use of applications and architectures. A platform decouples applications and system architectures, by defining a template architecture and programming model. In other words, by limiting the freedom in which an application can be implemented on an architecture, the interdependence of application and architecture is concentrated and reduced (Figure 4.2(a)). However, the convergence of applications entails an increasing diversity and dynamics in resource usage (such as communication and computation patterns), and this results in an increasing need for *differentiated services* [15]. It is therefore important that the platform offers the appropriate services (for communication, computation, power management, etc.). As a result, the communication infrastructure is a critical component of a platform, because it must solve the

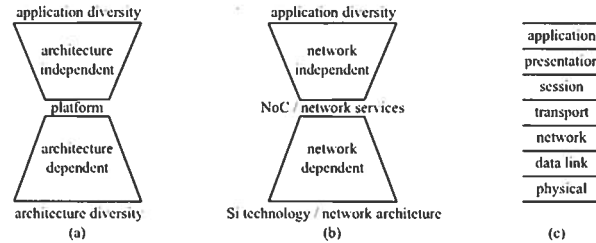


Figure 4.2. Decouple applications and architectures by means of a platform (a), or network (e.g. NOC, Internet) services (b), or network protocol stacks (c).

apparent contradiction of implementing diverse application behaviours with application-dependent IPs in an application-independent manner. Using a NOC for the platform interconnect tackles both problems: it integrates heterogeneous IPs in a standard fashion (in other words, the NOC services largely define the platform, Figure 4.2(a)); and it naturally provides differentiated services by means of a (partially application-dependent) protocol stack [17].

Figure 4.2(b) shows that NOCs offer a small set of services (“NoC services”), on top of which different kinds of communication can be implemented (e.g. shared-memory cache-coherent traffic, message passing). The OSI transport layer [19] is a natural place to separate the application and architecture, because it is the first network-independent layer. The set of services should be small, yet allow the upper layers in the protocol stack to offer differentiated services [20]. This method has been used successfully in computer networks [21], where the internet protocol (IP) plays a similar role. Higher layers offer more specialised protocols such as TCP and UDP, and FTP and HTTP. For NOCs, there are several proposals [22, 23, 24], but convergence has yet to occur.

Concluding, NOCs help IP re-use and platform-based design to combine IPs, and NOC services aid in decoupling applications and architectures.

### 2.3 QoS enables predictable composition

As we have seen in Section 1.3, to fully exploit Moore’s law, it is key to combine and control many local, perhaps autonomous, components in an efficient and flexible manner resulting in the required QoS. To localise timing, communication, data, scheduling, and so on, means that components (subsystems, IPs, tasks, threads, etc.) must first be identified. After composition, their interaction is controlled by means of arbitration, scheduling, or resource management. But we must address the apparent contradiction of providing *globally predictable* QoS (cf. Section 2.1) while current approaches tend to *globally unpredictable, locally predictable* (GULP!) regimes. How can global

timing guarantees be given when locally synchronous components are combined asynchronously [6]? How can global performance be quantified when tasks are combined in a latency-insensitive fashion [14, 7]? It is our tenet that QoS, in particular guaranteed services, allow us to move towards *globally predictable, locally predictable* methods and solutions. We elaborate this claim below, by first defining what role QoS plays, then stating how the local solutions (e.g. IP design) benefit, and finally how QoS eases the composition of local solutions.

**2.3.1 Services.** Effective steering of the limited resources in a NOC (cf. Section 2.1), and hence effective QoS, is contingent on a reliable reaction of resources to instructions. In other words, a predictable or guaranteed behaviour of system components is a prerequisite to providing the user with expected QoS, at whatever level. The essence of QoS-based system design is to restrict the interaction between components to well-defined services. Two phases can be distinguished: (a) negotiation, followed by (b) resource steering or scheduling, and observation or inspection. As an example, consider a microprocessor which negotiates a connection to a memory with guaranteed bandwidth, but without a constraint on transaction ordering. The NOC that offers communication services will honour the request if has sufficient resources available, and will reject it otherwise. Later, the microprocessor may renegotiate its connection, e.g. to a higher bandwidth. The NOC will release the resources of the connection for use by other connections, when it is closed by the microprocessor.

**2.3.2 Advantages for local solutions.** QoS-based interaction has a number of advantages for IP design. By limiting component interaction to a set of well-defined services, their interfaces are simplified because there are fewer eventualities to take into account. Moreover, failures of service users (IPs) are concentrated at the reconfiguration points: after a service provider (such as a NOC) has committed to the request (such as a connection with bounded jitter), its provision can be relied upon. Similarly, IPs do not interfere with each other because the service provider and user have a local contract. This allows components to be designed and implemented in isolation. An advantageous side effect of negotiation is that the requirements of IPs must be stated explicitly, aiding the design process.

**2.3.3 Advantages in composing local solutions.** We now consider the impact of QoS-based interaction when combining IPs to obtain a SOC. First, services that are guaranteed to an IP are not affected by other IPs in the network, making reasoning about the IP in isolation possible. This is essential for a *compositional construction* (design and programming) of SOCs.

Moreover, SOCs can be more *robust*, because rather than relying on cooperation to share resources, resource management enforces the contracts between service providers and users. An IP that (maliciously or erroneously) does not adhere to a cooperative protocol, such as TCP/IP, cannot, therefore, disturb the system as a whole [25]. Failure is therefore local in space (one IP) and time (at one of its reconfiguration points).

Next, when components are combined, the performance (i.e. the QoS) of their composition must be validated. This can be done by analysing the complete system implementation. However, this analysis may be too hard, because the behaviour of individual components or their interactions are complex to model accurately (e.g. traffic analysis of NOCs [26], cache behaviour [27]). *Statistical* approaches are therefore frequently used [28]. Unfortunately resource requests often do not fit models (e.g. bursty traffic versus fractal or normal traffic distributions [26]) invalidating the verification. The oxymoron “statistical guarantee” does therefore not guarantee a QoS, but implies a (usually post hoc) analysis relying on a statistical model of the resources and their usage. Instead, we propose to validate the composition of the local solutions at the level of services (their interface), instead of having to open them up, and consider their (global) combination. This *abstraction* is essential in reducing the verification state space.

Finally, services can make QoS provision *architecture independent*, because how a component, such as a NOC, offers its services is not relevant. This removes the need to second-guess the inner workings of a component, because its services describe all that is required to know. Interaction becomes prescriptive (state what is required, i.e. what must happen), rather than prognostic (try to predict the service provider’s behaviour) or reactive (act on how the service provider behaves).

We can therefore conclude that by abstracting local IP behaviours to their service requirements or provision, makes the global QoS of their composition more predictable and hence easier to reason about. This simplifies the design of SOCs.

### 3. On the cost of guaranteeing QoS

We have seen that a QoS-based approach is required to implement future applications, and to offer a predictable performance to users. In essence, offering a QoS requires a commitment. In this section, we present different levels of commitment, and their effect on predictability and cost in terms of resource usage. Although guaranteed services, which offer commitment, have many advantages over so-called best-effort services, which offer no commitment, we show that their combination is beneficial. The *ÆTHEREAL* NOC, described in the next section, is an example of that approach.

### 3.1 Different levels of commitment

As has been explained before, to offer a certain QoS with finite resources, the service provider and user negotiate to arrive at a contract, i.e. a *commitment* by the service provider to honour the request. If a commitment has been given, the service is *guaranteed*, otherwise it is a *best-effort* service. Commitment exists at several levels: 1) *correctness* and integrity of the result, if and when it is delivered. Examples are parity checking and error correcting codes for uncorrupted data transmission, and redundant computation with majority voting for safety-critical systems. 2) Promise of *completion* or delivery. This involves the cumulative availability (over time) of sufficient resources (e.g. memory, cpu cycles), and their performance (e.g. absence of deadlock or livelock). Note that a minimum number of resources may be required: for example, an in-place FFT needs to store all samples simultaneously, and a mobile communication may require a minimum battery charge to generate a sufficiently strong signal. 3) *Bounds* on the performance. Examples are the completion time (when is the result available), cumulative time to completion and its variations (e.g. bounded latency and jitter), and (peak and average) energy consumption.

Almost any form of commitment to progress leads to resource allocation (e.g. memory space, cpu cycles, communication bandwidth, battery power) and hence requires resource management. Moreover, levels of commitment depend on those below them. We illustrate this with two examples.

An IP connected to a NOC may not always be able to accept incoming data. Assuming its input buffers are finite, several solutions are possible, with direct consequences for the service level that can be offered. Packets that arrive at a full buffer are dropped, instantly precluding completion bounds such as latency and jitter guarantees. General computer networks typically use this approach. Alternatively, packets are not dropped and are left waiting in the network. Care then has to be taken to not introduce deadlock (e.g. if re-ordered packets that wait for free buffers do not overtake each other), or livelock (e.g. in deflection routing [29] packets keep moving, but may never arrive at their destination). In both cases, again, completion bounds may be in jeopardy. An approach to offer completion bounds (e.g. for a bounded communication latency) is to ensure packets never wait in the network by reserving buffer resources at the receiver, and by using end-to-end flow control to constrain the sender to never send more than the available space.

The second example concerns data transmission in unreliable media. This is a topic of importance in NOCs because wires suffer increasingly from interference, such as cross-talk and voltage drops. To ensure data is transported unchanged, it can be retransmitted when corrupted, or error correction can be used. The energy efficiency of both approaches is investigated by [30] for unreliable busses. What concerns us here, is that using retransmission takes a

variable, possibly unbounded, amount of time, whereas error correction takes place in a constant time. Therefore, time-related guarantees, such as minimum throughput, can only be given by the latter.

We conclude that the strictest guarantees, namely those involving performance bounds, pervade the system (all protocol layers): they cannot be grafted on as an afterthought [17, 31]. In NOCs we should and can choose communication protocols and styles to offer the required services, but, as the examples show, this requires vigilance.

### 3.2 Commitment and resource usage

The effect of the level of commitment on resource usage is illustrated in Figure 4.3. Suppose that the resource plotted vertically is bandwidth. For

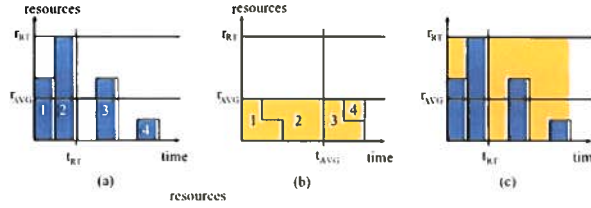


Figure 4.3. (a,b) Commitment to completion or bounded completion is reflected in resource usage. (c) Combining guaranteed services with best-effort services, discussed in Section 3.3.

real-time performance, the requested bandwidth must be offered in the same interval, see Figure 4.3(a). Thus, as many resources must be available as the largest request ( $r_{RT}$  at time  $t_{RT}$ ). With resources dimensioned for the average resource usage ( $r_{AVG}$  in Figure 4.3(b)), completion times may shift to the future: the peak resource request at time  $t_{RT}$  is only completed at time  $t_{AVG}$ , in the example. Depending on the required completion bound and variability in resource requests more or fewer resources can be added. However, more resources than  $r_{RT}$  do not improve performance, while fewer resources than  $r_{AVG}$  mean that no completion commitment can be given, because the backlog of requests keeps growing.

Architectures offering only best-effort services do not reserve resources, and hence can have a *better average resource utilisation*, at the cost of *unpredictable or unbounded worst-case behaviour*.

### 3.3 Both best-effort and guaranteed services

A service is guaranteed if a commitment is given, and best effort otherwise. This holds for individual services, not their ensemble. For example, data transport may be uncorrupted (commitment to correctness), and lossless (commit-

ment to delivery), and without throughput guarantees (best-effort throughput, i.e. no commitment to a completion bound). Moreover, a given service can be offered both with and without commitment, to flexibly use the available resources. For example, a data transport service can be offered both with and without completion bounds, to serve different users, in a single SOC. In Figure 4.3(c), the critical dark traffic, uses the guaranteed service, for real-time performance. The remaining  $r_{RT} - r_{AVG}$  resources are, on average, available to other traffic, for example with less strict completion bounds, or with no completion bound (best-effort completion).

Figure 4.3(c) shows that when resources must be dimensioned for the worst-case for a given service commitment (e.g. guaranteed latency), the resources can also be used to give less stringent commitments (e.g. guaranteed delivery), or even a best-effort service. A combination of best-effort and guaranteed services gives the advantages of guaranteed services to only part of the system, but the available resources are used more efficiently. In the next section we show how this can be put in practice, in the *ÆTHEREAL* NOC.

#### 4. The *Æthetical* network on chip

In this section we introduce the *ÆTHEREAL* NOC [17, 2]. It offers guaranteed services to obtain the advantages in composability, robustness, and so on of QoS-based design. These services are used for real-time and critical functions. The *ÆTHEREAL* NOC also provides best-effort services, to take advantage of their lower resource requirements and potentially better average performance. We describe the NOC services [24] in the next section.

In Section 4.2, we show how the services can be efficiently implemented, using a mix of time-division-multiplexed circuit switching, and packet switching [32, 31]. The programming model (connection creation and closing) and its advantages are also explained.

##### 4.1 *Æthetical* Services

The *ÆTHEREAL* NOC offers differentiated services with *connections*. A connection describes communication between one master and one or more slaves, with an associated service level, such as fifo transaction ordering, and maximum latency. Connections must be *created* stating the requested service level; this is the negotiation phase. The NOC either accepts or rejects the request for the connection. Connection acceptance may lead to resource reservations in the NOC, e.g. buffers or a link bandwidth percentage. After usage, *closing* a connection frees the resources. Different connections are created and closed independently, possibly at different points in time. Configurations can be computed at compile time (i.e. off-line), or at run time. To ensure that

the programming model scales as NOCs become larger, negotiations can be distributed.

**4.1.1 Transactions.** Once a connection has been created, the master initiates *transactions* by means of *requests* which zero or more slaves *execute*, perhaps leading to a *response*. Examples of transactions are read, write, acknowledged write, test and set, and flush. By offering these transactions the *ÆTHEREAL* transaction model is similar to existing bus protocols, to ease migration of IP from current interconnects to NOCs. However, to be able to take full advantage of increased NOC performance, transactions can also be pipelined, split, and posted [24].

**4.1.2 Connection Types.** The *ÆTHEREAL* NOC can offer three kinds of connections (Figure 4.4). A *simple* connection contains a master and

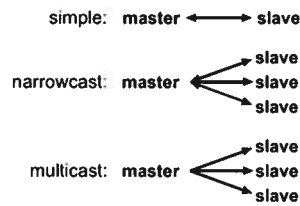


Figure 4.4. Connection types.

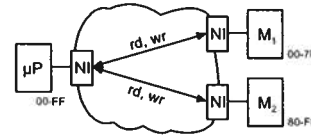


Figure 4.5. A narrowcast connection.

a single slave. The master initiates transaction which the slave executes, possibly resulting in a response (e.g. for a read). On a *narrowcast* connection, containing a master and one or more slaves, the request from the master is sent to and executed by exactly one slave. An example of the narrowcast connection is shown in Figure 4.5, where the master performs transactions on an address space that is mapped on two memory modules. Depending on the transaction address, a transaction is executed on one of the two memories. A *multicast* connection is a connection between one master and one or more slaves, in which requests are duplicated and each slave receives a copy of those requests. Currently, in a multicast connection no return messages are allowed, because of the volume of traffic they may generate (i.e., one response per destination), and the increased complexity in the master (because individual responses from slaves must be merged into a single response).

**4.1.3 Connection Properties.** We distinguish the following services, or connection properties: 1) data integrity, 2) transaction ordering, 3) transaction completion, 4) connection flow control, and 5) connection

throughput, latency, and jitter. A connection can request any combination of these properties (e.g. a throughput guarantee, flow control, but no transaction ordering). Recalling the levels of commitment, of Section 3.1, properties 1 and 2, above, commit to correctness (including order) of results. Completion is guaranteed by property 3 and 4 (4 implies 3). Connection latency and jitter give completion bounds. We discuss each property in turn.

1) *Data integrity* means that data is transported unchanged. We assume that data integrity is solved at the data link layer; every connection therefore offers data integrity. However, as noted in Section 3.1, this must be done in a way that supports higher-level commitments, in particular bounds on transport completion (latency).

2) *Transaction ordering*. Transaction orders are only defined on a single connection; transactions on different connections may be transported and executed in any order. In general, responses and requests may be reordered during their transport in the network. This means that requests (responses) may arrive in a different order at a single slave (master) than they were sent. (Note that we refer to requests and responses of different transactions. Within a single transaction, requests and responses are always ordered as follows: the master sends a request, a slave receives the request and executes it, the slave sends the response, the master receives the response.) We refer to [24] for a detailed analysis of different orderings. Here it suffices to state that three connections orderings are useful. *Unordered connections*, in which no order is assumed between any request and response. *Locally ordered connections*, where requests sent by the master for each slave are delivered to that slave in order they were sent. Requests for different slaves are unordered. Responses are delivered to the master in the order the requests were generated. For example, on a narrow-cast connection with multiple single-port memory slaves, the read and write order to a single memory is important, but not between memories. *Globally ordered connections*, in which requests for (responses from) all slaves are delivered to the slaves (master) in the order the requests were sent. This last ordering is not offered as a service, because local ordering is usually sufficient, and because the user can emulate global ordering by means of acknowledged transactions. Global ordering may be used when the order in which different devices (slaves) are programmed is of importance.

3) *Transaction completion* stipulates that transactions requesting a response (e.g. acknowledged write) guarantee that the master always receives a response. The response contains a) a report that the request was not delivered to the slave, or b) the response of the slave, after it successfully executed the request, c) a notification that the slave successfully executed the request, but the response was dropped, d) a report that the slave failed to execute the request. For connections with flow control, no data will be dropped, and transaction completion is automatic. Without flow control fewer resources are required in

the network, but data may be dropped. For example, it may be fine to drop a transaction, as long as you know it has been lost, so that you can resend it.

4) *Connection flow control* guarantees that data that is sent will fit in the buffers at the receiving end. End-to-end flow control is one of main techniques to avoid network congestion, if data cannot be dropped (cf. Section 3.1). If a slave is slower than its master, and the slave buffers fill up, then the master will be blocked until there is sufficient free space for a transaction. Similarly, a slave may be blocked by a slower master on the return path.

5) *Bounds on connection throughput, latency, and jitter*. The *ÆTHEREAL* NOC provides connections that guarantee a bandwidth per fixed time interval. Thus, a combined throughput, latency, and jitter guarantee is given. Depending the time interval, the latency and jitter bounds may be rather large. This is acceptable in many applications, such as audio and video streaming, where throughput is more important than latency. In the next section give the reasons for giving the throughput guarantee in this form.

The *ÆTHEREAL* services include transactions of bus protocols, to ease migration from current bus-based systems to NOCs. But where in OCP [23] and VCI [22] connections are used only to relax transaction ordering, we offer a more general connection property model. Not only are more properties considered, but the request and response communications can be independently configured (e.g. for flow control, and throughput guarantees), allowing more fine-grained resource management. Connection-based service provision therefore allows better differentiation of services, and allows users to make full use of NOC performance.

## 4.2 *Æthereal architecture*

The challenge of designing a NOC lies in finding a balance between the NOC services and their implementation complexity and cost. Moreover, as has been mentioned in Section 3.1, different services (levels of commitment) are dependent on each other. Offering time-related guarantees (a completion bound) influences the NOC to the core. The *ÆTHEREAL* NOC has both guaranteed and best-effort services, and an architectural challenge is how to combine these efficiently. In other words, how can *guaranteed worst-case behaviour be joined with good average resource usage*. In the following sections we describe two conceptually disjoint networks that each solve one part. An efficient combination is then presented. Both networks contain two components: routers and network interfaces. Routers transport data and can be described at the OSI network layer. As the *ÆTHEREAL* services are offered to IP at the transport layer and are end to end (master to slave and vice versa), network interfaces are required to bridge the network layer and transport layer views on communication.

**4.2.1 A guaranteed-throughput router.** To give time-related guarantees on a connection, such as throughput guarantees (on a finite time scale) or latency bounds, the interference of other traffic in the NOC must be limited and characterised. Circuit switching gives strong guarantees but at a high cost: connections have to be dimensioned for the worst case traffic. Time-division multiplexed circuit switching reduces this cost, but creating and closing of circuits still takes much time, and grows with the size of the network. The life-time of circuits must grow to amortise this cost [33]. Rate- and deadline-based scheduling [34, 35] can characterise the worst-case contention and offer bounds on latency, by regulating the inflow of data, but at the a high buffer cost.

The *ÆTHEREAL* NOC uses *contention-free routing*, which is based on a time-division-multiplexed circuit-switching approach, where one or more circuits are set up for a connection, which is assume to be relatively long-lived. Guaranteed throughput (GT) packets never use the same link at the same time, i.e. all contention is avoided. This can be achieved by controlling both the time GT packets enter the network, and their speed in the network. All routers logically have a common notion of time, embodied in a slot counter (see Figure 4.6). GT packets propagate at the rate of one router per slot counter increment. By regulating the time a GT packet is injected in the network, it is in effect scheduled to use each successive link in its path in a successive slot, see Figure 4.6. This method avoids the emptying and filling of circuits between

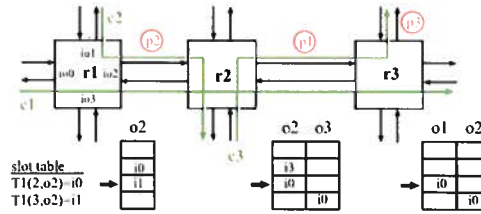


Figure 4.6. Three routers, all in slot  $s = 3$ , switch packets  $p_i$  of circuit  $c_i$  to outputs  $o_i$ , for  $T(s, o_i) = c_i$ . For each table, only the relevant columns (outputs) are shown. Input/output link pairs are labelled  $io_i$ .

switching of circuits. Conceptually it resembles input queuing with store-and-forward routing, which results in low buffering costs because GT packets are small. The end-to-end (network interface to network interface) latency is the number of hops multiplied by the size of the GT packet. This is minimal, once the packet has entered its slot. This model also allows multicast circuits. GT packets are headerless, and are routed by means of slot tables in every network interface and router. Section 4.2.3 explains how the slot tables are programmed.

**4.2.2 A best-effort router.** The best-effort router uses packet switching and has a more conventional structure. Our experiments [31] indicate that both input queueing with worm-hole routing or virtual-cut-through routing, and virtual output queueing with worm-hole routing are feasible, in terms of buffering costs. Input queueing uses fewer buffers, but suffers from head-of-line blocking. Virtual output queueing has a higher performance but at the cost of more buffers.

**4.2.3 A combined GT-BE router.** The logically separate guaranteed (GT) and best-effort (BE) routers are combined (Figure 4.7(a)) to share the router resources (e.g. switch and data path, Figure 4.7(b)), and to obtain the advantages of both. The GT router offers a fixed end-to-end latency for its traffic, which has the highest priority, enforced by the arbiter. The BE router

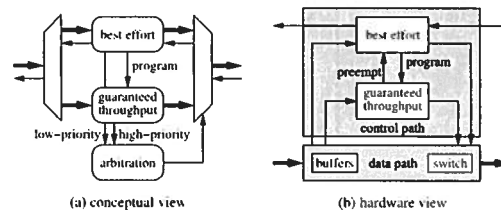


Figure 4.7. Two views of the combined GT-BE router.

uses all the bandwidth (slots) that has not been reserved or is not used by GT traffic. This allows the sharing of links and data path. Resources are therefore never left unused, when there is data, cf. Figure 4.3(b). They are either used for critical traffic with real-time requirements (for which a completion bound has been given), or for best-effort traffic (without a completion bound).

To allow distributed and scalable programming of connections, the GT router slot tables (cf. Figure 4.6), are programmed by means of BE packets, see the arrow “program” in Figure 4.7(a&b). This can be done in a pipelined and concurrent (multiple simultaneous negotiations, also from the same the source), and distributed (active in multiple routers) fashion. Negotiations, resulting in slot allocations, can be done at compile time, and be configured deterministically at run time. Negotiations can also be done at run time, centrally or distributed.

**4.2.4 A network interface.** *ÆTHEREAL* network interfaces convert the OSI network layer services of the routers to transport layer services for the user. All connection properties (cf. Section 4.1) that are end-to-end are implemented by the network interfaces. These are: reordering, transaction

completion, and flow control. Unordered connections require that transaction identifiers are used, these are added at the network interfaces. For locally ordered connections, reordering buffers are required in the network interfaces. Since our routers do not reorder traffic in a connection, this is only required for narrowcast connections at the master for responses. Transaction completion also requires resource reservations in the network interfaces. Flow control serves to prevent overflow of buffers at the slave or master network interface. In a credit-based approach, this requires state (credits for the amount of space available), and additional communication (when data is consumed credits are returned to the producer). This is taken care of by the network interface.

IPs negotiate with network interfaces to obtain connections with certain properties. For this network interfaces may reserve resources, such as network interface buffers and credit counters, and slots in router tables.

## 5. Related work

Differentiated services have received attention in general computer networks in the context of ATM [36] and the Internet [25]. Real-time traffic schemes have been described using rate-based and deadline-based scheduling [34, 35].

The differences between single-hop on-chip communication such as busses and switches and NOCs are described in [24]. Bus protocols such as [22] often have time-division multiplexing and/or priorities added, such as MicroNetwork [37], to give throughput guarantees. To reduce the guaranteed but average high latency, statistical approaches can be used (e.g. Lotterybus [38]). The bound on completion time is then lost (cf. Section 3.1). Switches [39, 40], also single-hop interconnects, can also offer performance guarantees.

On-chip busses with bridges [41, 42, 43], but also [44], are potentially full-blown NOCs. By judiciously placing restrictions on topology, bridging, and transaction models, many problems that arise in general networks can be avoided. For example, an appropriate topology simplifies routing to a single path, and avoids transaction reordering. Limiting a master to a single outstanding transaction has the same effect. By not buffering in bridges, effectively an end-to-end circuit is set up per transaction, avoiding reordering and the need for end-to-end flow control. In both cases, the increasing latency of larger networks make these simplifying assumptions untenable in the future [33, 45].

The octagon interconnect [46] is an interesting combination of packet and circuit switching. An octagon corresponds to a single  $8 \times 8$  router that uses circuit switching per transaction (or equivalently, packets of unlimited size), with a high performance for a connection in an octagon. An individual octagon can be made larger (the so-called core and edge node strategy), but this does not scale to larger networks for the reasons mentioned above. Packet switching can then be used instead, by using so-called bridge and member nodes strategy.

At this point all general NOC issues appear. This NOC seems a best-effort architecture, at least when multiple octagons are used.

The Spin [28, 29] NOC uses packet switching with worm-hole routing and input queuing in a fat tree topology. It is a scalable network for data transport, but uses a parallel network (bus) for control. It is a best-effort network, and is optimised for average performance (e.g. by the use of optimistic flow control coupled with deflection routing). Commitment is given for packet delivery, but latency bounds are only given statistically.

Dally et al [9] describes a preplaced mesh NOCs, with IPs that are synthesised in the tiles. It uses packet switching on lossless virtual channels with end-to-end flow control. No details are given on the programming model, but guaranteed throughput (and latency) services are supported.

In the context of FPGAs using bit-level circuit switching to implement inter-IP communication is expensive. A first optimisation is to use coarser circuits [47] to reduce the cost. A packet-switched interconnect with a predefined set of services allows sharing of communication resources, just like for ASICs, and enables dynamic reconfiguration [10, 11]. The NOC can be synthesised [10], but a preplaced hard-wired NOC is the next logical step.

New SOC architectures that rely on NOCs include chip multiprocessing [12, 13, 14, 7, 8, 48], to interconnect the homogeneous or heterogeneous tiles, and network processors [49].

## 6. Conclusions

We observe that, although future applications will be more dynamic, *users* expect embedded systems to behave predictably. A design style based on guaranteed quality of services (QoS) can solve this apparent contradiction. *Designers* of systems on chip (SOC) use networks on chip (NOC) to keep up with Moore's law. NOCs solve both deep-submicron problems (e.g. signal integrity), and narrow the design productivity gap (the efficiency with which we design SOCs) by dividing global problems into local, decoupled problems, e.g. GALS.

The combination of NOCs and QoS is natural, through network protocol stacks, and beneficial for several reasons. It enables IP re-use and platform-based design to decouple applications and architectures. QoS-based design also ensures that when global problems, such as clocking, are solved by combining local, decoupled solutions (e.g. GALS), the combination has a globally predictable behaviour, as required by the user.

There are several levels of QoS *commitment*, building on each other: correctness (e.g. uncorrupted transport), completion (e.g. packet delivery), and completion bounds (e.g. maximum latency). A service without commitment (a *best-effort* service, such as unbounded latency), can have a better average

resource utilisation than a *guaranteed* service, but at the cost of unpredictable or unbounded worst-case behaviour. The *combination of best-effort and guaranteed services is advantageous*: critical parts of the system are predictable, while resources are used more efficiently.

The ÆTHEREAL NOC combines best-effort and guaranteed services at different levels. Its programming model and architecture are also described. The ÆTHEREAL NOC is at the basis of a QoS-based design style, as advocated in this chapter.

## References

- [1] Semiconductor Industry Association. *The International Technology Roadmap for Semiconductors*. 2001.
- [2] Paul Wielage and Kees Goossens. Networks on silicon: Blessing or nightmare? In *Euromicro Symposium On Digital System Design*, Dortmund, Germany, September 2002. Keynote speech.
- [3] T. N. Theis. The future of interconnection technology. *IBM journal of research development*, 44(3):379–390, May 2000.
- [4] Marcel J. Pelgrom, Hans P. Tuinhout, and Maarten Vertregt. Transistor matching in analog CMOS applications. In *IEDM*, pages 915–918, 1998.
- [5] K. Keutzer, S. Malik, A. Richard Newton, Jan M. Rabaey, and A. Sangiovanni-Vincentelli. System-level design: Orthogonalization of concerns and platform-based design. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 19(12):1523–1543, 2000.
- [6] Jens Mutersbach, Thomas Villiger, and Wolfgang Fichtner. Practical design of globally-asynchronous locally-synchronous systems. In *6th International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, April 2000.
- [7] Paul Stravers and Jan Hoogerbrugge. Homogeneous multiprocessing and the future of silicon design paradigms. In *VLSI-TSA*, 2001.
- [8] Ken Mai, Tim Paaske, Nuwan Jayasena, Ron Ho, William J. Dally, and Mark Horowitz. Smart memories: A modular reconfigurable architecture. In *ISCA*, June 2000.
- [9] William J. Dally and Brian Towles. Route packets, not wires: On-chip interconnection networks. In *Design Automation Conference*, pages 684–689, June 2001.
- [10] Théodore Marescaux, Andrei Bartic, Dideriek Verkest, Serge Vernalde, and Rudy Lauwereins. Interconnection networks enable fine-grain dynamic multitasking on FPGAs. *FPL*, 2002. LNCS 2438.

- [11] Edson L. Horta, John W. Lockwood, David E. Taylor, and David Parlour. Dynamic hardware plugins in an FPGA with partial run-time configuration. In *Design Automation Conference*, June 2002.
- [12] Lance Hammond, Basam A. Hayfeh, and Kunle Olokotun. A single-chip multiprocessor. *IEEE Computer*, pages 79–85, September 1997.
- [13] Jaehyuk Huh, Stephen W. Keckler, and Doug Burger. Exploring the design space of future CMPs. In *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2001.
- [14] Eylon Caspi, André DeHon, and John Wawrzynek. A streaming multi-threaded model. In *Third Workshop on Media and Stream Processors (MSP-3)*, December 2001.
- [15] M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabaey, and A. Sangiovanni-Vincentelli. Addressing the system-on-a-chip interconnect woes through communication-based design. In *Design Automation Conference*, pages 667–672, June 2001.
- [16] Luca Benini and Giovanni De Micheli. Networks on chips: A new SoC paradigm. *IEEE Computer*, 35(1):70–80, 2002.
- [17] K. Goossens, J. van Meerbergen, A. Peeters, and P. Wielage. Networks on silicon: Combining best-effort and guaranteed services. In *Proceedings of Design Automation and Test Conference in Europe*, pages 423–425, March 2002.
- [18] A. Ferrari and A. Sangiovanni-Vincentelli. System design: traditional concepts and new paradigms. In *International Conference on Computer Design*, pages 2–12, 1999.
- [19] J. D. Day and H. Zimmerman. The OSI reference model. In *Proceedings of the IEEE*, volume 71, pages 1334–1340, 1983.
- [20] K. G. W. Goossens and O. P. Gangwal. The cost of communication protocols and coordination languages in embedded systems. In Farhad Arbab and Carolyn Talcott, editors, *Coordination languages and models*, number 2315 in *Lecture notes in computer science*, pages 174–190. Springer Verlag, April 2002.
- [21] Steve Deering. Watching the waist of the protocol hourglass. In *6th IEEE International Conference on Network Protocols*, October 1998. Keynote speech.
- [22] VSI Alliance. Virtual component interface standard, 2000.
- [23] OCP International Partnership. Open core protocol specification, 2001.
- [24] Andrei Rădulescu and Kees Goossens. Communication services for networks on silicon. In Shuvra Bhattacharyya, Ed Deprettere, and Juer-

- gen Teich, editors, *Domain-Specific Processors: Systems, Architectures, Modeling, and Simulation*. Marcel Dekker, December 2002.
- [25] Vijay P. Kumar, T. V. Lashman, and Dimitrios Stiliadis. Beyond best effort: Router architectures for the differentiated services of tomorrow's internet. *IEEE Communications Magazine*, pages 152–164, May 1998.
- [26] Girish Varatkar. Traffic analysis for on-chip networks design of multimedia applications. In *Design Automation Conference*, June 2002.
- [27] Sungjoo Yoo, Kyoungseok Rha, Youngchul Cho, Jinyong Kung, and Kiyoungh Choi. Performance estimation of multiple-cache IP-based systems: Case study of an interdependency problem and application of an extended shared memory model. In *International Workshop on Hardware/Software Codesign*, May 2002.
- [28] Pierre Guerrier and Alain Greiner. A generic architecture for on-chip packet-switched interconnections. In *Proceedings of Design Automation and Test Conference in Europe*, pages 250–256, 2000.
- [29] Pierre Guerrier. *Un Réseau D'Interconnexion pour Systèmes Intégrés*. PhD thesis, Université Paris VI, March 2000.
- [30] Davide Bertozzi, Luca Benini, and Giovanni De Micheli. Low power error resilient encoding for on-chip data buses. In *Proceedings of Design Automation and Test Conference in Europe*, March 2002.
- [31] E. Rijpkema, K. Goossens, A. Rădulescu, J. van Meerbergen, P. Wielage, and E. Waterlander. Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip. In *Proceedings of Design Automation and Test Conference in Europe*, March 2003.
- [32] Edwin Rijpkema, Kees Goossens, and Paul Wielage. A router architecture for networks on silicon. In *Proceedings of Progress 2001, 2nd Workshop on Embedded Systems*, Veldhoven, the Netherlands, October 2001.
- [33] André DeHon. Robust, high-speed network design for large-scale multiprocessing. A.I. Technical report 1445, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, September 1993.
- [34] Hui Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proceedings of the IEEE*, 83(10):1374–96, October 1995.
- [35] Jennifer Rexford, John Hall, and Kang G. Shin. A router architecture for real-time communication in multicomputer networks. *IEEE Transactions on Computers*, 47(10):1088–1101, October 1998.
- [36] ATM Forum. *ATM User-Network Interface Specification*. Prentice Hall, July 1994. Version 3.1.
- [37] Drew Wingard. MicroNetworks-based integration for SOCs. In *Design Automation Conference*, 2001.

- [38] Kanishka Lahari, Anand Raghunathan, and Ganesh Laskhminarayana. Lotterybus: A new high-performance communication architecture for system-on-chip designs. In *Design Automation Conference*, June 2001.
- [39] Jeroen A.J. Leijten, Jef L. van Meerbergen, Adwin H. Timmer, and Jochen A.G. Jess. Stream communication between real-time tasks in a high-performance multiprocessor. In *Proceedings of Design Automation and Test Conference in Europe*, pages 125–131, 1998.
- [40] Paul J.M. Havinga. *Mobile Multimedia Systems*. PhD thesis, University of Twente, The Netherlands, February 2000.
- [41] Kyeong Keol Ryu, Eung Shin, and Vincent J Mooney. A comparison of five different multiprocessor SoC bus architectures. In *Euromicro*, 2001.
- [42] Tycho van Meeuwen, Arnout Vandecappelle, Allert van Zelst, Francky Catthoor, and Diederik Verkest. System-level interconnect architectures exploration for custom memory organizations. In *International Symposium on System Synthesis*, pages 13–18, October 2001.
- [43] Milenko Drinic, Darko Kirovski, Seapahn Meguerdichian, and Miodrag Potkonjak. Latency-guided on-chip bus network design. In *Proc. of IEEE/ACM International Conference on Computer Aided Design*, pages 420–423, November 2000.
- [44] John Bainbridge and Steve Furber. CHAIN: A delay-insensitive chip area interconnect. *IEEE Micro*, 22(5), 2002.
- [45] Frederic Chong, Henry Minsky, André deHon, Matthew Becker, Samuel Peretz, Eran Egozy, and Frank F. Knight, Jr. Metro: A router architecture for high-performance, short-haul routing networks. In *International Symposium on Computer Architecture*, April 1994.
- [46] Faraydon Karim, Anh Nguyen, Sujit Dey, and Ramesh Rao. On-chip communication architecture for OC-768 network processors. In *Design Automation Conference*, June 2001.
- [47] André DeHon. Rent's rule based switching requirements. In *SLIP*, April 2001. Extended abstract.
- [48] Shashi Kumar, Axel Jantsch, Juha-Pekka Soininen, Martti Forsell, Mikael Millberg, Johnny Öberg, Tiensyrjä, and Ahmed Hemani. A network on chip architecture and design methodology. In *ISVLSI*, 2002.
- [49] David Whelihan and Herman Schmit. Memory optimization in single chip network switch fabrics. In *Design Automation Conference*, June 2002.

# Exhibit C

# Trade-offs in the design of a router with both guaranteed and best-effort services for networks on chip

E. Rijpkema, K. Goossens, A. Rădulescu, J. Dielissen, J. van Meerbergen, P. Wielage and E. Waterlander

**Abstract:** Managing the complexity of designing chips containing billions of transistors requires decoupling computation from communication. For the communication, scalable and compositional interconnects, such as networks on chip (NoC), must be used. It is shown that guaranteed services are essential in achieving this decoupling. Guarantees typically come at the cost of lower resource utilisation. To avoid this, they must be used in combination with best-effort services. The key element of this NoC is a router consisting conceptually of two parts; the so-called guaranteed-throughput (GT) and best-effort (BE) routers. The GT and BE router architectures are combined in an efficient implementation by sharing resources. The trade-offs between hardware complexity and efficiency of the combined router are shown that motivate the choices. The reasoning for the trade-offs is validated with a prototype router implementation. A layout is shown of an input-queued wormhole  $5 \times 5$  router with an aggregate bandwidth of 80 Gbit/s. It occupies  $0.26 \text{ mm}^2$  in a  $0.13 \text{ }\mu\text{m}$  technology. This shows that our router provides high performance at reasonable cost, bringing NoCs one step closer.

## 1 Introduction

Recent advances in technology raise the challenge of managing the complexity of designing chips containing billions of transistors. A key ingredient in tackling this challenge is decoupling the computation from communication [1, 2]. This decoupling allows IPs (the computation part) and the interconnect (the communication part) to be designed independently from each other.

In this paper, we focus on the communication part. Existing interconnects (e.g. buses) may no longer be feasible for chips with many IPs, because of the diverse and dynamic communication requirements of advanced current and future applications. *Networks on a chip* (NoC) are emerging as an alternative to existing on-chip interconnects because they

- (a) structure and manage global wires in new deep-submicron technologies [3–7]
- (b) share wires, lowering their number and increasing their utilisation [6, 7]
- (c) can be energy efficient and reliable [4, 8]
- (d) are scalable when compared to traditional buses [9].

Decoupling the computation from communication requires that the *services* that IPs use to communicate are well defined, and hide the implementation details of the interconnect [1], Fig. 1a. NoCs help, because they

are traditionally designed using layered protocol stacks [10], where each layer provides a well-defined interface which decouples service usage from service implementation [2, 5], Fig. 1b.

In particular, *guaranteed services* are essential because they make the requirements on the NoC explicit and limit the possible interactions of IPs with the communication environment. IPs can also be designed independently because their use of guaranteed services is not affected by the interconnect or by other IPs. This is essential for a compositional construction (design and programming) of systems on chip (SoC). Moreover, failures are restricted to the IP configuration phase (a service request is either granted or denied by the NoC), which simplifies the IP programming model [7]. We view the guaranteed services to be offered by an interconnect as a requirement from the applications, Fig. 1c.

The drawback of using guaranteed services is that they require resource reservations for worst-case scenarios. This is not acceptable in an SoC where cost constraints are typically very tight, Fig. 1d. Therefore, we also provide *best-effort* services to exploit the network capacity that is left over, or reserved but unused. Guaranteed services are then used for the critical (e.g. real-time) communication, and best-effort services are used for noncritical communication.

The combination of guaranteed and best-effort classes is known from general computer network research [11], but not for on-chip networks. As on-chip and off-chip networks have different characteristics, the trade-offs in their design are different. In this paper, we present the trade-offs between hardware complexity and efficiency for networks on chip, and motivate our choices.

We present a prototype router architecture that reflects one particular set of design choices. It has an aggregate bandwidth of 80 Gbit/s and its layout in a  $0.13 \text{ }\mu\text{m}$

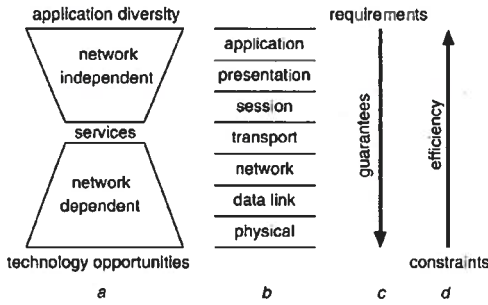


**Fig. 1** Network services

- a Hide the interconnect details and allow the construction of diverse applications on top of them
- b Are built using a layered approach
- c Are driven by the application requirements
- d Their efficiency relies on technology and network organisation

technology occupies  $0.26\text{mm}^2$ . We list other feasible variations that either increase performance or lower the router cost.

In this paper, we first list a set of network-independent communication services that are essential in chip design (Section 2). Then, we show the trade-offs between efficiency and cost that we make in our NoC. In Section 3, we present some general network-related issues that are used in the Sections that follow. In Section 4, we zoom into the internals of the key component of our NoC; a router that efficiently provides both guaranteed and best-effort services. In Section 5, we demonstrate the feasibility of our router design through a prototype implementation in a  $0.13\text{ }\mu\text{m}$  technology.

## 2 Services

The NoC services that we consider essential for chip design are [12]:

- *data integrity*, meaning that data is delivered uncorrupted
- *lossless data delivery*, which means no data is dropped in the interconnect
- *in-order data delivery*, which specifies that the order in which data is delivered is the same order in which it has been sent
- *throughput* and *latency* services that offer time-related bounds.

As motivated in Section 1, guaranteed services are essential to simplify IP design and integration. With the current technology we assume data integrity is solved at the data-link layer. All the other services can be guaranteed or not on request. In the following Section, we describe briefly how these services are provided by our NoC, and in Section 4 we describe how our router architecture enables an efficient implementation of these services.

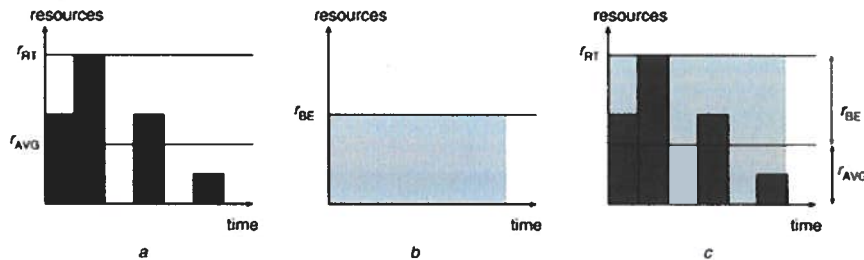


**Fig. 2** Guaranteed services requires worst-case ( $r_{RT}$ ) resource allocation (here: bandwidth) Best-effort service of (b) consumes the unused bandwidth of (a), resulting in (c)

*Guaranteed services* require resource reservation for worst-case scenarios, which can be expensive. For example, guaranteeing throughput for a stream of data implies reserving bandwidth for its peak throughput, even when its average is much lower. As a consequence, when using guarantees, resources are often underutilised.

*Best-effort services* do not reserve any resources, and hence provide no guarantees. Best-effort services use resources well because they are typically designed for average-case scenarios instead of worst-case scenarios. They are also easy and fast to use, as they require no resource reservation. Their main disadvantage is their unpredictability; one cannot rely on a given performance (i.e. they do not offer guarantees). In the best case, if certain boundary conditions are assumed, a statistical performance can be derived.

An example of diverse service requirements is that of a video processing IP; it will typically require a lossless, in-order video stream with guaranteed throughput, but possibly allows corrupted samples. Another example is cache updates which require uncorrupted, lossless, low-latency data transfer, but ordering and guaranteed throughput are less important.

The requirements for guaranteed services and the efficiency constraint (i.e. good resource utilisation) seem conflicting. Our approach to a predictable and low-cost interconnect is to integrate the guaranteed and best-effort services in the same interconnect. Guaranteed services would be used for critical traffic and best-effort services used for noncritical traffic. In this way the best-effort services can exploit the available resources left over by the guaranteed services. This is illustrated in Fig. 2 and is described in more detail in [13].

To meet the bandwidth requirements of a guaranteed service, the network must be dimensioned worst-case for that service, Fig. 2a. All bandwidth that is not used, i.e. the white space in Fig. 2a, can be used for a best-effort service in Fig. 2b. The best-effort traffic is delivered in bounded time when its volume is less than the white space in Fig. 2a.

For example a video processing IP requires a lossless, in-order video stream with guaranteed throughput, but possibly allows corrupted samples. Another example is cache updates which require uncorrupted, lossless, low-latency data transfer, but ordering and guaranteed throughput are less important. In Section 4.3 we show how integrated guaranteed and best-effort services efficiently use common resources. In the remainder of this Section we analyse the minimum level of abstraction at which the communication services must be offered to hide the network internals.

Traditionally, network services have been implemented and offered using a layered protocol stack, typically aligned to the ISO-OSI reference model [9], Fig. 1b. NoCs also take this approach [2, 4, 5, 7], because the protocol stack

concept structures and decomposes the service implementation, and aid the service positioning.

To achieve the decoupling of computation from communication, the communication services must be offered at least at the level of the transport layer in the OSI reference model. It is the first layer that offers end-to-end services, hiding the network details; see Fig. 1.

The lowest three layers in the protocol stack, namely physical, data-link, and network layers, are network specific. Therefore, these services should not be visible to the IPs when decoupling of computation from communication is desired. However, these layers are essential in implementing the services because constructing guarantees without guarantees at the layer below is either very expensive, or even impossible. For example, implementing a lossless communication on top of a lossy service requires acknowledgment, data retransmission and filtering duplicated data. This leads to an increase in traffic, and possibly larger buffer space requirements. Even worse, providing guarantees for time-related services is impossible if lower layers do not offer these guarantees. For example, latency cannot be guaranteed if communication at a lower layer is lossy. As a consequence, guarantees can only be built *on top of* guarantees, Fig. 1. Similarly, a layer's efficiency is based on efficient implementations of the layers below it, Fig. 1.

### 3 Networks on chip

General computer network research is a mature research field [10] which has many issues in common with NoCs. However, two significant differences between computer networks and on-chip networks make the trade-offs in their design very different [6]. First, routers of a NoC are more resource-constrained than those in a computer network, in particular in the control complexity and in the amount of memory. Secondly, communication links of a NoC are relatively shorter than those in computer networks, allowing tight synchronisation between routers.

To place the work described in this paper in context see Fig. 3. The services that are described in the previous Section are provided by the network interfaces. It is these services that are visible to the users (IP) of the network.

The scope of this paper is limited to the router network and we identify three important issues in its design. These are the *switching mode*, *contention resolution* and *network flow control*. Equally important are *end-to-end flow control* and *congestion control*, but these are handled by the

network interfaces and hence are out of the scope of this paper. Moreover, we assume guaranteed data integrity at the link level and retain it at the network layer and higher.

#### 3.1 Switching mode

The *switching mode* of a network specifies how data and control are related. We distinguish *circuit switching* and *packet switching*. In circuit switching data and control are separated. The control is provided to the network to *set up a connection*. This results in a *circuit* over which all subsequent data of the connection is transported. To better share the potential bandwidth of the links one can time-division multiplex circuits over the network. Circuit-switched networks inherently offer time-related guaranteed services after resources are reserved during the connection setup.

In *packet switching* data is divided into *packets* and every packet is composed of a control part, the *header*, and a data part, the *payload*. Network routers inspect, and possibly modify, the headers of incoming packets to switch the packet to the appropriate output port. Since in packet switching the packets are self-contained, there is no need for a setup phase to allocate resources. Therefore, best-effort services are naturally provided by packet switching.

#### 3.2 Contention resolution

When a router attempts to send multiple data items over the same link at the same time *contention* is said to occur. As only one data item can be sent over a link at any point in time, a selection among the contending data must be made; this process is called contention resolution.

In circuit switching, contention resolution takes place at set up at the granularity of connections, so that data sent over different connections do not conflict. Thus, there is no contention during data transport, and time-related guarantees can be given.

In packet switching contention resolution takes place at the granularity of individual packets. Because packet arrival cannot be predicted contention cannot be avoided. It is resolved dynamically by scheduling which data items are sent in turn. This requires data storage in the router (Section 4.2.1) and delays the data in an unpredictable manner which complicates the provision of guarantees (Section 4.1.1).

#### 3.3 Network flow control

*Network flow control*, also called *routing mode*, addresses the limited amount of buffering in routers and data acceptance between routers. In circuit switching connections are set up. The data sent over these connections is always accepted by the routers and hence no network flow control is needed. In packet switching, data items must be buffered at every router before they are sent on. Because routers have a limited amount of buffering, they accept data only when they have enough space to store the incoming data.

There are three types of network flow control, namely *store-and-forward*, *virtual cut-through* and *wormhole routing*. In store-and-forward routing, an incoming packet is received and stored in its entirety before it is forwarded to the next router. This requires storage for the complete packet and implies a per-router latency of at least the time required for the router to receive the packet.

In virtual cut-through routing, a packet is forwarded as soon as the next router guarantees that the complete packet will be accepted. When no guarantee is given, the router

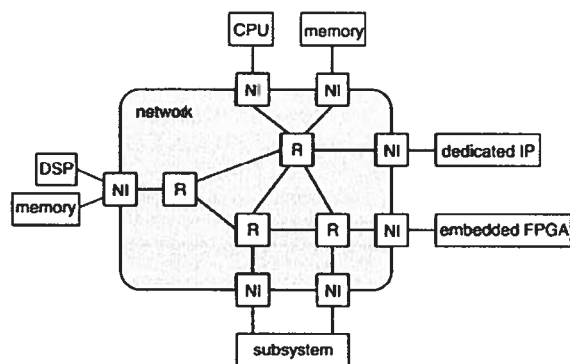


**Fig. 3** SoC composed of heterogeneous IP together with a NoC. NoC comprises interconnected routers (R) and network interfaces (NI) that connect the IP to the routers

must be able to store the whole packet. Thus, virtual cut-through routing requires buffer space for a complete packet, like store-and-forward routing, but allows lower-latency communication.

In wormhole routing packets are split into so-called *flits* (flow control digits). A flit is passed to the next router when the flit can be accepted, even when there is not enough buffer space for the complete packet. As soon as a flit of a packet is sent over an output port, that output port is reserved for flits of that packet only. When the first flit of a packet is blocked, the trailing flits can therefore be spread over multiple routers, blocking the intermediate links. Wormhole routing requires the least buffering (buffer flits instead of packets) and also allows low-latency communication. However, it is more sensitive to deadlock and generally results in lower link utilisation than virtual cut-through routing.

To allow low latency we consider both virtual cut-through and wormhole routing, which are both feasible in terms of buffer area, as shown in Section 5.

#### 4 Combined GT-BE router

Section 2 defines our requirements for NoCs in terms of services that are to be offered, in particular, both guaranteed and best-effort services. Using the general network issues of the previous Section we show in the following two Subsections that the guaranteed and best-effort services can conceptually be described by two independent router architectures. The combination of these two router architectures is efficient and has a flexible programming model, as described in Section 4.3. Section 5 then shows a prototype implementation.

##### 4.1 GT router architecture

Our guaranteed-throughput (GT) router guarantees uncorrupted, lossless and ordered data transfer, and both latency and throughput over a finite time interval. As mentioned earlier, data integrity is solved at the data-link layer; we do not address it further. The GT router is lossless because we use a variant of circuit switching, described in the following Section. Data is transported in fixed-size blocks. As only one block is stored per input in the GT router, data items remain ordered per connection. We now turn to the more challenging time-related guarantees, namely throughput and latency.

**4.1.1 Time-related guarantees:** Latency is defined as the duration as a packet is transported over the network. Guaranteeing latency, therefore, means that a worst-case upper bound must be given for this time. We define throughput for a given producer-consumer pair as the amount of data transported by the network over a finite, fixed time interval. Guaranteeing throughput means giving a lower bound.

We observe that guaranteeing latency even in a lossless router is difficult because contention requires scheduling and hence cause delays. Guaranteeing throughput is less problematic. Rate-based packet switching (for an overview see [14]) offers guaranteed throughput over a finite period, and hence a latency bound. This bound is very high, however, and the cost of buffering is also high. Deadline-based packet switching [15] offers preferential treatment for packets close to their deadline. This allows differential latency guarantees (under certain admissible traffic assumptions), but also at high buffer costs.

Circuit switching solves the contention at setup, so naturally providing guaranteed latency and throughput.

Circuits can be pipelined to improve throughput, at the cost of additional buffering and latency. Time division multiplexing connections over pipelined circuits additionally offers flexibility in bandwidth allocation. This requires a logical notion of router synchronicity, which is possible because a NoC is better controllable than a general network. We explain this variation in more detail in the following Subsection. The associated programming model is described in Section 4.3.2.

**4.1.2 Contention-free routing:** A router uses a *slot table* to

- (a) avoid contention on a link
- (b) divide up bandwidth per link between connections
- (c) switch data to the correct output.

Every slot table  $T$  has  $S$  time slots (rows) and  $N$  router outputs (columns). There is a logical notion of synchronicity; all routers in the network are in the same fixed-duration slot. In a slot  $s$  at most one *block* of data can be read/written per input/output port. In the next slot,  $(s+1)\%S$ , the read blocks are written to their appropriate output ports. Blocks thus propagate in a store-and-forward fashion. However, blocks are small as will be explained in Section 4.3.1. The latency a block incurs per router is equal to the duration of a slot, and bandwidth is guaranteed in multiples of block size per  $S$  slots.

The entries of the slot table map outputs to inputs for every slot;  $T(s, o) = i$ , meaning that blocks from input  $i$  (if present) are passed to output  $o$  at times  $s + kS$ ,  $k \in \mathbb{N}$ . An entry is empty when there is no reservation for that output in that slot. No contention can arise in table  $T$  because there is at most one input per output for each slot. Sending a single input to multiple outputs (multicast) is possible.

Figure 4 illustrates the operation of contention-free routing. It shows a snapshot of a router network with three routers  $R_1$ ,  $R_2$  and  $R_3$  at slot  $s = 2$ , indicated by the three arrows pointing to the third slot in the table (recall that slots are numbered from 0). The size of the slot tables is  $S = 4$ , and only the relevant columns are depicted.

Three connections,  $a$ ,  $b$  and  $c$ , are shown with the gray arrows; the black circles represents packets on the connection with the corresponding letter. Packets  $a$  and  $c$  were switched from the input of the network to their output links in slot 1. In slot 2, shown in Fig. 4, packet  $b$  is switched from input  $i_1$  to output  $o_2$  in router  $R_1$ , as indicated by the slot table  $T_1(2, o_2) = i_1$ . Packets  $a$  and  $c$  are switched similarly by the network.

The slots reserved for a block along its path from source to destination increase by one (modulo  $S$ ). If slot  $s$  is reserved in a router, slot  $(s+1)\%S$  must be reserved in the next router on the path. The assignment of slots to connections in the network is an optimisation problem,

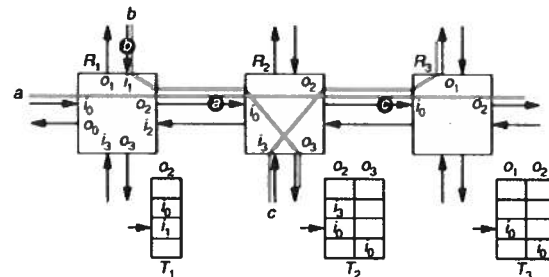


Fig. 4 Contention-free routing

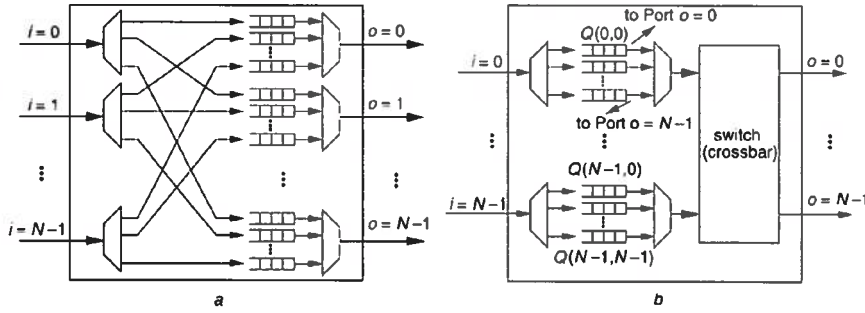


**Fig. 5** Schematics of two router architectures

*a* Output queued architecture  
*b* Virtual-output queued architecture

and is described in Section 4.3.3. Section 4.3.2 explains how slots are reserved in our network.

## 4.2 BE router architecture

Best-effort traffic can have a better *average* performance than offered by guaranteed services. This depends on boundary conditions, such as network load, that are unpredictable. Best-effort services thus fulfil our efficiency requirement, but without offering time-related guarantees. This Section describes an architecture for a best-effort service with uncorrupted, lossless, in-order data transport.

The BE router cost and performance are largely dependent on the contention resolution scheme of the router. The contention resolution scheme has two components; buffering and scheduling. The main trade-off in Section 4.2.1 is between total buffer size, buffering strategy and link utilisation. Without taking global network requirements into account, no decisions will be made, rather we present a router that allows different instances to trade-off hardware complexity for link utilisation at instantiation time. In Section 4.2.2 the trade-off is between link utilisation and schedule complexity and we select an efficient scheduling algorithm that is easily specialised to the different instances.

**4.2.1 Buffering strategy:** The buffering strategy determines the location of buffers inside the router. We distinguish *output queuing* and *input queuing*. In the following,  $N$  is the number of inputs, equal to the number of outputs, of our router. In output queuing  $N^2$  queues are located at the outputs of the router as in Fig. 5a. From the inputs to the outputs there is a fully connected bipartite interconnect to allow every input to write to the corresponding output simultaneously. Output queuing has the best performance among the buffering strategies. However, the interconnect will make the router wire dominated and expensive already for small values of  $N$ .

In input queuing the queues are at the input of the router. A scheduler determines at which times which queues are connected to which output ports, such that no contention occurs. The scheduler derives contention-free connections, a switch matrix (crossbar switch) can be used to implement the connections. In traditional input queuing, or input queuing for short, there is a single queue per input, resulting in a buffer cost of  $N$  queues per router. However, due to the so-called *head-of-line blocking*, for large  $N$ , router utilisation saturates at 59% [16]. Therefore, input queuing results in weak utilisation of the links.

Another version of input queuing is *virtual output queuing* (VOQ) [17]. VOQ combines the advantages of

input queuing and output queuing. It has a switch as in input queuing and has link utilisation close to that of output queuing; 100% link utilisation can still be achieved, when  $N$  is large [18]. As for output queuing, there are  $N^2$  queues. For every input  $i$  there are  $N$  queues  $Q(i, o)$ , one for each output  $o$ , Fig. 5b. Typically the set of  $N$  queues at each input port of a VOQ router are mapped onto a single RAM. However, for NoCs we strive for a small router and, therefore, we require the RAMs to have few addresses. But such RAMs have a large overhead. Therefore, we use dedicated fifos developed in-house, which have almost no overhead, see Section 5.

The decision to select traditional input queuing or VOQ depends on system-level aspects like topology, network utilisation and global wiring cost, and is outside the scope of this paper. In Section 5 we show a prototype of an input-queued router with dedicated hardware fifos and explain that VOQ is a valid option with minor additional cost.

**4.2.2 Matrix scheduling:** The switch matrix, present in input-queued architectures (see Fig. 5), is controlled by a contention resolution algorithm, also known as matrix scheduling, that computes which inputs and outputs must be connected.

The matrix scheduling problem can be modelled as a bipartite graph matching problem. Every input port  $i$  is modelled by a node  $u_i$  and every output port  $o$  by a node  $v_o$ . There is an edge between  $u_i$  and  $v_o$  if and only if queue  $Q(i, o)$  is nonempty. A *match* is a subset of these edges such that every node is incident to, at most, one edge. For example, Fig. 6a is a match of Fig. 6a.

Matching can be done optimally, but because of time complexity and fairness, a nonoptimal algorithm is preferred [19].

Our matching algorithm is iterative and one iteration has three stages, illustrated by an example in Fig. 6 for  $N = 4$ .

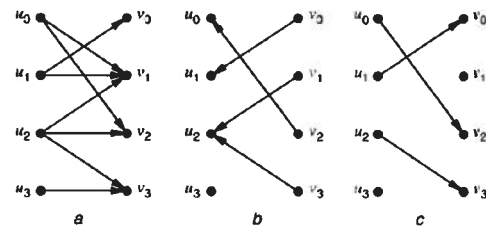


**Fig. 6** Three stages of a schedule iteration

*a* Request  
*b* Grant  
*c* Accept

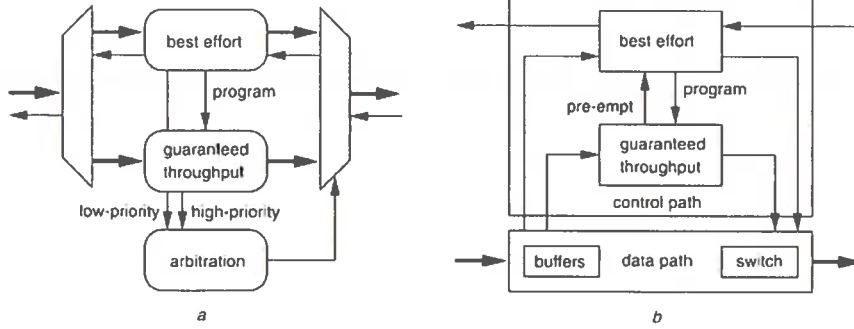


**Fig. 7** Two views of the combined GT-BE router  
a Conceptual view  
b Hardware view

In the first stage, Fig. 6a, every nonempty queue  $Q(i, o)$  requests access to output port  $o$  from input port  $i$ . In the second stage, Fig. 6b, every output port  $o$  grants one request, solving link contention at the output ports. In the third stage, Fig. 6c, every input port  $i$  accepts one grant, to resolve memory contention at the input port. A next iteration then starts with the matching found so far. This scheme is used in various scheduling algorithms, including parallel iterative matching, round robin matching and SLIP [19], and applies to both input queuing and VOQ. For input queuing, however, stage (c) in Fig. 6 is omitted since no contention on input ports can occur. To keep schedule latency as low as possible we use one iteration only.

### 4.3 Combining the GT and BE routers

The GT and BE router architectures are combined to share resources, in particular the links and the switch. Moreover, best-effort traffic enables a packet-based programming model for the guaranteed traffic, as shown in Section 4.3.2.

The principal constraint for a combined router architecture is that guaranteed services are never affected by best-effort services. Figure 7a shows that, conceptually, the combined router contains both router architectures (fat lines represent data, thin lines represent control). Incoming data is switched to either the GT or the BE router. The GT traffic, the traffic that is served by the GT router, has the higher priority to maintain guarantees. This is ensured by the arbitration unit, which therefore affects the best-effort scheduling. Furthermore, best-effort packets can program the guaranteed router, as shown by the arrow-labelled program. Thin lines going from the right to the left indicate network flow control, which is only required for best-effort packets because guaranteed blocks never encounter contention.

Figure 7b shows that the data path, consisting of buffers and switch matrix, is shared and that the control paths of the BE and GT routers are separate, yet interrelated. Moreover, the arbitration unit of Fig. 7a has been absorbed by the BE router. The following Subsection shows how this can be done.

**4.3.1 Arbitration and flit size:** When combining GT and BE traffic in a single network the impact on the network flow control scheme must be taken into account. Recall from Section 3.3 that a BE flit is the smallest unit at which flow control is performed. In other words, the BE scheduling can only react to GT blocks at flit granularity. To avoid alignment problems, the block size ( $B$  words) is a multiple of the flit size ( $F$  words,  $B = \ell F$ ) with  $\ell$  being constant. We prefer a small  $\ell$  to decrease the

store-and-forward delay and reduce the buffer size for guaranteed traffic, and a small  $F$  for fine-grained switching and better statistical multiplexing.

The router architecture contains a data path and a control path, Fig. 7b. The data path maximises throughput for high link utilisation, and the control path maximises the rate of scheduling and switching. They can be designed and optimised independently. Given any combination of their operating frequencies, the router has both maximum throughput and switching rate by using the appropriate flit size  $F_{opt}$ . For  $F > F_{opt}$ , the control path is ready while data is still being transported, lowering the switching rate. For  $F < F_{opt}$ , flits have been transported before the control path finishes, wasting bandwidth. This optimal flit size is defined as  $f_{data}/f_{ctrl}$ , where  $f_{data}$  is the clock frequency of the data path and  $f_{ctrl}$  is the scheduling frequency. Long wires can be pipelined, but due to the scheduler's dependency on flow control this results in a lower scheduling frequency, and thus larger flits.

The combination of GT and BE traffic must be addressed by the matrix scheduling. It does not use priorities, as suggested in [19], because this significantly increases the cost of an iteration. Instead, for every block from input  $i$  to output  $o$ , we remove all edges incident from  $i$  and incident to  $o$  before doing the matrix scheduling. The advantage of this scheme is that the best-effort scheduling adapts to the presence of GT data. The overall scheduling thus effectively adds a single stage to the first iteration of the best-effort scheduling algorithm and, therefore, is a low-cost and effective arbitration scheme.

**4.3.2 Programming model:** In this Section we show how GT connections are set up and torn down by means of special BE packets, called *system packets*, to avoid introducing an additional communication infrastructure only to program the network. To ensure scalability, programming must not require a global view or centralised resources. Section 4.1.2 explains why our contention-free routing uses slot tables; we now see that they are distributed over routers for scalability.

The programming functionality of the router is provided by the *reconfiguration unit*. Because multiple system packets may arrive simultaneously for the reconfiguration unit (i.e. contention), they must be scheduled. This is achieved by viewing the reconfiguration unit as just another router, complete with flow control, which is placed in between the last output and input port of the router, see Fig. 8. In this way contention on the reconfiguration unit is moved to contention on the output port, which is resolved by the matrix scheduling algorithm described before. After

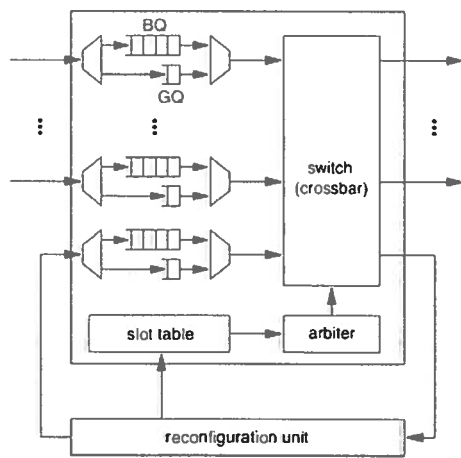


Fig. 8 Architecture view of the combined GT-BE router

programming, the system packets re-enter the router and are routed for the next router to program.

The remainder of this Section explains how system packets program the network. Initially the slot table of every router is empty. There are three system packets; SetUp, TearDown and AckSetUp. They are used to program the slot table in every router on their path. The SetUp packet creates a connection from a source to a destination, and travels in the direction of the data ('downstream'). When a SetUp packet arrives at the destination, it is successful and is acknowledged by returning an AckSetUp. TearDown packets destroy (partial) connections and can travel in either direction. SetUp packets contain the source of the data, the destination or a path to it and a slot number. Every router along the path of the SetUp packet checks if the output to the next router in the path is free in the slot indicated by the packet. If it is free, the output is reserved in that slot and the SetUp packet is forwarded with an incremented (modulo  $S$ ) slot. Otherwise, the SetUp packet is discarded and a TearDown packet returns along the same path. Thus, every path must be reversible; this is the only assumption we make about the network topology. The upstream TearDown packet frees the slot and continues with a decremented slot. Downstream TearDown packets work

similarly and remove existing connections. A connection is successfully opened when an AckSetUp is received, else a TearDown is received. With minor additions, system packets can also be used to program multicast connections.

The programming of the network is illustrated in , where we attempt to set up three connections, shown as light gray arrows. The Figure shows four snapshots of the same network at successive times. Two SetUp packets  $a$  and  $b$  enter the network in Fig. 9a. The number alongside the packet is the slot that is to be programmed in the next router. This is reflected in the slot tables in Fig. 9b, where only the column for output port  $out_3$  is shown. A dark line shows the progress of a connection setup over time. Every snapshot the SetUp packets are routed to their next link and the slot to be programmed is incremented by one. In Fig. 9c, packet  $a$  cannot reserve slot 2 for output port  $out_3$  of the bottom router because it has been reserved for connection  $c$ , and the setup of connection  $a$  fails. The packet  $a$  is changed from SetUp to TearDown and routed back along its path to remove the reservations made so far, Fig. 9d. Note that the slot of packet  $a$  is decremented by one at every router. In Fig. 9d, packet  $a$  has removed the reservation of slot 1 that it made in Fig. 9b. For clarity AckSetUp packets of connections  $b$  and  $c$  are not shown in the Figure.

The programming model is pipelined and concurrent (multiple system packets can be active in the network simultaneously, also from the same source) and distributed (active in multiple routers). Given the distributed nature of the programming model, ensuring consistency and determinism is crucial. The outcome of programming may depend on the execution order of system packets, but is always consistent. The next Section shows how to use this programming model.

**4.3.3 Compile- and run-time slot allocation:** This Section explains how to determine the slots specified in SetUp packets. A slot allocation for a single connection requires that, at every router along the path, the required output is free (not reserved by another connection) in the appropriate slot. Computing an optimal slot allocation for all connections requires a global network view and may be expensive. To reduce computational cost, heuristics can be used, possibly leading to nonoptimal solutions.

SetUp packets of different connections do not fail if connections are set up with conflict-free slots or paths.

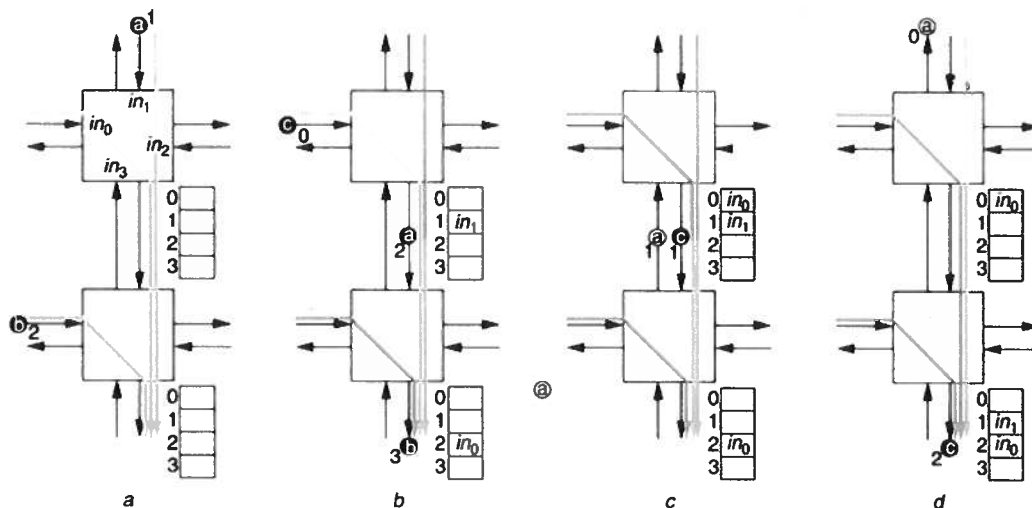


Fig. 9 Setup of connections

All execution orders of SetUp packets then give the same result, so that compile-time slot allocations can be recreated deterministically at run time.

Optimal run-time slot allocation is hard without a global (and central) slot table view, which is nonscalable and slows down programming. Distributed run-time slot allocation is scalable, but lacks a global view and may, therefore, be suboptimal. Moreover, SetUp packets may interfere, making programming more involved and perhaps adaptive (in the sense of depending on the programming actions of other NoC users). However, dynamic connection management at high rates will require distributed slot allocation. In a simple distributed greedy algorithm, all sources repeatedly generate random slot numbers for each setup until their connection succeeds. We conclude that our programming model allows both compile-time and run-time slot allocation. Computational complexity, deterministic results and scalability can be balanced according to system requirements.

## 5 Current results and future work

The previous Section shows a prototype combined GT-BE architecture. We have synthesised an input-queued router using wormhole routing with arity 5, a queue depth of 8 flits of 3 words of 32 bits and 256 slots in a 0.13  $\mu\text{m}$  technology. The layout is shown in Fig. 10, where the router and reconfiguration unit are shown separately (cf. Fig. 8). It has an aggregate bandwidth of  $5 \times 500 \text{ MHz} \times 32 \text{ bit} = 80 \text{ Gbit/s}$ . The area of the router is  $0.26 \text{ mm}^2$ .

The area of  $0.26 \text{ mm}^2$  depends on the use of dedicated hardware fifos, labelled GQ and BQ in Fig. 10. The router would have been at least three times larger with register-based or RAM-based fifos. The RAMs required for input queuing and VOQ in an on-chip router have few addresses, so that their overhead makes them as large as (area-inefficient) register files. Decreasing the queue depths reduces the buffering area (with registers at least), but also degrades the router performance.

Dedicated hardware fifos enable both input and virtual output queuing strategies using wormhole routing because of the reasonable buffering cost. For example, VOQ with two-flit deep fifos is only moderately larger than the input queuing with fifos of depth 8 of Fig. 10. Virtual cut-through routing in combination with input queuing is also affordable now, because for packets of at most 8 flits, it has the same cost as the prototype.

The slot table (labelled STU in Fig. 10) occupies a significant part of the router, for two reasons. Logically the slot table is very large (256 slots). It is not worthwhile to reduce the number of slots because the RAM is very area-inefficient. We are investigating more advanced slot table schemes and new memory architectures to reduce the size and area of the slot table. The cost of offering time-related guaranteed services is then lower.

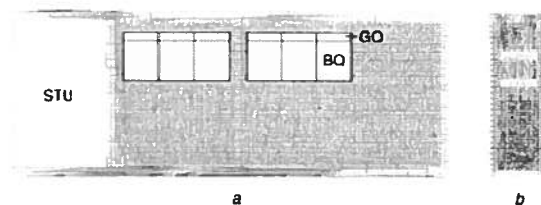


**Fig. 10** Layout

a Combined GT-BE router  
b Reconfiguration unit (RCU)

We separately synthesised the data and control paths (cf. ) with arities in the range 3–13 to verify their speeds. With increasing arity, the speed of the data path reduces little. The speed of the control path decreases by a factor of two, corresponding to the complexity increase of the scheduling. For each arity, we balance the performance of the data and control paths by adjusting the flit size as needed, as shown in Section 4.3.1. The data and scheduling frequencies of the prototype router are 500 MHz and 166 MHz, respectively, with a flit size of 3 ( $F_{opt} = 500/166$ , cf. Section 4.3.1).

Our results show that the cost and performance of the combined GT-BE router can make it the basis of a router-based network on chip. It further shows that dedicated hardware fifos significantly reduce the buffering area and so enable both input queuing and VOQ, with wormhole and virtual cut-through routing.

## 6 Conclusions

In this paper we show that guaranteed services are essential to provide predictable interconnects that enable compositional system design and integration. However, guarantees typically utilise resources inefficiently. Best-effort services overcome this problem, but provide no guarantees. So, integrating guaranteed and best-effort services allows efficient resource utilisation, while still providing guarantees for critical traffic.

Time-related guarantees, such as throughput and latency, can only be constructed on a NoC that intrinsically has these properties. We therefore define a router-based NoC architecture that combines guaranteed and best-effort services. The router architecture has conceptually two parts; the guaranteed-throughput (GT) and best-effort (BE) routers. Both offer data integrity, lossless data delivery and in-order data delivery. Additionally, the GT router offers guaranteed throughput and latency services using pipelined circuit switching with time division multiplexing. The BE router uses packet switching, virtual cut-through or wormhole routing, and input queuing or virtual-output queuing.

We combine the GT and BE router architectures efficiently by sharing router resources. The guarantees are never affected by the BE traffic, and links are efficiently utilised because BE traffic uses all bandwidth left over by GT traffic. Connections are programmed using BE packets. The programming model is robust, concurrent and distributed. It enables run-time and compile-time, deterministic and adaptive connection management.

For all our architecture choices, we show the trade-offs between hardware complexity and efficiency. Our choices are motivated by a prototype router which has an area of  $0.26 \text{ mm}^2$  in a 0.13 micron technology and offers 80 Gbit/s aggregate throughput. We use dedicated hardware fifos to significantly reduce the area of the data queues. With RAM-based or register-based queues, the router area would have been at least three times larger.

Dedicated hardware fifos enable

- (a) input queuing using both wormhole and virtual cut-through routing,
- (b) virtual-output queuing using wormhole routing.

The buffer costs are too high, however, for virtual-output queuing with virtual cut-through routing.

The cost of offering time-related guaranteed services is still high for our router. We are investigating how to reduce this cost.

An attractive feature of our router architecture is the ability to combine separately optimised data and control paths by adjusting the flit size.

In conclusion, we describe and motivate a choice of architectures for routers, which are an essential component in a NoC. They fulfil our NoC requirements by providing guaranteed services and satisfy the efficiency constraint by offering best-effort services.

## 7 References

- 1 Keutzer, K., Malik, S., Richard Newton, A., Rabaey, J.M., and Sangiovanni-Vincentelli, A.: 'System-level design: orthogonalization of concerns and platform-based design', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Sys.*, 2000, **19**, (12), pp. 1523–1543
- 2 Sgroi, M., Sheets, M., Mihal, A., Keutzer, K., Malik, S., Rabaey, J., and Sangiovanni-Vincentelli, A.: 'Addressing the system-on-a-chip interconnect woes through communication-based design'. Proc. Design Automation Conf., Las Vegas, NV, June 2001, pp. 667–672
- 3 Bainbridge, J., and Furber, S.: 'CHAIN: A delay-insensitive chip area interconnect', *IEEE Micro*, 2002, **22**, (5), pp. 16–23
- 4 Benini, L., and De Micheli, G.: 'Powering networks on chips'. Proc. Int. Symp. on Systems Synthesis (ISSS), October 2001, pp. 33–38
- 5 Benini, L., and De Micheli, G.: 'Networks on chips: A new SoC paradigm', *IEEE Comput.*, 2002, **35**, (1), pp. 70–80
- 6 Dally, W.J., and Towels, B.: 'Route packets, not wires: on-chip interconnection networks'. Proc. Design Automation Conf., June 2001, pp. 684–689
- 7 Goossens, K., van Meerbergen, J., Peeters, A., and Wielage, P.: 'Networks on silicon: Combining best-effort and guaranteed services'. Proc. Design Automation and Test in Europe Conf., Paris, March 2002, pp. 423–425
- 8 Zimmer, H.: Fault modelling and error-control coding in a network-on-chip. Master's thesis, Laboratory of Electronics and Computer Systems, Royal Institute of Technology (KTH), IMIT/LECS 2002-26, Stockholm, 2002
- 9 Guerrier, P., and Greiner, A.: 'A generic architecture for on-chip packet-switched interconnections'. Proc. Design Automation and Test in Europe Conf., Paris, 2000, pp. 250–256
- 10 Rose, M.T.: 'The open book: a practical perspective on OSI' (Prentice-Hall, Englewood Cliffs, NJ, 1990)
- 11 Tanenbaum, A.S.: 'Computer networks' (Prentice-Hall, NJ, 1996)
- 12 Rădulescu, A., and Goossens, K.: 'Communication services for networks on silicon' in Bhattacharyya, S., Deprettere, E., and Teich, J. (Eds.): 'Domain-specific processors: systems, architectures, modeling and simulation' (Marcel Dekker, New York, 2003), pp. 275–299, to be published
- 13 Goossens, K., Dielissen, J., van Meerbergen, J., Poplavko, P., Rădulescu, A., Rijpkema, E., Waterlander, E., and Wielage, P.: 'Guaranteeing the quality of services in networks on chip' in Jantsch, A., and Tenhunen, H. (Eds.): 'Networks on chip' (Kluwer, Dordrecht, 2003), pp. 61–82
- 14 Zhang, H.: 'Service disciplines for guaranteed performance service in packet-switching networks', *Proc. IEEE*, 1995, **83**, (10), pp. 1374–1396
- 15 Rexford, J.: 'Tailoring router architectures to performance requirements in cut-through networks'. PhD thesis, University of Michigan, Department of Computer Science and Engineering, 1999
- 16 DeHon, A.: 'Robust, high-speed network design for large-scale multiprocessing'. A.I. Technical report 1-445, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, September 1993
- 17 Karol, M.J., Hluchyj, M.G., and Morgan, S.P.: 'Input versus output queueing on a space-division packet switch', *IEEE Trans. Commun.*, 1987, **35**, (12), pp. 1347–1356
- 18 Ali, M.K.M., and Youssefi, M.: 'The performance analysis of an input access scheme in a high-speed packet switch'. Proc. Joint Conf. IEEE Computer and Communications Societies, Networking in the 90s., (INFOCOM), Bal Harbour, FL, April 1991, vol. 2, pp. 454–461
- 19 McKeown, W.N.: 'Scheduling algorithms for input-queued cell switches'. PhD thesis, University of California, Berkeley, CA, USA, 1995

# Exhibit D

# Trade Offs in the Design of a Router with Both Guaranteed and Best-Effort Services for Networks on Chip (Extended Version)

E. Rijpkema, K. Goossens, A. Rădulescu,  
J. Dielissen, J. van Meerbergen, P. Wielage, and E. Waterlander  
Philips Research Laboratories, Eindhoven, The Netherlands  
E-mail: edwin.rijpkema@philips.com

September 12, 2003

## Abstract

Managing the complexity of designing chips containing billions of transistors requires decoupling computation from communication. For the communication, scalable and compositional interconnects, such as networks on chip (NoC), must be used. In this paper we show that guaranteed services are essential in achieving this decoupling. Guarantees typically come at the cost of lower resource utilization. To avoid this, they must be used in combination with best-effort services. The key element of our NoC is a router consisting conceptually of two parts: the so-called guaranteed throughput (GT) and best-effort (BE) routers. We combine the GT and BE router architectures in an efficient implementation by sharing resources. We show the trade offs between hardware complexity and efficiency of the combined router, and motivate our choices. Our reasoning for the trade offs is validated with a prototype router implementation. We show a layout of an input-queued wormhole  $5 \times 5$  router with an aggregate bandwidth of 80 Gbit/s. It occupies  $0.26 \text{ mm}^2$  in CMOS12. This shows that our router provides high performance at reasonable cost, bringing NoCs one step closer.

## 1 Introduction

Recent advances in technology raise the challenge of managing the complexity of designing chips containing billions of transistors. A key ingredient in tackling this challenge is *decoupling the computation from communication* [11, 16]. This decoupling allows IPs (the computation part), and the interconnect (the communication part) to be designed independently from each other.

In this paper, we focus on the communication part. Existing interconnects (e.g., buses) may no longer be feasible for chips with many IPs, because of the diverse and dynamic communication requirements. *Networks on a chip* (NoC) are emerging as an alternative to existing on-chip interconnects because they (a) structure and manage global wires in new deep-submicron technologies [2–5, 7], (b) share wires, lowering their number and increasing their utilization [5, 7], (c) can be energy efficient and reliable [3], and (d) are scalable when compared to traditional buses [9].

Decoupling the computation from communication requires that the *services* that IPs use to communicate are well-defined, and hide the implementation details of the interconnect [11], Figure 1(a). NoCs help, because they are traditionally designed using layered protocol stacks [14], where each layer provides a well-defined interface which decouples service usage from service implementation [4, 16], see Figure 1(b).

In particular, *guaranteed services* are essential because they make the requirements on the NoC explicit, and limit the possible interactions of IPs with the communication environment. IPs can also be designed independently, because their use of guaranteed services is not affected by the interconnect or by other IPs. This is essential for a compositional construction (design and programming) of systems on chip (SoC). Moreover, failures are restricted to the IP configuration phase (a service request is either granted or denied by the NoC) which simplifies the IP programming model [7]. We view the guaranteed services to be offered by an interconnect as a requirement from the applications, see Figure 1(c).

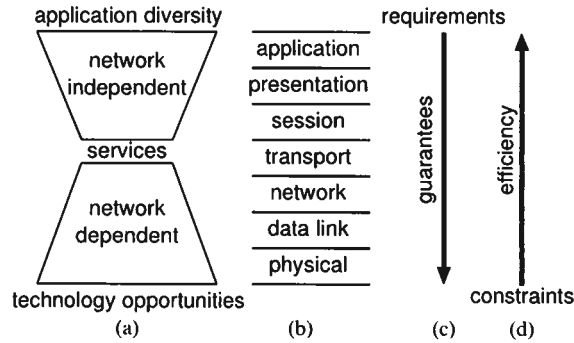


Figure 1: Network services (a) hide the interconnect details and allow reusable components to be built on top of them, (b) are built using a layered approach (c) are driven by the application requirements, and (d) their efficiency relies on technology and network organization.

The drawback of using guaranteed services is that they require resource reservations for worst-case scenarios. This is not acceptable in a SoC where cost constraints are typically very tight, see Figure 1(d). Therefore, we also provide *best-effort services* to exploit the network capacity that is left over, or reserved but unused. Guaranteed services are then used for the critical (e.g. real-time) traffic, and best-effort services for non-critical communication.

The combination of guaranteed and best-effort classes is known from general computer network research [17], but not for on-chip networks. As on- and off-chip networks have different characteristics, the trade offs in their design are different. In this paper, we present the trade offs between hardware complexity and efficiency for networks on chip, and motivate our choices.

We present a prototype router architecture that reflects one particular set of design choices. It has an aggregate bandwidth of 80 Gbit/s, and its CMOS12 lay-out occupies 0.26 mm<sup>2</sup>. We list other feasible variations that either increase performance, or lower the router cost.

In this paper, we first list a set of network-independent communication services that are essential in chip design (Section 2). Then, we show the trade-offs between efficiency and cost that we make in our NoC. In Section 3, we present some general network-related issues that are used in the sections to follow. In Section 4, we zoom into the internals of the key component of our NoC: a router that efficiently provides both guaranteed and best-effort services. In Section 5, we demonstrate the feasibility of our router design through a prototype implementation in CMOS12.

## 2 Services

The NoC services that we consider essential for chip design are [15]:

- *data integrity*, meaning that data is delivered uncorrupted,
- *lossless data delivery*, which means no data is dropped in the interconnect,
- *in-order data delivery*, which specifies that the order in which data is delivered is the same order in which it has been sent, and
- *throughput* and *latency* services that offer time-related bounds.

As shown in Section 1, guaranteed services are essential to simplify IP design and integration. With the current technology, we assume data integrity is solved at the data-link layer. All the other services can be guaranteed or not on request. In the next section, we describe briefly how these services are provided by our NoC, and in Section 4 we describe how our router architecture enables an efficient implementation of these services.

*Guaranteed services* require resource reservation for worst-case scenarios, which can be expensive. For example, guaranteeing throughput for a stream of data implies reserving bandwidth for its peak throughput, even when its average is much lower. As a consequence, when using guarantees, resources are often underutilized.

*Best-effort services* do not reserve any resources, and hence provide no guarantees. Best-effort services use resources well because they are typically designed for average-case scenarios instead of worst-case scenarios. They are also easy and fast to use, as they require no resource reservation. Their main disadvantage is their unpredictability: one cannot rely on a given performance (i.e., they

do not offer guarantees). In the best case, if certain boundary conditions are assumed, a statistical performance can be derived.

The requirements for guaranteed services and the efficiency constraint (i.e., good resource utilization) seem conflicting. Our approach to a predictable and low-cost interconnect is to integrate the guaranteed and best-effort services in the same interconnect. Guaranteed services would be used for critical traffic, and best-effort services for non-critical traffic. In this way the best-effort services can exploit the available resources left over by the guaranteed services. This is illustrated further in Figure 2 and described in more detail in [8].

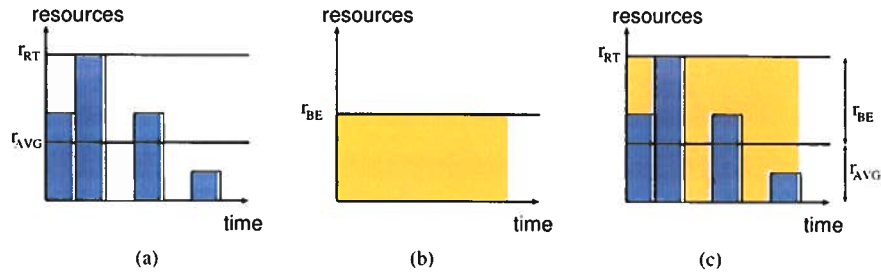


Figure 2: (a) Guaranteed traffic requires worst-case ( $r_{RT}$ ) resource allocation (here: bandwidth). The best-effort traffic of (b) consumes the unused bandwidth of (a), resulting in (c).

The figure shows that the network must be dimensioned to meet worst-case bandwidth requirements of guaranteed traffic, see Figure 2(a). All bandwidth that is not used, the white space in Figure 2(a), can be used for the best-effort traffic in (b). The best-effort traffic is delivered in bounded time when its volume is less than the white space in Figure 2(a).

For example a video processing IP will typically require a lossless, in-order video stream with guaranteed throughput, but possibly allows corrupted samples. Another example is cache updates which require uncorrupted, lossless, low-latency data transfer, but ordering and guaranteed throughput are less important. In Section 4.3 we show how integrated guaranteed and best-effort services efficiently can use common resources. In the remainder of this section we analyze the minimum level of abstraction at which the communication services must be offered to hide the network internals.

Traditionally, network services have been implemented and offered using a layered protocol stack, typically aligned to the ISO-OSI reference model [14], see Figure 1(b). NoCs also take this approach [3, 4, 7, 16], because it structures and decomposes the service implementation, and the protocol stack concepts aid positioning of services.

To achieve the decoupling of computation from communication, the communication services must be offered at least at the level of the transport layer in OSI reference model. It is the first layer that

offers end-to-end services, hiding the network details; see Figure 1(a, b) [4].

The lowest three layers in the protocol stack, namely physical, data-link, and network layers, are network specific. Therefore, these services should not be visible to the IPs when decoupling between computation from communication is desired. However, these layers are essential in implementing the services, because constructing guarantees without guarantees at the layer below is either very expensive, or even impossible. For example, implementing a lossless communication on top of a lossy service requires acknowledgment, data retransmission, and filtering duplicated data. This leads to an increase in traffic, and possibly larger buffer space requirements. Even worse, providing guarantees for time-related services is impossible if lower layers do not offer these guarantees. For example, latency can not be guaranteed if communication at a lower layer is lossy. As a consequence, guarantees can only be built *on top* of guarantees, see Figure 1(c). Similarly, a layer's efficiency is based on efficient implementations of the layers below it, see Figure 1(d).

### 3 Networks on chip

General computer network research is a mature research field [17] which has many issues in common with NoCs. However, two significant differences between computer networks and on-chip networks make the trade offs in their design very different [5]. First, routers of a NoC are more resource constrained than those in computer network, in particular in the control complexity and in the amount of memory. Second, communication links of a NoC are relatively shorter than those in computer networks, allowing tight synchronization between routers.

To place the work described in this paper in context see Figure 3. The services that are described

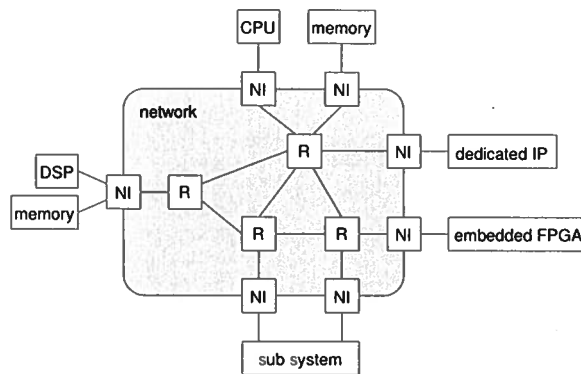


Figure 3: A SoC composed of heterogeneous IP together with a NoC. The NoC comprises interconnected routers (R), and network interfaces (NI) that connect the IP to the routers.

in the previous section are provided by the network interfaces. It is these services that are visible to

the users (IP) of the network.

The scope of this paper is limited to the router network and we identify three important issues in its design. These are: the *switching mode*, *contention resolution*, and *network flow control*. Equally important are *end-to-end flow control* and *congestion control*, but these are handled by the network interfaces and hence are out of the scope of this paper. Moreover, we assume guaranteed data integrity at the link level and retain it at the network layer and higher.

### 3.1 Switching mode

The *switching mode* of a network specifies how data and control are related. We distinguish *circuit switching* and *packet switching*. In circuit switching data and control are separated. The control is provided to the network to *set up a connection*. This results in a *circuit* over which all subsequent data of the connection is transported. In *time-division circuit switching* bandwidth is shared by time-division multiplexing connections over circuits. Circuit-switched networks inherently offer time-related guaranteed services after resources are reserved during the connection set up.

In *packet switching* data is divided into *packets* and every packet is composed of a control part, the *header*, and a data part, the *payload*. Network routers inspect, and possibly modify, the headers of incoming packets to switch the packet to the appropriate output port. Since in packet switching the packets are self contained, there is no need for a set-up phase to allocate resources. Therefore, best-effort services are naturally provided by packet switching.

### 3.2 Contention resolution

When a router attempts to send multiple data items over the same link at the same time *contention* is said to occur. As only one data item can be sent over a link at any point in time, a selection among the contending data must be made; this process is called contention resolution.

In circuit switching, contention resolution takes place at set up at the granularity of connections, so that data sent over different connections do not conflict. Thus, there is no contention during data transport, and time-related guarantees can be given.

In packet switching contention resolution takes place at the granularity of individual packets. Because packet arrival cannot be predicted contention can not be avoided. It is resolved dynamically by scheduling which data items are sent in turn. This requires data storage in the router, see Section 4.2.1, and delays the data in an unpredictable manner which complicates the provision of guarantees, see Section 4.1.1.

### 3.3 Network flow control

*Network flow control*, also called *routing mode*, addresses the limited amount of buffering in routers and data acceptance between routers. In circuit switching connections are set up. The data sent over these connections is always accepted by the routers and hence no network flow control is needed. In packet switching, data items must be buffered at every router before they are sent on. Because routers have a limited amount of buffering they accept data only when they have enough space to store the incoming data.

There are three types of network flow control, namely *store-and-forward*, *virtual cut-through*, and *wormhole* routing. In store-and-forward routing, an incoming packet is received and stored in its entirety before it is forwarded to the next router. This requires storage for the complete packet, and implies a per-router latency of at least the time required for the router to receive the packet.

In virtual cut-through routing a packet is forwarded as soon as the next router guarantees that the complete packet will be accepted. When no guarantee is given, the router must be able to store the whole packet. Thus, virtual cut-through routing requires buffer space for a complete packet, like store-and-forward routing, but allows lower-latency communication.

In wormhole routing packets are split in so-called *flits* (flow control digits). A flit is passed to the next router when the flit can be accepted, even when there is not enough buffer space for the complete packet. As soon as a flit of a packet is sent over an output port, that output port is reserved for flits of that packet only. When the first flit of a packet is blocked the trailing flits can therefore be spread over multiple routers, blocking the intermediate links. Wormhole routing requires the least buffering (buffer flits instead of packets) and also allows low-latency communication. However, it is more sensitive to deadlock and generally results in lower link utilization than virtual cut-through routing.

To allow low latency we consider both virtual-cut through and wormhole routing, which are both feasible in terms of buffer area, as shown in Section 5.

## 4 A combined GT-BE router

Section 2 defines our requirements for NoCs in terms of services that are to be offered, in particular, both guaranteed and best-effort services. Using the general network issues of the previous section we show in the following two subsections that the guaranteed and best-effort services can conceptually be described by two independent router architectures. The combination of these two router architectures is efficient and has a flexible programming model, as described in Section 4.3. Section 5 then shows

a prototype implementation.

## 4.1 A GT router architecture

Our guaranteed-throughput (GT) router guarantees uncorrupted, lossless, and ordered data transfer, and both latency and throughput over a finite time interval. As mentioned earlier, data integrity is solved at the data-link layer; we do not address it further. The GT router is lossless because we use a variant of circuit switching, described in the next section. Data is transported in fixed-size blocks. As only one block is stored per input in the GT router, data items remain ordered per connection. We now turn to the more challenging time-related guarantees, namely throughput and latency.

### 4.1.1 Time-related guarantees

Latency is defined as the duration a packet is transported over the network. Guaranteeing latency, therefore, means that a worst-case upper bound must be given for this time. We define throughput for a given producer-consumer pair as the amount of data transported by the network over a finite, fixed time interval. Guaranteeing throughput means giving a lower bound.

We observe that guaranteeing latency even in a lossless router is difficult because contention requires scheduling and hence cause delays. Guaranteeing throughput is less problematic. Rate-based packet switching (for an overview see [18]) offers guaranteed throughput over a finite period, and hence a latency bound. This bound is very high, however, and the cost of buffering is also high. Deadline-based packet switching [13] offers preferential treatment for packets close to their deadline. This allows differential latency guarantees (under certain admissible traffic assumptions), but also at high buffer costs.

Circuit switching solves the contention at set up, so naturally providing guaranteed latency and throughput. Circuits can be pipelined to improve throughput [6], at the cost of additional buffering and latency. Time-division multiplexing connections over pipelined circuits additionally offers flexibility in bandwidth allocation. This requires a logical notion of router synchronicity, which is possible because a NoC is better controllable than a general network. We explain this variation in more detail in the next subsection. The associated programming model is described in Section 4.3.2.

### 4.1.2 Contention-free routing

A router uses a *slot table* to (a) avoid contention on a link, (b) divide up bandwidth per link between connections, and (c) switch data to the correct output. Every slot table  $T$  has  $S$  time slots (rows), and  $N$  router outputs (columns). There is a logical notion of synchronicity: all routers in the network

are in the same fixed-duration slot. In a slot  $s$  at most one *block* of data can be read/written per input/output port. In the next slot,  $(s + 1)\%S$ , the read blocks are written to their appropriate output ports. Blocks thus propagate in a store and forward fashion. The latency a block incurs per router is equal to the duration of a slot and bandwidth is guaranteed in multiples of block size per  $S$  slots.

The entries of the slot table map outputs to inputs for every slot:  $T(s, o) = i$ , meaning that blocks from input  $i$  (if present) are passed to output  $o$  at times  $s + kS, k \in \mathbb{N}$ . An entry is empty when there is no reservation for that output in that slot. No contention can arise in table  $T$  because there is at most one input per output for each slot. Sending a single input to multiple outputs (multicast) is possible.

Figure 4 illustrates the operation of contention-free routing. It shows a snapshot of a router network with three routers  $R_1$ ,  $R_2$ , and  $R_3$  at slot  $s = 2$ , indicated by the arrows pointing to the third slot in the table (recall that slots are numbered from 0). The size of the slot tables is  $S = 4$ , and only the relevant columns are depicted.

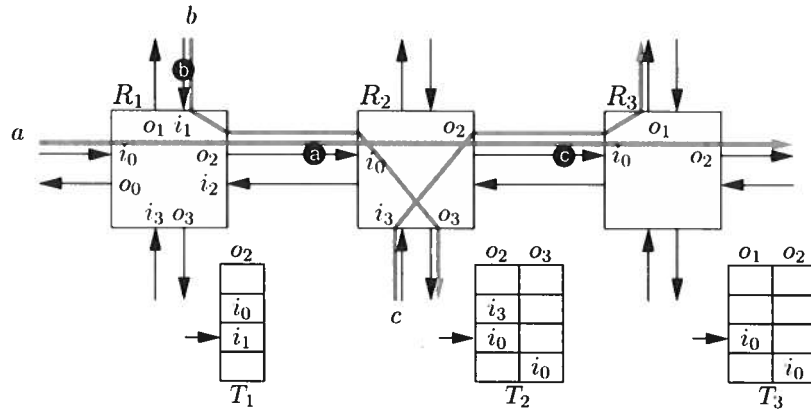


Figure 4: Contention-free routing.

Three connections,  $a$ ,  $b$ , and  $c$ , are shown with the gray arrows; the black circles represents packets on the connection with the corresponding letter. Packets  $a$  and  $c$  were switched from the input of the network to their output links in slot 1. In slot 2, shown in Figure 4, packet  $b$  is switched from input  $i_1$  to output  $o_2$  in router  $R_1$ , as indicated by the slot table  $T_1(2, o_2) = i_1$ . Packets  $a$  and  $c$  are switched similarly by the network.

The slots reserved for a block along its path from source to destination increase by one (modulo  $S$ ). If slot  $s$  is reserved in a router, slot  $(s + 1)\%S$  must be reserved in the next router on the path. The assignment of slots to connections in the network is an optimization problem, and is described in Section 4.3.3. Section 4.3.2 explains how slots are reserved in our network.

## 4.2 A BE router architecture

Best-effort traffic can have a better *average* performance than offered by guaranteed services. This depends on boundary conditions, such as network load, that are unpredictable. Best-effort services thus fulfill our efficiency requirement, but without offering time-related guarantees. This section describes an architecture for a best-effort service with uncorrupted, lossless, in-order data transport.

The BE router cost and performance are largely dependent on the contention resolution scheme of the router. The contention resolution scheme has two components: buffering and scheduling. The main trade off in Section 4.2.1 is between total buffer size, buffering strategy, and link utilization. Without taking global network requirements into account, no decisions will be made, rather we present a router that allows different instances to trade off hardware complexity for link utilization at instantiation time. In Section 4.2.2 the trade off is between link utilization and schedule complexity and we select an efficient scheduling algorithm that is easily specialized to the different instances.

### 4.2.1 Buffering strategy

The buffering strategy determines the location of buffers inside the router. We distinguish *output queuing* and *input queuing*. In the following,  $N$  is the number of inputs, equal to the number of outputs, of our router. In output queuing  $N^2$  queues are located at the outputs of the router as in Figure 5(a). From the inputs to the outputs there is a fully connected bipartite interconnect to allow every input to write to the corresponding output simultaneously. Output queuing has the best performance among the buffering strategies, however, the interconnect will make the router wire dominated and expensive already for small values of  $N$ .

In input queuing the queues are at the input of the router. A scheduler determines at which times which queues are connected to which output ports such that no contention occurs. The scheduler derives contention-free connections, a switch matrix (crossbar switch) can be used to implement the connections. In traditional input queuing, or input queuing for short, there is a single queue per input, resulting in a buffer cost of  $N$  queues per router. However, due to the so-called *head-of-line blocking*, for large  $N$ , router utilization saturates at 59% [10]. Therefore, input queuing results in weak utilization of the links.

Another version of input queuing is *virtual output queuing* (VOQ) [1]. VOQ combines the advantages of input queuing and output queuing. It has a switch like in input queuing and has the link utilization close to that of output queuing; 100% link utilization can still be achieved, when  $N$  is large [12]. As for output queuing, there are  $N^2$  queues. For every input  $i$  there are  $N$  queues  $Q(i, o)$ ,

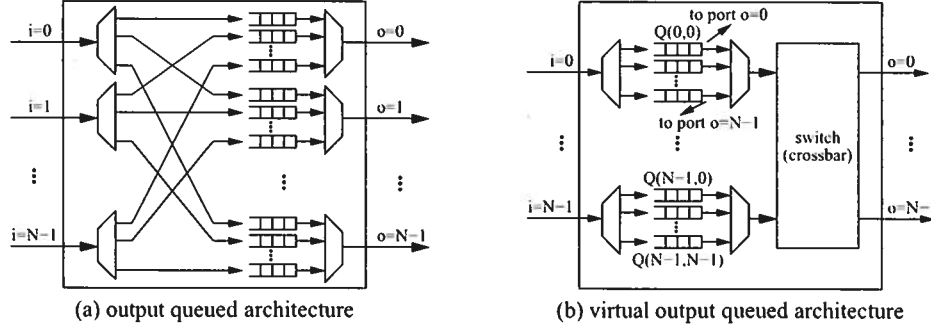


Figure 5: Schematics of two router architectures.

router are mapped onto a single RAM. However, for NoCs we strive at a small router and therefore we require the RAMs to have few addresses. But such RAMs have large overhead. Therefore, we use dedicated fifos developed in-house, which have almost no overhead, see Section 5.

The decision to select traditional input queuing or VOQ depends on system-level aspects like topology, network utilization, and global wiring cost, and is outside the scope of this paper. In Section 5 we show a prototype of an input queued router with dedicated hardware fifos and explain that VOQ is a valid option with minor additional cost.

#### 4.2.2 Matrix scheduling

The switch matrix, present in input-queued architectures (see Figure 5), is controlled by a contention resolution algorithm, also known as matrix scheduling, that computes which inputs and outputs must be connected.

The matrix scheduling problem can be modeled as a bipartite graph matching problem. Every input port  $i$  is modeled by a node  $u_i$  and every output port  $o$  by a node  $v_o$ . There is an edge between  $u_i$  and  $v_o$  if and only if queue  $Q(i, o)$  is non-empty. A *match* is a subset of these edges such that every node is incident to at most one edge. For example, Figure 6(c) is a match of Figure 6(a).

Matching can be done optimally, but because of time complexity and fairness, a non-optimal algorithm is preferred [12].

Our matching algorithm is iterative and one iteration has three stages, illustrated by an example in Figure 6 for  $N = 4$ . In the first stage, see Figure 6(a), every non-empty queue  $Q(i, o)$  *requests* access to output port  $o$  from input port  $i$ . In the second stage, see Figure 6(b), every output port  $o$  *grants* one request, solving link contention at the output ports. In the third stage, see Figure 6(c), every input port  $i$  *accepts* one grant, to resolve memory contention at the input port. A next iteration then starts with

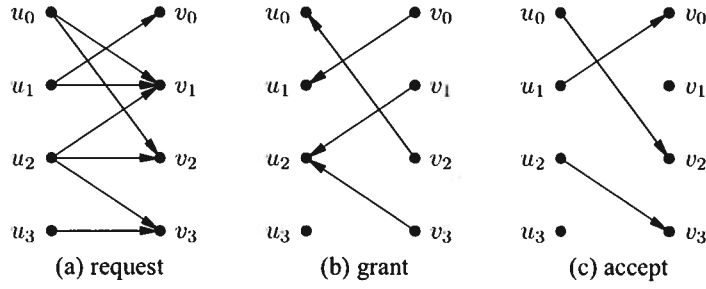


Figure 6: The three stages of a schedule iteration.

the matching found so far. This scheme is used in various scheduling algorithms, including parallel iterative matching, round robin matching, and SLIP [12], and applies to both input queuing and VOQ. For input queuing, however, stage (c) in Figure 6 is omitted since no contention on input ports can occur. To keep schedule latency as low as possible we use one iteration only.

### 4.3 Combining the GT and BE routers

The GT and BE router architectures are combined to share resources, in particular the links and the switch. Moreover, best-effort traffic enables a packet-based programming model for the guaranteed traffic, as shown later, in Section 4.3.2.

The principal constraint for a combined router architecture is that guaranteed services are never affected by best-effort services. Figure 7(a) shows that, conceptually, the combined router contains both router architectures (fat lines represent data, thin lines represent control). Incoming data is switched to either the GT or the BE router. The GT traffic, the traffic that is served by the GT router, has the higher priority, to maintain guarantees. This is ensured by the arbitration unit, which therefore affects the best-effort scheduling. Furthermore, best-effort packets can program the guaranteed router, as shown by the arrow labeled program. Thin lines going from the right to the left indicate network flow control, which is only required for best-effort packets because guaranteed blocks never encounter contention.

Figure 7(b) shows that the data path, consisting of buffers and switch matrix, is shared, and that the control paths of the BE and GT routers are separate, yet interrelated. Moreover, the arbitration unit of Figure 7(a) has been absorbed by the BE router. The following subsection shows how this can be done.

#### 4.3.1 Arbitration and flit size

When combining GT and BE traffic in a single network the impact on the network flow control scheme must be taken into account. Recall from Section 3.3 that a BE flit is the smallest unit at which flow

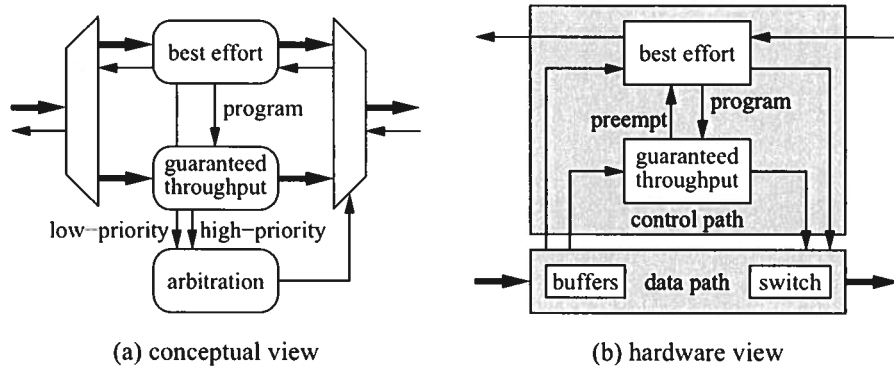


Figure 7: Two views of the combined GT-BE router.

control is performed. In other words, the BE scheduling can only react to GT blocks at flit granularity. To avoid alignment problems, the block size ( $B$  words) is a multiple of the flits ( $F$  words,  $B = \ell F$ ) with  $\ell$  being constant. We prefer a small  $\ell$  to decrease the store-and-forward delay and reduce the buffer size for guaranteed traffic, and a small  $F$  for fine-grained switching and better statistical multiplexing.

The router architecture contains a data path and a control path, see Figure 7(b). The data path maximizes throughput for high link utilization, and the control path maximizes the rate of scheduling and switching. They can be designed and optimized independently. Given any combination of their operating frequencies, the router has both maximum throughput and switching rate by using the appropriate flit size  $F_{opt}$ . For  $F > F_{opt}$ , the control path is ready while data is still being transported, lowering the switching rate. For  $F < F_{opt}$ , flits have been transported before the control path finishes, wasting bandwidth. This *optimal* flit size is defined as  $f_{data}/f_{ctrl}$ , where  $f_{data}$  is the clock frequency of the data path and  $f_{ctrl}$  is the scheduling frequency. Long wires can be pipelined, but due to the scheduler's dependency on flow control, this results in a lower scheduling frequency, and thus larger flits.

The combination of GT and BE traffic must be addressed by the matrix scheduling. It does not use priorities, as suggested in [12], because this significantly increases the cost of an iteration. Instead, for every block from input  $i$  to output  $o$  we remove all edges incident from  $i$  and incident to  $o$ , before doing the matrix scheduling. The advantage of this scheme is that the best-effort scheduling adapts to the presence of GT data. The overall scheduling thus effectively adds a single stage to the first iteration of the best-effort scheduling algorithm and therefore is a low-cost and effective arbitration scheme.

### 4.3.2 Programming model

In this section we show how GT connections are set up and torn down by means of special BE packets, called *system packets*, to avoid introducing an additional communication infrastructure only to program the network. To ensure scalability, programming must not require a global view or centralized resources. Section 4.1.2 explains why our contention-free routing uses slot tables; we now see that they are distributed over routers for scalability.

The programming functionality of the router is provided by the *reconfiguration unit*. Because multiple system packets may arrive simultaneously for the reconfiguration unit (contention) they must be scheduled. This is achieved by viewing the reconfiguration unit as yet another router, complete with flow control, which is placed in between the last output and input port of the router, see Figure 8. In this way contention on the reconfiguration unit is moved to contention on the output port, which is resolved by the matrix scheduling algorithm described before. After programming, the system packets re-enter the router and are routed to the next router to program it.

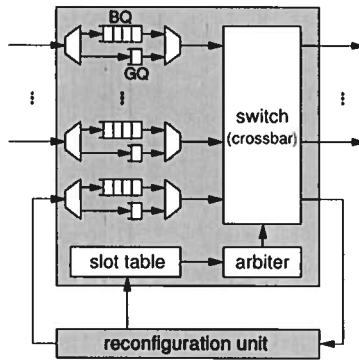


Figure 8: Architecture view of the combined GT-BE router.

Initially the slot table of every router is empty. There are three system packets: *SetUp*, *TearDown*, and *AckSetUp*. They are used to program the slot table in every router on their path. The *SetUp* packet creates a connection from a source to a destination, and travels in the direction of the data (“downstream”). When a *SetUp* packet arrives at the destination it is successful and is acknowledged by returning an *AckSetUp*. *TearDown* packets destroy (partial) connections, and can travel in either direction. *SetUp* packets contain the source of the data, the destination or a path to it, and a slot number. Every router along the path of the *SetUp* packet checks if the output to the next router in the path is free in the slot indicated by the packet. If it is free, the output is reserved in that slot, and the *SetUp* packet is forwarded with an incremented (modulo  $S$ ) slot. Otherwise, the *SetUp* packet is discarded and a *TearDown* packet returns along the same path. Thus every path must be reversible;

this is the only assumption we make about the network topology. The upstream TearDown packet frees the slot, and continues with a decremented slot. Downstream TearDown packets work similarly, and remove existing connections. A connection is successfully opened when an AckSetUp is received, else a TearDown is received. With minor additions, system packets can also be used to program multicast connections.

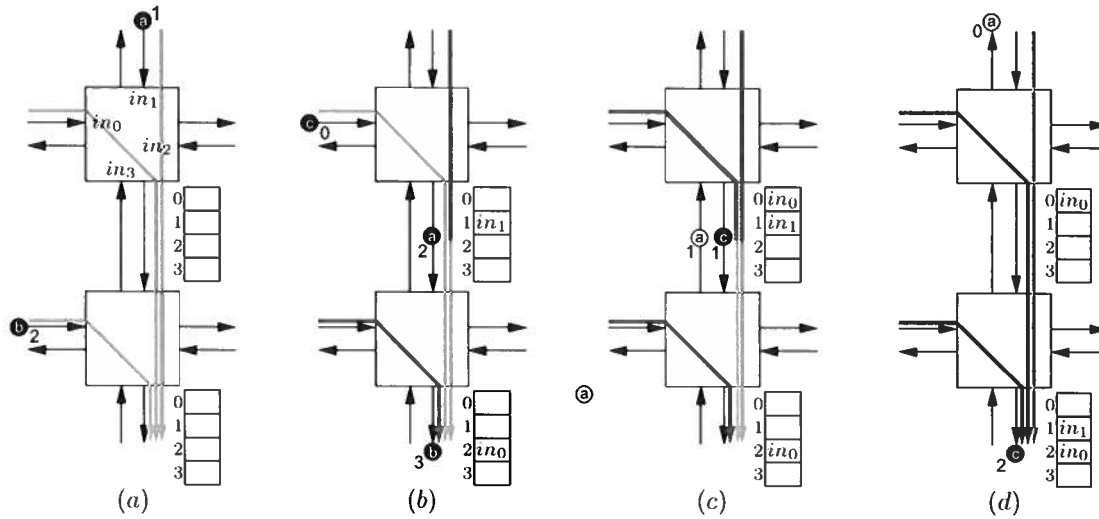


Figure 9: Set up of connections.

[The programming of the network is illustrated in Figure 9, where we attempt to set up three connections, shown as light gray arrows. The figure shows four snapshots of the same network at successive times. Two SetUp packets, *a* and *b*, enter the network in Figure 9(a). The number alongside the packet is the slot that is to be programmed in the next router. This is reflected in the slot tables in Figure 9(b), where only the column for output port *out*<sub>3</sub> is shown. A dark line shows the progress of a connection set up over time. The SetUp packets are routed to their next link and the slot to be programmed is incremented by one. In Figure 9(b) packet *a* cannot reserve slot 2 for output port *out*<sub>3</sub> of the bottom router, because it has been reserved for connection *c*. The set up of connection *a* fails. The packet *a* is changed from SetUp to TearDown and routed back along its path to remove the reservations made so far, see Figure 9(c). Note that the slot of packet *a* is decremented by one at every router. In Figure 9(d) packet *a* has removed the reservation of slot 1 that it made in Figure 9(b). For clarity AckSetUp packets of connections *b* and *c* are not shown in the figure.]

The programming model is pipelined and concurrent (multiple system packets can be active in

the network simultaneously, also from the same source) and distributed (active in multiple routers). Given the distributed nature of the programming model, ensuring consistency and determinism is crucial. The outcome of programming may depend on the execution order of system packets, but is always consistent. The next section shows how to use this programming model.

#### 4.3.3 Compile- and run-time slot allocation

This section explains how to determine the slots specified in `SetUp` packets. A slot allocation for a single connection requires that, at every router along the path, the required output is free (not reserved by another connection) in the appropriate slot. Computing an optimal slot allocation for all connections requires a global network view and may be expensive. To reduce computational cost, heuristics can be used, possibly leading to non-optimal solutions.

`SetUp` packets of different connections do not fail if connections are set up with conflict-free slots or paths. All execution orders of `SetUp` packets then give the same result, so that compile-time slot allocations can be recreated deterministically at run time.

Optimal run-time slot allocation is hard without a global (and central) slot table view, which is non-scalable and slows down programming. Distributed run-time slot allocation is scalable, but lacks a global view and may, therefore, be suboptimal. Moreover, `SetUp` packets may interfere, making programming more involved, and perhaps non-deterministic (in the sense of depending on the programming actions of other NoC users). However, dynamic connection management at high rates will require distributed slot allocation. In a simple distributed greedy algorithm, all sources repeatedly generate random slot numbers for each set up until their connection succeeds. We conclude that our programming model allows both compile-time and run-time slot allocation. Computational complexity, deterministic results, and scalability can be balanced according to system requirements.

## 5 Current results and future work

The previous section shows a prototype combined GT-BE architecture. We have synthesized an input-queued router using wormhole routing with arity 5, a queue depth of 8 flits of 3 words of 32 bits, and 256 slots in CMOS12 technology. The lay-out is shown in Figure 10, where the router and reconfiguration unit are shown separately (cf. Figure 8). It has an aggregate bandwidth of  $5 \times 500 \text{ MHz} \times 32 \text{ bit} = 80 \text{ Gbit/s}$ . The area of the router is  $0.26 \text{ mm}^2$ .

The area of  $0.26 \text{ mm}^2$  depends on the use of dedicated hardware fifos, labeled GQ and BQ in Figure 10. The router would have been at least three times larger with register- or RAM-based fifos.

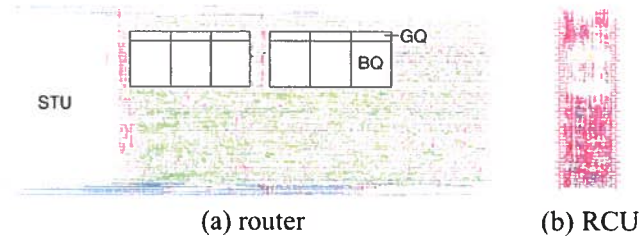


Figure 10: Lay-out of (a) a combined GT-BE router and (b) reconfiguration unit (RCU).

The RAMs required for input queuing and VOQ in an on-chip router have few addresses so that their overhead makes them as large as (area-inefficient) register files. Decreasing the queue depths reduces the buffering area (with registers at least), but also degrades the router performance.

Dedicated hardware fifos enable both input and virtual output queuing strategies using wormhole routing because of the reasonable buffering cost. For example, VOQ with two-flit deep fifos is only moderately larger than the input queuing with fifos of depth 8 of Figure 10. Virtual cut-through routing in combination with input queuing is also affordable now, because for packets of at most 8 flits, it has the same cost as the prototype.

The slot table (labeled STU in Figure 10) occupies a significant part of the router, for two reasons. Logically the slot table is very large (256 slots). It is not worthwhile to reduce the number of slots because the RAM is very area inefficient. We are investigating more advanced slot table schemes and new memory architectures to reduce the size and area of the slot table. The cost of offering time-related guaranteed services is then lower.

We separately synthesized the data and control paths (cf. Figure 7) with arities ranging from 3 to 13 to verify their speeds. With increasing arity, the speed of the data path reduces little. The speed of the control path decreases by a factor of two, corresponding to the complexity increase of the scheduling. For each arity, we balance the performance of the data and control paths by adjusting the flit size as needed, as shown in Section 4.3.1. The data and scheduling frequencies of the prototype router are 500 MHz and 166 MHz, respectively, with a flit size of 3 ( $F_{opt} = 500/166$ , cf. Section 4.3.1).

Our results show that the cost and performance of the combined GT-BE router can make it the basis of a router-based network on chip. It further shows that dedicated hardware fifos significantly reduce buffering area and so enable both input queuing and VOQ, with wormhole and virtual-cut through routing.

## 6 Conclusions

In this paper we show that guaranteed services are essential to provide predictable interconnects that enable compositional system design and integration. However, guarantees typically utilize resources inefficiently. Best-effort services overcome this problem but provide no guarantees. So, integrating guaranteed and best-effort services allows efficient resource utilization, yet still providing guarantees for critical traffic.

Time-related guarantees, such as throughput and latency, can only be constructed on a NoC that intrinsically has these properties. We therefore define a router-based NoC architecture that combines guaranteed and best-effort services. The router architecture has conceptually two parts: the guaranteed-throughput (GT) and best-effort (BE) routers. Both offer data integrity, lossless data delivery, and in-order data delivery. Additionally, the GT router offers guaranteed throughput and latency services using pipelined circuit switching with time-division multiplexing. The BE router uses packet switching, virtual cut-through or wormhole routing, and input queuing or virtual output queuing.

We combine the GT and BE router architectures efficiently by sharing router resources. The guarantees are never affected by the BE traffic, and links are efficiently utilized because BE traffic uses all bandwidth left over by GT traffic. Connections are programmed using BE packets. The programming model is robust, concurrent, and distributed. It enables run-time and compile-time, deterministic and adaptive connection management.

For all our architecture choices, we show the trade offs between hardware complexity and efficiency. Our choices are motivated by a prototype router which has an area of  $0.26 \text{ mm}^2$  in CMOS12 and offers 80 Gbit/s aggregate throughput. We use dedicated hardware fifos to significantly reduce the area of the data queues. With RAM- or register-based queues the router area would have been at least 3 times larger.

Dedicated hardware fifos enable (a) input queuing using both wormhole and virtual cut-through routing, and (b) virtual output queuing using wormhole routing. The buffer costs are too high, however, for virtual output queuing with virtual cut-through routing.

The cost of offering time-related guaranteed services is still high for our router. We are investigating how to reduce this cost.

An attractive feature of our router architecture is the ability to combine separately optimized data and control paths by adjusting the flit size.

In conclusion, we describe and motivate a choice of architectures for routers, which are an essential component in a NoC. They fulfill our NoC requirements by providing guaranteed services, and

satisfy the efficiency constraint by offering best-effort services.

## References

- [1] M.K.M. Ali and M. Youssefi. The performance analysis of an input access scheme in a high-speed packet switch. In *INFOCOM '91. Proceedings. Tenth Annual Joint Conference of the IEEE Computer and Communications Societies. Networking in the 90s., IEEE*, volume 2, pages 454–461, 1991.
- [2] John Bainbridge and Steve Furber. CHAIN: A delay-insensitive chip area interconnect. *IEEE Micro*, 22(5), 2002.
- [3] Luca Benini and Giovanni De Micheli. Powering networks on chips. In *ISSS*, pages 33–38, October 2001.
- [4] Luca Benini and Giovanni De Micheli. Networks on chips: A new SoC paradigm. *IEEE Computer*, 35(1):70–80, 2002.
- [5] William J. Dally and Brian Towles. Route packets, not wires: On-chip interconnection networks. In *Design Automation Conference*, pages 684–689, June 2001.
- [6] André DeHon. Robust, high-speed network design for large-scale multiprocessing. A.I. Technical report 1445, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, September 1993.
- [7] K. Goossens, J. van Meerbergen, A. Peeters, and P. Wielage. Networks on silicon: Combining best-effort and guaranteed services. In *Proceedings of Design Automation and Test Conference in Europe*, pages 423–425, March 2002.
- [8] Kees Goossens, John Dielissen, Jef van Meerbergen, Peter Poplavko, Andrei Rădulescu, Edwin Rijpkema, Erwin Waterlander, and Paul Wielage. Guaranteeing the quality of services in networks on chip. In Axel Jantsch and Hannu Tenhunen, editors, *Networks on Chip*, chapter 4, pages 61–82. Kluwer, 2003.
- [9] Pierre Guerrier and Alain Greiner. A generic architecture for on-chip packet-switched interconnections. In *Proceedings of Design Automation and Test Conference in Europe*, pages 250–256, 2000.

- [10] Mark J. Karol, Michael G. Hluchyj, and Samuel P. Morgan. Input versus output queueing on a space-division packet switch. *IEEE Trans. on Communications*, COM-35(12):1347–1356, 1987.
- [11] K. Keutzer, S. Malik, A. Richard Newton, Jan M. Rabaey, and A. Sangiovanni-Vincentelli. System-level design: Orthogonalization of concerns and platform-based design. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 19(12):1523–1543, 2000.
- [12] Nicholas William McKeown. *Scheduling Algorithms for Input-Queued Cell Switches*. PhD thesis, University of California, Berkeley, 1995.
- [13] Jennifer Rexford. *Tailoring Router Architectures to Performance Requirements in Cut-Through Networks*. PhD thesis, University of Michigan, department of Computer Science and Engineering, 1999.
- [14] Marshall T. Rose. *The Open Book: A Practical Perspective on OSI*. Prentice Hall, 1990.
- [15] Andrei Rădulescu and Kees Goossens. Communication services for networks on silicon. In Shuvra Bhattacharyya, Ed Deprettere, and Juergen Teich, editors, *Domain-Specific Processors: Systems, Architectures, Modeling, and Simulation*, pages 275–299. Marcel Dekker, 2003.
- [16] M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabaey, and A. Sangiovanni-Vincentelli. Addressing the system-on-a-chip interconnect woes through communication-based design. In *Design Automation Conference*, pages 667–672, June 2001.
- [17] Andrew S. Tanenbaum. *Computer Networks*. Prentice-Hall, 1996.
- [18] Hui Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proceedings of the IEEE*, 83(10):1374–96, October 1995.

# Exhibit E

# Trade Offs in the Design of a Router with Both Guaranteed and Best-Effort Services for Networks on Chip

E. Rijpkema, K.G.W. Goossens, A. Rădulescu,  
J. Dielissen, J. van Meerbergen, P. Wielage, and E. Waterlander  
Philips Research Laboratories, Eindhoven, The Netherlands

## Abstract

*Managing the complexity of designing chips containing billions of transistors requires decoupling computation from communication. For the communication, scalable and compositional interconnects, such as networks on chip (NoC), must be used. In this paper we show that guaranteed services are essential in achieving this decoupling. Guarantees typically come at the cost of lower resource utilization. To avoid this, they must be used in combination with best-effort services. The key element of our NoC is a router consisting conceptually of two parts: the so-called guaranteed throughput (GT) and best-effort (BE) routers. We combine the GT and BE router architectures in an efficient implementation by sharing resources. We show the trade offs between hardware complexity and efficiency of the combined router, and motivate our choices. Our reasoning for the trade offs is validated with a prototype router implementation. We show a lay-out of an input-queued wormhole  $5 \times 5$  router with an aggregate bandwidth of 80 Gbit/s. It occupies  $0.26 \text{ mm}^2$  in CMOS12. This shows that our router provides high performance at reasonable cost, bringing NoCs one step closer.*

## 1 Introduction

Recent advances in technology raise the challenge of managing the complexity of designing chips containing billions of transistors. A key ingredient in tackling this challenge is *decoupling the computation from communication* [10, 14]. This decoupling allows IPs (the computation part), and the interconnect (the communication part) to be designed independently from each other.

In this paper, we focus on the communication part. Existing interconnects (e.g., buses) may no longer be feasible for chips with many IPs, because of the diverse and dynamic communication requirements. *Networks on a chip* (NoC) are emerging as an alternative to existing on-chip interconnects because they (a) structure and manage global wires in new deep-submicron technologies [2–5, 7], (b) share wires, lowering their number and increasing their utilization [5, 7], (c) can be energy efficient and reliable [3], and (d) are scalable when compared to traditional buses [8].

Decoupling the computation from communication requires that the *services* that IPs use to communicate are well-defined, and hide the implementation details of the interconnect [10], Figure 1(a). NoCs help, because they are traditionally designed using layered protocol stacks [13], where each layer provides a well-defined interface which decouples service usage from service implementation [4, 14], see Figure 1(b).

In particular, *guaranteed services* are essential because they make the requirements on the NoC explicit, and limit the possible interactions of IPs with the communication environment. IPs can also be designed independently, because their use of guaranteed services is not affected by the interconnect or by other IPs.

This is essential for a compositional construction (design and programming) of systems on chip (SoC). Moreover, failures are restricted to the IP configuration phase (a service request is either granted or denied by the NoC) which simplifies the IP programming model [7]. We view the guaranteed services to be offered by an interconnect as a requirement from the applications, see Figure 1(c).

The drawback of using guaranteed services is that they require resource reservations for worst-case scenarios. This is not acceptable in a SoC where cost constraints are typically very tight, see Figure 1(d). Therefore, we also provide *best-effort services* to exploit the network capacity that is left over, or reserved but unused. Guaranteed services are then used for the critical (e.g. real-time) traffic, and best-effort services for non-critical communication.

The combination of guaranteed and best-effort classes is known from general computer network research [15], but not for on-chip networks. As on- and off-chip networks have different characteristics, the trade offs in their design are different. In this paper, we present the trade offs between hardware complexity and efficiency for networks on chip, and motivate our choices.

We present a prototype router architecture that reflects one particular set of design choices. It has an aggregate bandwidth of 80 Gbit/s, and its CMOS12 lay-out occupies  $0.26 \text{ mm}^2$ . We list other feasible variations that either increase performance, or lower the router cost.

In this paper, we first list a set of network-independent communication services that are essential in chip design (Section 2). Then, we show the trade-offs between efficiency and cost that we make in our NoC. In Section 3, we present some general network-related issues that are used in the sections to follow. In Section 4, we zoom into the internals of the key component of our NoC: a router that efficiently provides both guaranteed and best-effort services. In Section 5, we demonstrate the feasibility of our router design through a prototype implementation in CMOS12.

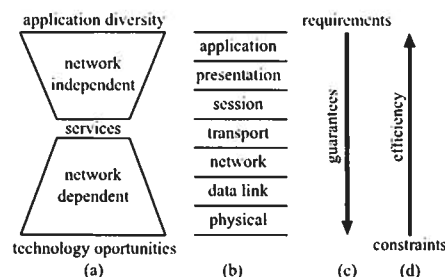


**Figure 1.** Network services (a) hide the interconnect details and allow reusable components to be build on top of them, (b) are build using a layered approach (c) are driven by the application requirements, and (d) their efficiency relies on technology and network organization.

## 2 Services

The NoC services that we consider essential for chip design are: *data integrity*, meaning that data is delivered uncorrupted, *lossless data delivery*, which means no data is dropped in the interconnect, *in-order data delivery*, which specifies that the order in which data is delivered is the same order in which it has been sent, and *throughput* and *latency* services that offer time-related bounds. As shown in Section 1, guaranteed services are essential to simplify IP design and integration. With the current technology, we assume data integrity is solved at the data-link layer. All the other services can be guaranteed or not on request. In the next section, we describe briefly how these services are provided by our NoC, and in Section 4 we describe how our router architecture enables an efficient implementation of these services.

*Guaranteed services* require resource reservation for worst-case scenarios, which can be expensive. For example, guaranteeing throughput for a stream of data implies reserving bandwidth for its peak throughput, even when its average is much lower. As a consequence, when using guarantees, resources are often under-utilized.

*Best-effort services* do not reserve any resources, and hence provide no guarantees. Best-effort services use resources well because they are typically designed for average-case scenarios instead of worst-case scenarios. They are also easy and fast to use, as they require no resource reservation. Their main disadvantage is their unpredictability: one cannot rely on a given performance (i.e., they do not offer guarantees). In the best case, if certain boundary conditions are assumed, a statistical performance can be derived.

The requirements for guaranteed services and the efficiency constraint (i.e., good resource utilization) are conflicting. Our approach to a predictable and low-cost interconnect is to integrate the guaranteed and best-effort services in the same interconnect. Guaranteed services would be used for critical traffic, and best-effort services for non-critical traffic. For example a video processing IP will typically require a lossless, in-order video stream with guaranteed throughput, but possibly allows corrupted samples. Another example is cache updates which require uncorrupted, lossless, low-latency data transfer, but ordering and guaranteed throughput are less important. In Section 4.3 we show how integrated guaranteed and best-effort services efficiently can use common resources. In the remainder of this section we analyze the minimum level of abstraction at which the communication services must be offered to hide the network internals.

Traditionally, network services have been implemented and offered using a layered protocol stack, typically aligned to the ISO-OSI reference model [13], see Figure 1(b). NoCs also take this approach [3, 4, 7, 14], because it structures and decomposes the service implementation, and the protocol stack concepts aid positioning of services.

To achieve the decoupling of computation from communication, the communication services must be offered at least at the level of the transport layer in OSI reference model. It is the first layer that offers end-to-end services, hiding the network details; see Figure 1(a, b) [4].

The lowest three layers in the protocol stack, namely physical, data-link, and network layers, are network specific. Therefore, these services should not be visible to the IPs when decoupling between computation from communication is desired. However, these layers are essential in implementing the services, because

constructing guarantees without guarantees at the layer below is either very expensive, or even impossible. For example, implementing a lossless communication on top of a lossy service requires acknowledgment, data retransmission, and filtering duplicated data. This leads to an increase in traffic, and possibly larger buffer space requirements. Even worse, providing guarantees for time-related services is impossible if lower layers do not offer these guarantees. For example, latency can not be guaranteed if communication at a lower layer is lossy. As a consequence, guarantees can only be built *on top of* guarantees, see Figure 1(c). Similarly, a layer's efficiency is based on efficient implementations of the layers below it, see Figure 1(d).

## 3 Networks on chip

General computer network research is a mature research field [15] which has many issues in common with NoCs. However, two significant differences between computer networks and on-chip networks make the trade offs in their design very different [5]. First, routers of a NoC are more resource constrained than those in computer network, in particular in the control complexity and in the amount of memory. Second, communication links of a NoC are relatively shorter than those in computer networks, allowing tight synchronization between routers.

We identify three important issues in the design of the router network architecture. These are: the *switching mode*, *contention resolution*, and *network flow control*. Equally important, *end-to-end flow control* and *congestion control* are handled in our NoC at the network edge instead of the routers; we therefore omit their discussion here. Similarly, we assume guaranteed data integrity at the link level and retain it at the network layer and higher.

### 3.1 Switching mode

The *switching mode* of a network specifies how data and control are related. We distinguish *circuit switching* and *packet switching*. In circuit switching data and control are separated. The control is provided to the network to *set up a connection*. This results in a *circuit* over which all subsequent data of the connection is transported. In *time-division circuit switching* bandwidth is shared by time-division multiplexing connections over circuits. Circuit-switched networks inherently offer time-related guaranteed services after resources are reserved during the connection set up.

In *packet switching* data is divided into *packets* and every packet is composed of a control part, the *header*, and a data part, the *payload*. Network routers inspect, and possibly modify, the headers of incoming packets to switch the packet to the appropriate output port. Since in packet switching the packets are self contained, there is no need for a set-up phase to allocate resources. Therefore, best-effort services are naturally provided by packet switching.

### 3.2 Contention resolution

When a router attempts to send multiple data items over the same link at the same time *contention* is said to occur. As only one data item can be sent over a link at any point in time, a selection among the contending data must be made; this process is called contention resolution.

In circuit switching, contention resolution takes place at set up at the granularity of connections, so that data sent over different connections do not conflict. Thus, there is no contention during data transport, and time-related guarantees can be given.

In packet switching contention resolution takes place at the granularity of individual packets. Because packet arrival cannot be predicted contention can not be avoided. It is resolved dynamically by scheduling in which data items are sent in turn. This requires data storage in the router, see Section 4.2.1, and delays the data in a non predictable manner which complicates the provision of guarantees, see Section 4.1.1.

### 3.3 Network flow control

*Network flow control*, also called *routing mode*, addresses the limited amount of buffering in routers and data acceptance between routers. In circuit switching connections are set up. The data sent over these connections is always accepted by the routers and hence no network flow control is needed. In packet switching, data items must be buffered at every router before they are sent on. Because routers have a limited amount of buffering they accept data only when they have enough space to store the incoming data.

There are three types of network flow control, namely *store-and-forward*, *virtual cut-through*, and *wormhole* routing. In store-and-forward routing, an incoming packet is received and stored in its entirety before it is forwarded to the next router. This requires storage for the complete packet, and implies a per-router latency of at least the time required for the router to receive the packet.

In virtual cut-through routing a packet is forwarded as soon as the next router guarantees that the complete packet will be accepted. When no guarantee is given, the router must be able to store the whole packet. Thus, virtual cut-through routing requires buffer space for a complete packet, like store-and-forward routing, but allows lower-latency communication.

In wormhole routing packets are split in so-called *flits* (flow control digits). A flit is passed to the next router when the flit can be accepted, even when there is not enough buffer space for the complete packet. As soon as a flit of a packet is sent over an output port, that output port is reserved for flits of that packet only. When the first flit of a packet is blocked the trailing flits can therefore be spread over multiple routers, blocking the intermediate links. Wormhole routing requires the least buffering (buffer flits instead of packets) and also allows low-latency communication. However, it is more sensitive to deadlock and generally results in lower link utilization than virtual cut-through routing.

To allow low latency we consider both virtual-cut through and wormhole routing, which are both feasible in terms of buffer area, as shown in Section 5.

## 4 A combined GT-BE router

Section 2 defines our requirements for NoCs in terms of services that are to be offered, in particular, both guaranteed and best-effort services. Using the general network issues of the previous section we show in the following two subsections that the guaranteed and best-effort services can conceptually be described by two independent router architectures. The combination of these two router architectures is efficient and has a flexible programming model, as described in Section 4.3. Section 5 then shows a prototype implementation.

### 4.1 A GT router architecture

Our guaranteed-throughput (GT) router guarantees uncorrupted, lossless, and ordered data transfer, and both latency and throughput over a finite time interval. As mentioned earlier, data

integrity is solved at the data-link layer; we do not address it further. The GT router is lossless because we use a variant of circuit switching, described in the next section. Data is transported in fixed-size blocks. As only one block is stored per input in the GT router, data items remain ordered per connection. We now turn to the more challenging time-related guarantees, namely throughput and latency.

#### 4.1.1 Time-related guarantees

Latency is defined as the duration a packet is transported over the network. Guaranteeing latency, therefore, means that a worst-case upper bound must be given for this time. We define throughput for a given producer-consumer pair as the amount of data transported by the network over a finite, fixed time interval. Guaranteeing throughput means giving a lower bound.

We observe that guaranteeing latency even in a lossless router is difficult because contention requires scheduling and hence cause delays. Guaranteeing throughput is less problematic. Rate-based packet switching (for an overview see [16]) offers guaranteed throughput over a finite period, and hence a latency bound. This bound is very high, however, and the cost of buffering is also high. Deadline-based packet switching [12] offers preferential treatment for packets close to their deadline. This allows differential latency guarantees (under certain admissible traffic assumptions), but also at high buffer costs.

Circuit switching solves the contention at set up, so naturally providing guaranteed latency and throughput. Circuits can be pipelined to improve throughput [6], at the cost of additional buffering and latency. Time-division multiplexing connections over pipelined circuits additionally offers flexibility in bandwidth allocation. This requires a logical notion of router synchronicity, which is possible because a NoC is better controllable than a general network. We explain this variation in more detail in the next subsection. The associated programming model is described in Section 4.3.2.

#### 4.1.2 Contention-free routing

A router uses a *slot table* to (a) avoid contention on a link, (b) divide up bandwidth per link between connections, and (c) switch data to the correct output. Every slot table  $T$  has  $S$  time slots (rows), and  $N$  router outputs (columns). There is a logical notion of synchronicity: all routers in the network are in the same fixed-duration slot. In a slot  $s$  at most one *block* of data can be read/written per input/output port. In the next slot  $(s + 1) \% S$ , the read blocks are written to their appropriate output ports. Blocks thus propagate in a store and forward fashion. The latency a block incurs per router is equal to the duration of a slot and bandwidth is guaranteed in multiples of block size per  $S$  slots.

The entries of the slot table map outputs to inputs for every slot:  $T(s, o) = i$ . An entry is empty, when there is no reservation for that output in that slot. No contention arises because there is at most one input per output. Sending a single input to multiple outputs (multicast) is possible.

The slots reserved for a block along its path from source to destination increase by one (modulo  $S$ ). If slot  $s$  is reserved in a router, slot  $(s + 1) \% S$  must be reserved in the next router on the path. The assignment of slots to connections in the network is an optimization problem, and is described in Section 4.3.3. Section 4.3.2 explains how slots are reserved in our network.

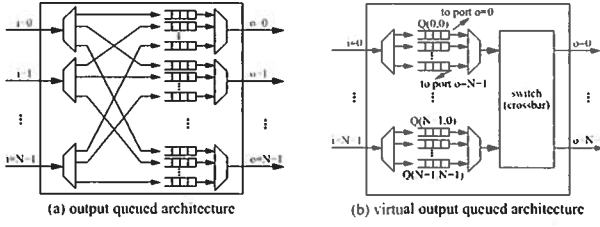


Figure 2. Schematics of two router architectures.

## 4.2 A BE router architecture

Best-effort traffic can have a better *average* performance than offered by guaranteed services. This depends on boundary conditions, such as network load, that are unpredictable. Best-effort services thus fulfill our efficiency requirement, but without offering time-related guarantees. This section describes an architecture for a best-effort service with uncorrupted, lossless, in-order data transport.

The BE router cost and performance are largely dependent on the contention resolution scheme of the router. The contention resolution scheme has two components: buffering and scheduling. The main trade off in Section 4.2.1 is between total buffer size, buffering strategy, and link utilization. Without taking global network requirements into account, no decisions will be made, rather we present a router that allows different instances, to trade off hardware complexity for link utilization at instantiation time. In Section 4.2.2 the trade off is between link utilization and schedule complexity and we select an efficient scheduling algorithm that is easily specialized to the different instances.

### 4.2.1 Buffering strategy

The buffering strategy determines the location of buffers inside the router. We distinguish *output queuing* and *input queuing*. In the following,  $N$  is the number of inputs, equal to the number of outputs, of our router. In output queuing  $N^2$  queues are located at the outputs of the router as in Figure 2(a). From the inputs to the outputs there is a fully connected bipartite interconnect to allow every input to write to every output. Output queuing has the best performance among the buffering strategies, however, the interconnect will make the router wire dominated and expensive already for small values of  $N$ .

In input queuing the queues are at the input of the router. A scheduler determines at which times which queues are connected to which output ports such that no contention occurs. The scheduler derives contention-free connections, a switch matrix (crossbar switch) can be used to implement the connections. In traditional input queuing, or input queuing for short, there is a single queue per input, resulting in a buffer cost of  $N$  queues per router. However, due to the so-called *head-of-line blocking*, for large  $N$ , router utilization saturates at 59% [9]. Therefore, input queuing results in weak utilization of the links.

Another version of input queuing is *virtual output queuing* (VOQ) [1]. VOQ combines the advantages of input queuing and output queuing. It has a switch like in input queuing and has the link utilization close to that of output queuing: 100% link utilization can still be achieved, when  $N$  is large [11]. As for output queuing, there are  $N^2$  queues. For every input  $i$  there are  $N$  queues  $Q(i, o)$ , one for each output  $o$ , see Figure 2(b). Typically the set of  $N$  queues at each input port of a VOQ router are mapped onto a single RAM. However, for NoCs we strive at a small router and therefore we require the RAMs to have few addresses. But

such RAMs have large overhead. Therefore, we use in-house developed dedicated fifos, which have almost no overhead, see Section 5.

The decision to select traditional input queuing or VOQ depends on system-level aspects like topology, network utilization, and global wiring cost, and is outside the scope of this paper. In Section 5 we show a prototype of an input queued router with dedicated hardware fifos and explain that VOQ is a valid option with minor additional cost.

### 4.2.2 Matrix scheduling

The switch matrix, present in input queued architectures, is controlled by a contention resolution algorithm, known as matrix scheduling, to properly connects inputs to outputs.

The matrix scheduling problem can be modeled as a bipartite graph matching problem. Every input port  $i$  is modeled by a node  $u_i$  and every output port  $o$  by a node  $v_o$ . There is an edge between  $u_i$  and  $v_o$  if and only if queue  $Q(i, o)$  is non-empty. A *match* is a subset of these edges such that every node is incident to at most one edge. For example, Figure 3(c) is a match of Figure 3(a).

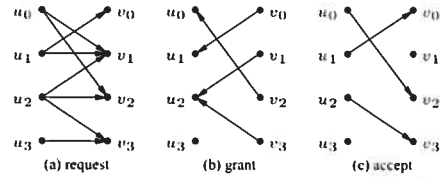


Figure 3. The three stages of a schedule iteration.

Matching can be done optimally, but because of time complexity and fairness, a non-optimal algorithm is preferred [11].

Our matching algorithm is iterative and one iteration has three stages, illustrated by an example in Figure 3 for  $N = 4$ . In the first stage, see Figure 3(a), every non-empty queue  $Q(i, o)$  requests access to output port  $o$  from input port  $i$ . In the second stage, see Figure 3(b), every output port  $o$  grants one request, solving link contention at the output ports. In the third stage, see Figure 3(c), every input port  $i$  accepts one grant, to resolve memory contention at the input port. A next iteration then starts with the matching found so far. This scheme is used in various scheduling algorithms, including parallel iterative matching, round robin matching, and SLIP [11], and applies to both input queuing and VOQ. For input queuing, however, stage (c) in Figure 3 is omitted since on contention on input ports can occur. To keep schedule latency as low as possible we use one iteration only.

### 4.3 Combining the GT and BE routers

The GT and BE router architectures are combined to share resources, in particular the links and the switch. Moreover, best-effort traffic enables a packet-based programming model for the guaranteed traffic, as shown later, in Section 4.3.2.

The principal constraint for a combined router architecture is that guaranteed services are never affected by best-effort services. Figure 4(a) shows that, conceptually, the combined router contains both router architectures (fat lines represent data, thin lines represent control). Incoming data is switched to either the GT or the BE router. The GT traffic, the traffic that is served by the GT router, has the higher priority, to maintain guarantees. This is ensured by the arbitration unit, which therefore affects the best-effort scheduling. Furthermore, best-effort packets can program the guaranteed router, as shown by the arrow labeled program. Thin lines going from the right to the left indicate network flow control, which

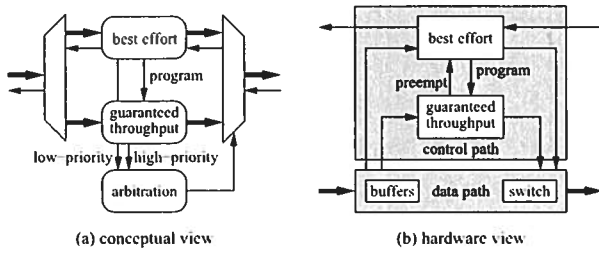


Figure 4. Two views of the combined GT-BE router.

is only required for best-effort packets because guaranteed blocks never encounter contention.

Figure 4(b) shows that the data path, consisting of buffers and switch matrix, is shared, and that the control paths of the BE and GT routers are separate, yet interrelated. Moreover, the arbitration unit of Figure 4(a) has been absorbed by the BE router. The following subsection shows how this can be done.

#### 4.3.1 Arbitration and flit size

When combining GT and BE traffic in a single network the impact on the network flow control scheme must be taken into account. Recall from Section 3.3 that a BE flit is the smallest unit at which flow control is performed. In other words, the BE scheduling can only react to GT blocks at flit granularity. To avoid alignment problems, the block size ( $B$  words) is a multiple of the flits ( $F$  words,  $B = \ell F$ ) with  $\ell$  being constant. We prefer a small  $\ell$  to decrease the store-and-forward delay and reduce the buffer size for guaranteed traffic, and a small  $F$  for fine-grained switching and better statistical multiplexing.

The router architecture contains a data path and a control path, see Figure 4(b). The data path maximizes throughput for high link utilization, and the control path maximizes the rate of scheduling and switching. They can be designed and optimized independently. Given any combination of their operating frequencies, the router has both maximum throughput and switching rate by using the appropriate flit size  $F_{opt}$ . For  $F > F_{opt}$ , the control path is ready while data is still being transported, lowering the switching rate. For  $F < F_{opt}$ , flits have been transported before the control path finishes, wasting bandwidth.

We extend the schedule algorithm in Section 4.2.2, to handle the combination of GT and BE traffic. In this combination GT traffic always has priority over BE traffic. This is to ensure that guarantees are never corrupted.

#### 4.3.2 Programming model

In this section we show how GT connections are set up and torn down by means of BE packets to avoid introducing an additional communication infrastructure only to program the network. To ensure scalability, programming must not require a global view or centralized resources. Section 4.1.2 explains why our contention-free routing uses slot tables; we now see that they are distributed over routers for scalability.

Initially the slot table of every router is empty. There are three system packets: SetUp, TearDown, and AckSetUp. They are used to program the slot table in every router on their path. The SetUp packet creates a connection from a source to a destination, and travels in the direction of the data ("downstream"). When a SetUp packet arrives at the destination it is successful and is acknowledged by returning an AckSetUp. TearDown packets destroy (partial) connections, and can travel in either direction. SetUp packets

contain the source of the data, the destination or a path to it, and a slot number. Every router along the path of the SetUp packet checks if the output to the next router in the path is free in the slot indicated by the packet. If it is free, the output is reserved in that slot, and the SetUp packet is forwarded with an incremented (modulo  $S$ ) slot. Otherwise, the SetUp packet is discarded and a TearDown packet returns along the same path. Thus every path must be reversible; this is the only assumption we make about the network topology. These upstream TearDown packets free the slot, and continue with a decremented slot. Downstream TearDown packets work similarly, and remove existing connections. A connection is successfully opened when an AckSetUp is received, else a TearDown is received.

The programming model is pipelined and concurrent (multiple system packets can be active in the network simultaneously, also from the same source) and distributed (active in multiple routers). Given the distributed nature of the programming model, ensuring consistency and determinism is crucial. The outcome of programming may depend on the execution order of system packets, but is always consistent. The next section shows how to use this programming model.

#### 4.3.3 Compile- and run-time slot allocation

This section explains how to determine the slots specified in SetUp packets. A slot allocation for a single connection requires that, at every router along the path, the required output is free (not reserved by another connection) in the appropriate slot. Computing an optimal slot allocation for all connections requires a global network view and may be expensive. To reduce computational cost, heuristics can be used, possibly leading to non-optimal solutions.

SetUp packets of different connections do not fail if connections are set up with conflict-free slots or paths. All execution orders of SetUp packets then give the same result, so that compile-time slot allocations can be recreated deterministically at run time.

Optimal run-time slot allocation is hard without a global (and central) slot table view, which is non-scalable and slows down programming. Distributed run-time slot allocation is scalable, but lacks a global view and is, therefore, perhaps suboptimal. Moreover, SetUp packets may interfere, making programming more involved, and perhaps non-deterministic. However, dynamic connection management at high rates will require distributed slot allocation. In a simple distributed greedy algorithm, all sources repeatedly generate random slot numbers for each set up until their connection succeeds. We conclude that our programming model allows both compile-time and run-time slot allocation. Computational complexity, deterministic results, and scalability can be balanced according to system requirements.

### 5 Current results and future work

The previous section shows a prototype combined GT-BE architecture. We have synthesized an input-queued router using wormhole routing with arity 5, a queue depth of 8 flits of 3 words of 32 bits, and 256 slots in CMOS12 technology. The lay-out is shown in Figure 5. It has an aggregate bandwidth of  $5 \times 500 \text{ MHz} \times 32 \text{ bit} = 80 \text{ Gbit/s}$ . The area of the router is  $0.26 \text{ mm}^2$ .

The area of  $0.26 \text{ mm}^2$  depends on the use of dedicated hardware fifos, labeled GQ and BQ in Figure 5. The router would have been at least three times larger with register- or RAM-based fifos. The RAMs required for input queuing and VOQ in an on-chip router have few addresses so that their overhead makes them



Figure 5. Lay-out of a combined GT-BE router.

as large as (area-inefficient) register files. Decreasing the queue depths reduces the buffering area (with registers at least), but also degrades the router performance.

Dedicated hardware fifos enable both input and virtual output queuing strategies using wormhole routing because of the reasonable buffering cost. For example, VOQ with two-flit deep fifos is only moderately larger than the input queuing with fifos of depth 8 of Figure 5. Virtual cut-through routing in combination with input queuing is also affordable now, because for packets of at most 8 flits, it has the same cost as the prototype.

The slot table (labeled STU in Figure 5) occupies a significant part of the router, for two reasons. Logically the slot table is very large (256 slots). It is not worthwhile to reduce the number of slots because the RAM is very area inefficient. We are investigating more advanced slot table schemes and new memory architectures to reduce the size and area of the slot table. The cost of offering time-related guaranteed services is then lower.

We separately synthesized the data and control paths (cf. Figure 4) with arities ranging from 3 to 13 to verify their speeds. With increasing arity, the speed of the data path reduces little. The speed of the control path decreases by a factor of two, corresponding to the complexity increase of the scheduling. For each arity, we balance the performance of the data and control paths by adjusting the flit size as needed, as shown in Section 4.3.1. The data and scheduling frequencies of the prototype router are 500 MHz and 166 MHz, respectively, with a flit size of 3.

Our results show that the cost and performance of the combined GT-BE router can make it the basis of a router-based network on chip. It further shows that dedicated hardware fifos significantly reduce buffering area and so enable both input queuing and VOQ, with wormhole and virtual-cut through routing.

## 6 Conclusions

In this paper we show that guaranteed services are essential to provide predictable interconnects that enable compositional system design and integration. However, guarantees typically utilize resources inefficiently. Best-effort services overcome this problem but provide no guarantees. So, integrating guaranteed and best-effort services allows efficient resource utilization, yet still providing guarantees for critical traffic.

Time-related guarantees, such as throughput and latency, can only be constructed on a NoC that intrinsically has these properties. We therefore define a router-based NoC architecture that combines guaranteed and best-effort services. The router architecture has conceptually two parts: the guaranteed-throughput (GT) and best-effort (BE) routers. Both offer data integrity, lossless data delivery, and in-order data delivery. Additionally, the GT router offers guaranteed throughput and latency services using pipelined circuit switching with time-division multiplexing. The BE router uses packet switching, virtual cut-through or wormhole routing, and input queuing or virtual output queuing.

We combine the GT and BE router architectures efficiently by sharing router resources. The guarantees are never affected by the BE traffic, and links are efficiently utilized because BE traffic uses all bandwidth left over by GT traffic. Connections are programmed using BE packets. The programming model is robust, concurrent, and distributed. It enables run-time and compile-time, deterministic and adaptive connection management.

For all our architecture choices, we show the trade offs between hardware complexity and efficiency. Our choices are motivated by a prototype router which has an area of 0.26 mm<sup>2</sup> in CMOS12 and offers 80 Gbit/s aggregate throughput. We use dedicated hardware fifos to significantly reduce the area of the data queues. With RAM- or register-based queues the router area would have been at least 3 times larger.

Dedicated hardware fifos enable (a) input queuing using both wormhole and virtual cut-through routing, and (b) virtual output queuing using wormhole routing. The buffer costs are too high, however, for virtual output queuing with virtual cut-through routing.

The cost of offering time-related guaranteed services is still high for our router. We are investigating how to reduce this cost.

An attractive feature of our router architecture is the ability to combine separately optimized data and control paths by adjusting the flit size.

In conclusion, we describe and motivate a choice of architectures for routers, which are an essential component in a NoC. They fulfill our NoC requirements by providing guaranteed services, and satisfy the efficiency constraint by offering best-effort services.

## References

- [1] M. Ali and M. Youssefi. The performance analysis of an input access scheme in a high-speed packet switch. In *INFOCOM*, 1991.
- [2] J. Bainbridge and S. Furber. CHAIN: A delay-insensitive chip area interconnect. *IEEE Micro*, (5), 2002.
- [3] L. Benini and G. De Micheli. Powering networks on chips. In *ISSS*, 2001.
- [4] L. Benini and G. De Micheli. Networks on chips: A new SoC paradigm. *IEEE Computer*, 35(1):70–80, 2002.
- [5] W. J. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *DAC*, 2001.
- [6] A. deHon. Robust, high-speed network design for large-scale multiprocessors. TR 1445, MIT, AI Lab., 1993.
- [7] K. Goossens et al. Networks on silicon: Combining best-effort and guaranteed services. In *DATE*, 2002.
- [8] P. Guerrier and A. Greiner. A generic architecture for on-chip packet-switched interconnections. In *DATE*, 2000.
- [9] M. J. Karol et al. Input versus output queueing on a space-division packet switch. *IEEE Trans. on Communications*, COM-35(12):1347–1356, 1987.
- [10] K. Keutzer et al. System-level design: Orthogonalization of concerns and platform-based design. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 19(12):1523–1543, 2000.
- [11] N. McKeown. *Scheduling Algorithms for Input-Queued Cell Switches*. PhD thesis, Univ. of California, Berkeley, 1995.
- [12] J. Rexford. *Tailoring Router Architectures to Performance Requirements in Cut-Through Networks*. PhD thesis, Univ. Michigan, 1999.
- [13] M. T. Rose. *The Open Book: A Practical Perspective on OSI*, 1990.
- [14] M. Sgori et al. Addressing the system-on-a-chip interconnect woes through communication-based design. In *DAC*, 2001.
- [15] A. S. Tanenbaum. *Computer Networks*, 1996.
- [16] H. Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proc. of the IEEE*, 83(10):1374–96, 1995.

# Exhibit F

# A Router Architecture for Networks on Silicon

Edwin Rijpkema, Kees Goossens, and Paul Wielage  
Philips Research Laboratories

Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands

E-mail: [edwin.rijpkema, kees.goossens, paul.wielage@philips.com](mailto:edwin.rijpkema, kees.goossens, paul.wielage@philips.com)

**Abstract**— To deal with the increasing design complexity of integrated systems reuse of intellectual property (IP) blocks is promoted. A system architecture then becomes a composition of a heterogeneous set of such IP blocks together with a network that interconnects these blocks. The main challenge of system design therefore shifts from computation (IP blocks) to communication and storage (interconnect and memories). This means that applications become dynamic compositions of IP blocks which requires that the network is scalable (in the number of attached IP blocks), programmable and behaves predictably under the traffic offered by those blocks.

As the feature size decreases the relative cost of wires increases. We therefore search for an interconnect network that efficiently uses wires through sharing by introducing routers.

For a flexible and efficient solution at least two traffic classes must be supported by the network, viz., guaranteed-throughput (GT) and best-effort (BE). For GT traffic communication channels are set up to transport data between IP blocks (possibly via memory). Best-effort traffic is never lost, but no latency or throughput guarantees are given. We also address the conflicting requirements of GT and BE traffic [1]. Our router is packet-switched and uses input-queuing with an efficient packet/flit scheduling [2] for BE traffic, whereas efficient time division multiplexing scheme is used for GT traffic.

The focus of this paper is on the derivation of a cost-effective router and network suitable for on-chip integration.

## I. INTRODUCTION

As the feature size of transistors decreases more functionality can be put on a single chip which implies an increased design complexity. To deal with this complexity the reuse of previously designed IP is to be supported. IP reuse can be done between two extremes.

In one extreme approach the granularity of reuse does not change: the IP blocks of old systems are reused directly in the new system. All memory that appears in the old system is mapped onto off-chip shared memory. Typically the communication bandwidth within a system is an order of magnitude higher than the bandwidth that is communicated over the ports of that system. So, *opening* such a system causes the communication bandwidth that is vis-

ible in the architecture to increase rapidly with the speed<sup>1</sup> of the circuits and the number of IP blocks. This approach benefits from the relatively low cost of off-chip memory compared with that of on-chip memory but suffers from the high off-chip memory bandwidth.

In the other extreme approach IP reuse is coarser: a complete old system defines a single new IP block for reuse in the new system. The memory of the old system is mapped to local embedded memory in the new system, which is composed now of subsystems exclusively. This approach preserves the locality in the subsystem. As a consequence the total communication that is visible at the top level in the architecture may be roughly the same or even less than in the old system. Clearly, this approach suffers from the relative high cost of embedded memory.

The way system designers design systems will be somewhere in between these extreme approaches but we observe a shift from the first to the second approach. To be precise, this shift is not so much in the size of the memories but rather in the communication bandwidth to and from the memories. This means that with relatively small on-chip embedded memories, the potential communication bottleneck to off-chip memory is avoided.

From this we conclude that the communication bandwidth requirements at the top-level architecture are moderate.

To allow the communication between the IP blocks and between IP blocks and memory a communication network is required. In order to derive such a network we take a look to it from two different perspectives, viz., the system-level design and circuit-level design perspectives.

### System-level design perspective

In general the IP blocks have various traffic characteristics and require differing services of the interconnection network. For example, real-time signal processing IP blocks may require a service that provides them with a guaranteed bandwidth and, at the same time, minimize the jitter (delay variation), whereas a CPU-like IP block requires a latency as low as possible for it to operate efficiently.

<sup>1</sup> Up-scaling of the speed of a circuit from one technology to a newer one is a problem on itself and is not addressed in this paper.

To design a system a communication network is required that meets the set of all simultaneous communication requirements of the individual IP blocks in a certain application. To support the design of such system predictable behavior of the network is essential. For example, when a set of streaming IP blocks is to be connected to the network, it is essential to know whether or not a particular network can simultaneously provide the IP blocks with their guaranteed-throughput requirements.

The IP blocks that will appear in future systems may contain applications on itself. Thus new applications are compositions of smaller applications. To allow the mapping of a complete application domain onto such a system the network must allow a flexible composition of IP blocks to form a new application. It also requires a high degree of connectivity. In addition, communication patterns may change dynamically, for example as the result of user interaction.

### Circuit-level design perspective

The decrease of feature size speeds up the IP in every new process generation. When the global wires that connects these IP blocks are scaled in width and spacing with a factor of about 0.7 every process generation, then the available bandwidth on these wires decreases [3]. Since not only GT services but also BE services are required and since BE traffic may be latency constrained, low latency interconnect is required. This means that it is not sufficient to just segment all long wires into short wires (only increases available bandwidth) but that long wires must have low latency themselves. This can be accomplished by not scaling down the width and spacing of the wires to meet lower latency requirements than was needed in a previous generation. Since the cost of these low latency wires is relatively high, it is important to exploit the offered bandwidth of the global wires as most as possible without degrading the latency of the packet too much.

From these two perspectives we conclude that we need a high degree of connectivity to support both GT and BE traffic while global wires are efficiently utilized. A network of routers allows sharing of wires. A router provides both GT and BE services to keep the utilization of the links high.

The rest of this paper is organized as follows: Section II deals with a number of issues that arise when designing such a network. Section III and Section IV describe an architecture for GT traffic and BE traffic, respectively. In Section V we draw some conclusions.

## II. NETWORK DESIGN ISSUES

This section deals with a number of issues that arise when designing a network. Services that are to be provided by the network and how the network internally is organized in order to enable such services are discussed in Section II-A. Section II-B introduces the *switching mode* of a network. These are *circuit switching* and *packet switching*. Section II-C evaluates the *routing mode* of a packet-switched network. The routing modes are *store-and-forward routing*, *virtual cut-through routing*, and *worm-hole routing*.

### A. Network services and organization

This section deals with the services the network must provide for the GT and BE traffic classes, and the internal organization of the network on top of which these services are to be implemented.

Two important issues in the design of a network (the network layer in the OSI model [4]) are the services provided by the network (to the transport layer) and internal organization of the network [4]. The services to be provided must support the traffic classes GT and BE efficiently. Proper internal organization of the network helps in implementing the required services. For example, when the network is such that no reordering of data can ever happen, no additional ordering hardware at the boundary of the network is required to implement in-order delivery.

Basically, interconnection networks may provide two types of services, viz., *connectionless* and or *connection-oriented* which can either be reliable or unreliable [4]. In a connection-oriented service a connection is set up prior to the communication of the actual data packets. When a connection has been established the packets arrive at the destination in the same order as they have been sent by the source. In a connectionless service all packets are sent independently and may arrive out of order at the destination. Moreover, a reliable service is one in which all packets that has been sent arrive at the destination (no packets are lost).

### Services provided by the network

Clearly, for GT traffic a reliable connection-oriented services is required. In addition to guaranteed, in-order delivery the service must secure the required bandwidth for each connection. To set up a GT connection admission control is done to decide whether or not the connection with the requested bandwidth is established. The network will not accept more data from a GT source than was agreed to prevent interference with other GT connections.

Guaranteed delivery is important for BE and GT traffic

because retransmission will increase the observed latency, decrease the throughput of the network, and increase contention (leading to more packet loss). No packets are therefore ever lost by the network. For BE traffic we believe that in-order delivery is also a desired property since it avoids the necessity to reorder at the network boundary. We will see that this does not complicate the internal organization of the network. Since for BE traffic we require that latency is as low as possible it is not acceptable to go through a connection set-up phase before the actual data may be communicated. Therefore we require that network provides a lossless connectionless service that provides in-order delivery.

### Internal organization of the network

As already pointed out in the introduction, we aim for high utilization of the physical wires. At the same time the internal rate the routers operate on is comparable to the rate on which the links are to be used. We therefore restrict the routers to the functionality required to implement the previously mentioned services (lossless, order-preserving).

### B. Switching mode

There are basically two switching modes: circuit switching and packet switching. In a circuit-switched network connections are set up by establishing a conceptual physical path from a source to a destination. In this basic scheme links are shared in the sense that two connections are allowed to use the same link only at different points in time. A way to share a link over multiple connections is to use a time-division multiplexing (TDM) scheme which is discussed in Section III.

In packet switching data is segmented into *packets*. A packet is composed of a header and a payload. The payload is the actual data to be transmitted and the header contains information that is used by the router to forward the packet to the destination. Classically the header contains the destination address but other approaches, like source routing, exist as well. In a packet-switched network links are not reserved for connections; sharing of a link is interpreted as interleaving of packets routed over that link.

An advantage of circuit switching with TDM over packet switching is that it naturally provides us with guaranteed-throughput connections. A disadvantage of circuit switching with TDM is the bad utilization of the links when there is much bursty traffic [5]. We therefore propose a router that supports both time-division multiplexed streams of GT traffic and packet switching for BE traffic to allow high link utilization. In Section IV-B we show that the proposed scheme can use the unreserved GT

and unused GT bandwidth for BE packets. Most alternative approaches are based on output queuing or use priority queues for time-stamped packets, both of which are expensive to implement. A packet-switched alternative that we know of is presented in Hung *et al.* [6]. However, our circuit-switching approach requires one buffer per input port instead of  $N$  buffers for packet switching of Hung *et al.* ( $N$  is the number of output ports of the router).

### C. Routing mode

The routing mode is also referred to as network flow control and applies to packet-switched networks. It deals with the way packets are transmitted between the routers. There are three routing modes, viz., store-and-forward routing, virtual cut-through routing, and wormhole routing [7].

In store-and-forward routing, an input packet is received in its entirety and stored in the router. Only then is it forwarded to the next router in the network. This requires storage for the complete packet. Moreover, a packet is sent only after complete reception, introducing a latency of the packet per router of at least the time required to receive the packet.

In virtual-cut-through routing a packet is forwarded as soon as the next router gives a guarantee that a packet will be accepted completely. So, when at some router the header of a packet arrives and the next router guarantees that it will consume a complete packet, the packet at the first router is sent immediately without first storing the complete packet. When at some time no guarantee is given, the packet whole packet is not forwarded and stored, in its entirety, in the router. Thus, virtual-cut-through routing requires buffers for a complete packet, like store-and-forward routing, but allows low-latency communication.

In wormhole routing packets are partitioned in so called flits (flow control digits). A flit is passed to the next router when that router indicates it accepts that flit. As soon as a flit of a packet is sent over an output port, that output port is reserved for flits of that packet only. So, a greedy choice of starting to communicate the first flit of a packet can cause a packet to be spread over a number of routers and when it blocks, it occupies all links in between them. So, like virtual-cut-through wormhole routing allows for low-latency communication. In addition wormhole routing requires less memory because it stores a flit rather than a packet. On the other hand, wormhole routing is more sensitive to deadlock and generally results in lower link utilization than virtual-cut through routing. One way to improve on wormhole routing's performance is by multiplexing multiple logical ports onto a single physical port. However, this substantially increases the cost of the

router.

So, for low-latency BE traffic we must select between virtual cut-through and wormhole routing. A complicated issue is how to combine the TDM circuit switching with packet switching with either one of these routing modes. In [1] it is shown that combining traffic that uses wormhole routing with traffic for which throughput guarantees are to be given can degrade link utilization drastically. In our context this is illustrated in Figure 1. Two  $2 \times 2$ -routers are indicated by  $R_1$  and  $R_2$ , and the network terminals are identified by  $t_i$ ,  $i = 1, 2, \dots, 6$ . Assume that  $R_1$  receives BE packets via terminal  $t_1$  that are all destined to  $t_5$  and that the bandwidth of these packets require 10% of the capacity of a link. Similarly, packets go from  $t_2$  to  $t_6$  and require only 1% of the link capacity. Finally there is a GT stream from  $t_4$  to  $t_6$ . The GT stream claims and uses 99%

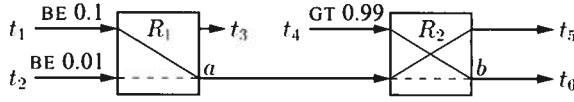


Fig. 1. Combining GT with wormhole routing can drastically degrade link utilization.

of the bandwidth and thus occupies the output link from output port  $b$  to terminal  $t_6$  for 99% of time. The BE stream shares port  $b$  can send a flit only in the remaining 1% link capacity. Every time GT data arrives for port  $b$  the transmission of the BE packet over port  $b$  is preempted. This can cause long latencies for the packets that belong to the 1% BE stream. It also causes the link between  $R_1$  and  $R_2$  to be occupied almost continuously by the 1% BE stream (because flits of different packet are not interleaved). Thus the BE packets of the 10% stream obtain less than 10% of the rate of the link. This means that in this example, the link between  $R_1$  and  $R_2$  has a utilization that is even below 11% of its theoretical capacity.

The cause of the problem in Figure 1 is clear. It is the combination of *flit preemption* and wormhole routing that blocks links for significant part of the time. To get around this problem there are basically two approaches: 1) use virtual cut-through routing rather than wormhole routing, and 2) perform GT communication in relatively large blocks of data and large periods of no data.

The first approach guarantees that a complete packet will be accepted in the next router such that the incoming link of the next router does not block. As mentioned before this is at the cost of extra memory.

The second approach ensures that flit preemption rarely occurs. Referring to the example this is seen as follows. When the 99% of GT data is grouped in blocks of 10 time

units then this bandwidth is obtained by alternative sending 99 blocks of data followed by 10 time units nothing. Now, when the packet size of the BE stream is small compared to these 10 time units, a complete packet of the 1% BE stream is sent in the 10 time units and the link between  $R_1$  and  $R_2$  can be used by the 10% BE stream immediately after the packet has been sent. While the first approach suffers from additional memory requirements in the router, this second approach may suffer from additional latency in the GT stream.

### III. A GUARANTEED-THROUGHPUT (GT) ROUTER ARCHITECTURE

This section explains how time-division-multiplexed circuit switching enables us to guarantee throughput. Since the operation rate of the links is about the same as the internal operation rate of the router, the router internally switches in space. We propose a crossbar switch for this. To prevent contention on the switch a separate set-up phase takes place. When a new GT connection would introduce a clash with existing connections on the switch, the requesting party is informed about this. Every router has a *router table* in which every row contains the information to program the crossbar switch. A row of the router table is referred to as a *slot*. We avoid contention by: first ensuring there is a notion of synchrony between all routers such that all routers go from one slot to the next simultaneously, and secondly filling slots such that the set of simultaneous GT connections is contention free.

Section III-A introduces the model of computation that specifies the required notion of synchrony in an abstract manner. Section III-B deals with the internal working of the GT router. Section III-C explains how GT connections are programmed.

#### A. Model of computation

We now explain the network of routers in the context of the synchronous dataflow (SDF) model [8]. An SDF network is a set of so called *actors* that are connected by *arcs*. An actor is an executable piece of computation and may contain state. When the piece of computation is executed the actor is said to *fire* and consumes a number of *tokens* from its input arcs, and produces a number of tokens on its output arcs. A token is an abstract representation of quantum of information and allows the actors to communicate. Every input port and output port of an SDF actor specifies the number of tokens that are produced (for an output port) or consumed (for an input port) when the actor fires. An actor may only be fired when it is *enabled*, i.e. when all its input arcs contain at least as many tokens as the number on corresponding input ports.

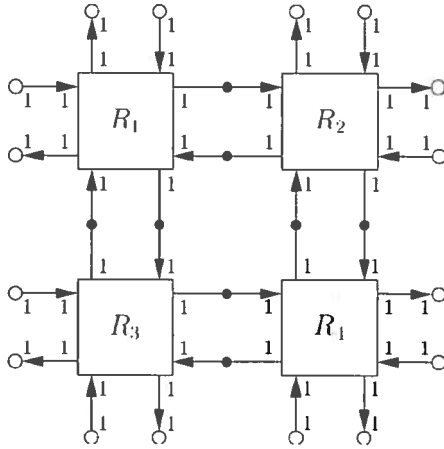


Fig. 2. Example of a router network in the SDF model of computation.

In SDF networks arcs may contain delays. A delay may be seen as a token that is initially present at the arc at which the delay is placed. In this paper we consider the case that every internal arc in the network has one delay, that is, one initial token.

The numbers associated with the ports are constant during the execution of the system but may vary from port to port. However, in this paper all ports consume one token per firing.<sup>2</sup> In this case an *SDF iteration* of the network is the single firing of every actor in the network, and the number of tokens on every arc before an after an SDF iteration is equal. From now we will iteration as shorthand for SDF iteration.

Every router of the router network is thus represented by an actor and every link is represented by an arc. An example of a simple router network in the SDF model with four  $2 \times 2$ -router is depicted in Figure 2. The circles in the figure represent network interfaces that connect the network to the sources and sinks. Note that after every iteration all internal arcs contain a token. Thus, when the network interfaces provide enough data to the network, all routers in the network can operate in parallel.

### B. Internal GT router architecture

To explain the internal router architecture consider an  $N \times N$  router, a router with  $N$  input ports and  $N$  output ports. The ports of the router are denoted by  $i_n$  and  $o_n$ ,  $n = 0, 1, \dots, N-1$ , respectively. The router table  $R$  is specified by an  $S \times N$  matrix. The rows of  $R$  are denoted

<sup>2</sup>In future work we will consider the production and consumption of more than one token per arc per firing to model multi-rate execution. In order to avoid “dummy” communication we might switch to cyclostatic dataflow [9], [10] as well.

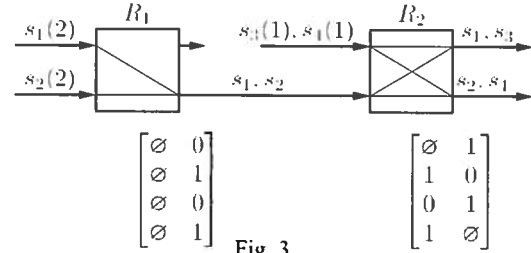


Fig. 3.

by vectors  $r_s$ ,  $s = 0, 1, \dots, S-1$ . The slot a router is in is called the current slot of that router. All routers are in the same current slot. The elements  $R(s, n)$  are in the set  $\{\emptyset, 0, 1, \dots, N-1\}$ . Let  $m = R(s, n)$ . When  $m \neq \emptyset$  output port  $o_m$  has been reserved. This means that GT tokens read at input port  $i_n$  in slot  $(s-1) \% S$  are sent to destination port  $o_m$  in slot  $s$ . When  $m = \emptyset$  it means that no such reservation is made for any of the input ports.

In our GT routing approach, every GT token that is read in slot  $r$ , in some router is read in slot  $r_{(s+1) \% S}$  in the next router in the path the token follows. (The symbol  $\%$  represents the remainder after integer division.)

An example of a simple router network with two  $2 \times 2$  routers with a router tables size  $S = 4$  is shown in Figure 3. In this figure four GT connections are represented by the streams  $s_1$ ,  $s_2$ ,  $s_3$ , and  $s_4$ . The number of slots allocated for that stream shown in parentheses.

The first output port of  $R_1$  is unused and consequently the first column of the routing table is empty. The second column of the routing matrix of  $R_1$  indicates that tokens from its inputs are written alternately on the second output port. Consequently both  $s_1$  and  $s_2$  are routed with the desired bandwidth without contention in the first router. In the second router the first output port receives tokens belonging to stream  $s_1$  and stream  $s_3$ . Since the tokens from  $s_1$  are routed in slots 0 and 2 in the first router, they are routed at slots 1 and 3 in the second router. This is seen by the two ones in the first column of  $R_2$ 's router table. The single slot that  $s_3$  requires is scheduled in slot 2 of the first column. Similarly, as indicated by the ones in the second column of  $R_2$ 's routing table, tokens of  $s_2$  are scheduled in slots 0 and 2. Finally,  $s_4$ 's tokens are scheduled in slot 1.

It is not required a GT token is available in every reserved slot. When no GT packet arrives in a reserved slot, a BE packet can be sent over the claimed but unused slot of the link. Since the SDF model requires a token to be written every iteration, when no BE or GT data is available, *empty tokens* are produced on the output arcs.

### C. Programming model

This section deals with the way GT connections are set up and torn down. Initially every router table is empty. Therefore, the only way to communicate with the routers is by means of BE packets. *System packets* are a special type of BE packets that are dedicated to manage the setting up and tearing down of GT connections. The way these system packets are routed is dealt with in Section IV-B.

There are three packet types that are used to set up and tear down connections, viz., *SetUp*, *AckSetUp*, and *TearDown*. In addition there are two packet types to reset and start a complete router network, viz., *Reset* and *Start*. There are more types of packet types (also to manage connections) but these are outside the scope of this paper.

A *SetUp* packet is used to set up a GT connection from a source to a destination. It can be sent by either the source or the destination of the GT packets. The same holds for *TearDown* and *AckSetUp* packets. However, here we assume that the *SetUp* is sent by the source of the GT tokens and that the *TearDown* and *AckSetUp* are sent in the direction of the source of the GT tokens.

*SetUp* packets contain three pieces of information: the source of the GT tokens, the destination of the GT tokens, and a slot number. The *SetUp* packet is routed towards the destination and tries to reserve the slot specified by the slot number. When this reservation is successful the slot number is incremented by one and the *SetUp* packet is routed to the next router (or the destination). When the *SetUp* packet successfully arrives at the destination, a *AckSetUp* packet is sent back to the source to inform it about the successful set-up. If, however, at some point the slot requested is already occupied the *SetUp* is discarded and a *TearDown* packet is sent back to the destination. This *TearDown* packet follows the same route the corresponding *SetUp* packet followed in order to remove the reservations made from the routing table. When the source receives the *TearDown* packet it knows the set-up has been unsuccessful.

The *Reset* packet initializes the entire network to bring it in a well-defined idle state. The *Start* packet is then used to activate the network services. Both packets flood the network such that every router receives these packets.

## IV. A BEST-EFFORT ROUTER (BE) ARCHITECTURE

The term queuing strategy is used in the context of packet switching. Basically there are two queuing strategies, viz., input queuing and output queuing. Classically, in input queuing a router has a single input queue for every incoming link while in output queuing there are  $N$

output queues for each output link<sup>3</sup> where  $N$  is the number of input links for the output queued router. Most research on giving guarantees assumes non-blocking output-queued routers [11]. When the internal rate of operation of the router is about the rate at which the wires are driven, output-queued routers require  $N^2$  physical buffers making them expensive for system-on-chip implementations. To avoid these high costs, we use an alternative queuing strategy, known as virtual output queuing (VOQ) [12], that tries to combine the advantages of input-queuing and output-queuing. Unlike input-queuing, VOQ does not suffer from head-of-line blocking. Not all virtual output queues cannot be accessed simultaneously, but with a proper scheduling algorithm 100% throughput can be achieved.

### A. Virtual-output-queuing and iSLIP

A basic scheme of an virtual-output-queued architecture for an  $N \times N$  router is shown in Figure 4.

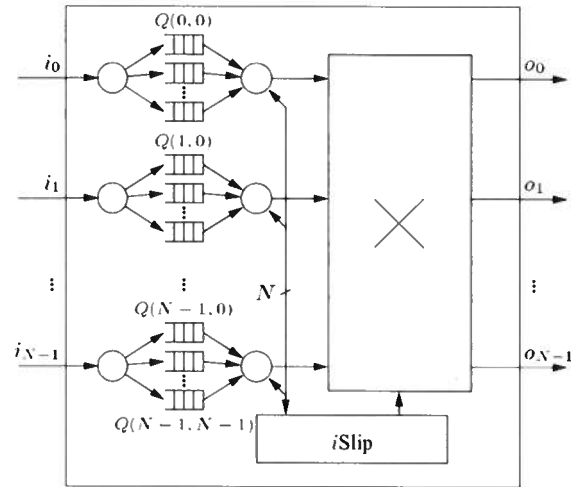


Fig. 4. Schematic of an virtual-output-queued router architecture.

For every input port there is a set of  $N$  logical queues, one for each output port. Queue  $Q(m, n)$  is the queue that contain the tokens that are read from input port  $i_m$  and are destined for output port  $o_n$ . One essential difference between VOQ and traditional output queuing is that in VOQ only one queue per input port can be read per iteration. This allows all logical queues of one input port to be implemented with a physical single memory. We assume that

<sup>3</sup> Instead of having  $N$  output queues per output port a single output queue that allows writing with a factor  $N$  times the rate of the link can be used. This is just a matter of switching in time or switching in space.

the actual switching is done with a crossbar switch. In order to meet the constraints of only one read per physical memory, of conflict free routing, and at most one write per output port, the read accesses on the queues must be scheduled. These two constraints are formulated as

$$\begin{aligned} \forall m. \sum_n A(m, n) &\leq 1 \\ \forall n. \sum_m A(m, n) &\leq 1 \end{aligned} \quad (1)$$

where  $A$  is a matrix with binary elements  $A(m, n)$  that indicate whether or not a token from input  $i_m$  is switched to output  $o_n$  in the current iteration of the router. The problem how to find  $A$  such that as many elements of  $A$  are 1 (at most  $N$ ) can be formulated as a bipartite graph matching problem. A bipartite graph is constructed as follows. Create a vertex  $u_m$  and  $v_n$  for every input port  $i_m$  and output port  $o_n$ , respectively. Create an edge  $(u_m, v_n)$  if and only if queue  $Q(m, n)$  is non-empty. Let  $R$  be the *request matrix* whose elements  $R(m, n) \in \{0, 1\}$  indicate whether or not  $Q(m, n)$  contains data. Figure 5 gives an example of a request matrix for  $N = 4$ , the corresponding bipartite graph and a matching for that graph.

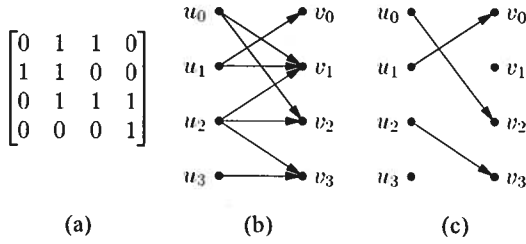


Fig. 5. Example of (a) a request matrix, (b) its derived bipartite graph, and (c) a corresponding matching.

The matching in the figure has *size* three. The matching is a *maximal matching* since no edges from the original graph can be added to increase the size of the matching. It is not a *maximum size matching* since there exists a matching of size four.

An efficient algorithm to find a maximal matching is iterative SLIP (*i*SLIP) [2]. The output of *i*SLIP can be used directly control a crossbar switch to perform the actual switching of the tokens. *i*SLIP iteratively computes a maximal matching in  $\log_2(N)$  iterations. However, for a low-latency implementation of *i*SLIP with low complexity we consider to use *i*SLIP with only one iteration.

Basically an *i*SLIP iteration works in three steps. In the first step for every non-empty queue  $Q(m, n)$  a *request* to access output port  $o_n$  is made. In the second step among all requests made to output port  $o_n$  a single request is *granted*.

In the third step for every input port at most one grant is *accepted*. For the bipartite graph in Figure 6(b) the three steps are illustrated in Figure 6.

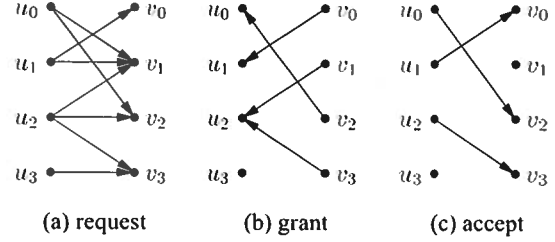


Fig. 6. Single iteration of *i*SLIP.

An integer value between 0 and  $N - 1$  is associated with every node  $u_m$  and every node  $v_n$  in the bipartite graph. Of all requests for output port  $o_n$  the request from the input port  $o_m$  with the smallest value  $m$  that is higher (modulo  $N$ ) than the value associated with  $o_m$ . Similarly, among the set of grants at input port  $i_n$  the one with the lowest index higher than the value associated with  $i_n$  is selected. In *i*SLIP, the values associated with the nodes are set to the selected indices only after an accept. An implementation of this arbitration method is done by means of grant arbiters [2].

#### B. Combining the BE with the GT Architecture

To efficiently use the bandwidth all slots that are not reserved and all slots that are reserved but not used by GT connections can be used by BE traffic. Only when there are no GT tokens and BE packets available to switch to an output port does the output port remain idle (recall that an empty packet is written to the output port in such cases). In our combined architecture the routing of the GT tokens is never affected by the BE traffic.

In order to deal with the system packets these packets are routed to an internal unit to process them and then pass them on to the proper output port. This internal unit has the responsibility to handle the system packets properly and to access the routing table accordingly.

#### V. CONCLUSIONS

We have motivated that high link utilization and high connectivity are two objectives in current and future system-on-chip designs. On the other hand, as system design becomes a composition (interconnection) of IP blocks, their communication requirements are to be met in a predictable and systematic way. In order to do so we introduced guaranteed-throughput (GT) and best-effort (BE) traffic classes and proposed a router architecture that

supports both. Internally the router architecture uses time-division-multiplexed circuit switching for GT traffic and uses virtual-input-queuing packet switching for BE traffic. We further proposed to use single iteration iSLIP for the contention resolution of the crossbar switch.

#### REFERENCES

- [1] Jennifer Rexford, *Tailoring Router Architectures to Performance Requirements in Cut-Through Networks*, Ph.D. thesis, University of Michigan, Dept. of Computer Science and Engineering, 1999.
- [2] Nicholas William McKeown, *Scheduling Algorithms for Input-Queued Cell Switches*, Ph.D. thesis, University of California, Berkeley, 1995.
- [3] Robert H. Havemann and James A. Hutchby, "High-performance interconnects: An integration overview," *Proceedings of the IEEE, special issue on Interconnections-Addressing the next Challenge of the IC Technology*, vol. 89, no. 5, pp. 586–601, may 2001.
- [4] Andrew S. Tanenbaum, *Computer Networks*, Prentice-Hall, Inc., 1996.
- [5] Jean Walrand, *Communication Networks: A First Course*, Richard D. Irwin, Inc., and Aksen Associates, Inc., 1991.
- [6] A. Hung, G. Kesidis, and N. McKeown, "ATM input-buffered switches with the guaranteed-rate property," in *IEEE ISCC'98*, Athens, Greece, july 1998.
- [7] Anujan Varma and Raghavendra C.S., *Interconnection Networks for Multiprocessors and Multicomputers: Theory and Practice*, IEEE Computer Society Press, 1994.
- [8] Edward A. Lee and D. G. Messerschmitt, "Synchronous data flow," *Proceedings of the IEEE*, september 1987.
- [9] G. Bilsen, M. Engels, R. Lauwereins, and J. Peperstaete, "Cyclo-static data flow," in *ICASSP-95, 1995 International Conference on Acoustics, Speech, and Signal Processing*, 1995, vol. 5, pp. 3255–3258.
- [10] Thomas M. Parks, José Luis Pino, and Edward A. Lee, "A comparison of synchronous and cycle-static dataflow," in *Conference Record of the Twenty-Ninth Asilomar Conference on Signals, Systems and Computers*, 1996, vol. 1, pp. 204–210.
- [11] Hui Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *Proceedings of the IEEE*, vol. 83, no. 10, pp. 1374–96, october 1995.
- [12] M.K.M. Ali and M. Youssefi, "The performance analysis of an input access scheme in a high-speed packet switch," in *INFOCOM '91. Proceedings. Tenth Annual Joint Conference of the IEEE Computer and Communications Societies. Networking in the 90s.*, IEEE, 1991, vol. 2, pp. 454–461.

# Exhibit G

# An Efficient On-Chip Network Interface Offering Guaranteed Services, Shared-Memory Abstraction, and Flexible Network Configuration

Andrei Rădulescu, John Dielissen, Kees Goossens, Edwin Rijpkema, and Paul Wielage  
Philips Research Laboratories, Eindhoven, The Netherlands

## Abstract

*In this paper we present a network interface for an on-chip network. Our network interface decouples computation from communication by offering a shared-memory abstraction, which is independent of the network implementation. We use a transaction-based protocol to achieve backward compatibility with existing bus protocols such as AXI, OCP and DTL. Our network interface has a modular architecture, which allows flexible instantiation. It provides both guaranteed and best-effort services via connections. These are configured via network interface ports using the network itself, instead of a separate control interconnect. An example instance of this network interface with 4 ports has an area of  $0.143\text{mm}^2$  in a  $0.13\mu\text{m}$  technology, and runs at 500 MHz.*

## 1 Introduction

Networks on chip (NoC) have been proposed as a solution to the interconnect problem for highly complex chips [2, 3, 5, 9, 12, 14, 15, 17, 21, 27]. NoCs help designing chips in several ways: they (a) structure and manage wires in deep submicron technologies [2, 3, 9, 12, 21], (b) allow good wire utilization through sharing [5, 9, 12, 21], (c) scale better than buses [14, 21], (d) can be energy efficient and reliable [2, 5], and (e) decouple computation from communication through well-defined interfaces, enabling IP modules and interconnect to be designed in isolation, and to be integrated more easily [2, 13, 21, 24].

Networks are composed of *routers*, which transport the data from one place to another, and *network interfaces* (NI), which implement the interface to the IP modules. In a previous article [21], we have shown the trade-offs in designing a cost-effective router combining guaranteed with best-effort traffic. In this paper, we focus on the other network component, the network interface.

Network interface design has received considerable attention for parallel computers [8, 25], and computer networks [6, 7]. These designs are optimized for performance (high throughput, low latency), and often consist of a dedicated processor, and large amount of buffering. As a consequence, their cost is too large to be applicable on chip.

On-chip network interfaces must provide a low-area overhead, because the size of IP modules attached to the NoC is relatively small. Designs of network interfaces with a low area have been proposed [4, 28]. However, they do not provide throughput or latency guarantees, which are essential for a compositional construction of complex SoCs.

Our NI is intended for systems on chip (SoC), hence, it must have a low area. To enable the reuse of existing IP modules, we must provide a smooth transition from buses to NoCs. A shared-

memory abstraction via transactions (e.g., read, write) ensures this. Further, we also have to provide a simple and flexible configuration, preferably using the NoC itself to avoid the need for a separate scalable interconnect.

We achieve a low-cost implementation of the NI by implementing the protocol stack in hardware, and by exploiting on-chip characteristics (such as the absence of transmission errors, relatively static configuration, tight synchronization) to implement only the relevant parts of a complete OSI stack. A hardware implementation of the protocol stack provides a much lower latency overhead compared to a software implementation. Further, a hardware implementation allows both hardware and software cores to be reused without change [4].

Our NI provides services at the transport layer in the ISO-OSI reference model [22], because this is the first layer where offered services are independent of the network implementation. This is a key ingredient in achieving the *decoupling between computation and communication* [16, 24], which allows IP modules and interconnect to be designed independently from each other. We provide transport-layer services by defining connections (e.g., point-to-point or multicast) configured for specific properties (e.g., throughput, ordering).

We offer *guaranteed services* as they are essential for a compositional construction (design and programming) of SoC. The reasons are that they limit the possible interactions of IPs with the communication environment [12, 13], separate the IP requirements and their implementation, and make application quality of service independent of the IP and NoC implementations. Examples of such guarantees are lower bounds on throughput, and upper bounds on latency.

Our NoC, called *Æthereal*, offers a *shared-memory abstraction* to the IP modules. Communication is performed using a transaction-based protocol, where master IP modules issue *request messages* (e.g., read and write commands at an address, possibly carrying data) that are executed by the addressed slave modules, which may respond with a *response message* (i.e., status of the command execution, and possibly data) [23]. We adopt this protocol to provide backward compatibility to existing on-chip communication protocols (e.g., AXI [1], OCP [18], DTL [19]), and also to allow future protocols better suited to NoCs.

We provide a modular NI, which can be configured at design time. This is, the number of ports and their type (i.e., configuration port, master port, or slave port), the number of connections at each port, memory allocated for the queues, the level of services per port, and the interface to the IP modules are all configurable at design (instantiation) time using an XML description [11].

The NI allows flexible NoC configuration at run time. Each connection can be configured individually, requiring configurable NoC components (i.e., router and NI). However, instead of using

a separate control interconnect to program them, the NoC is used to program itself. This is performed through configuration ports using DTL-MMIO (memory-mapped IO) transactions [19]. The NoC can be configured in a distributed fashion (i.e., via multiple configuration ports), or centralized (i.e., via a single port).

The paper is organized as follows. In the next section, the services that we implement, and the interface offered to the IP modules are described. In Section 3, we show that NoCs can be configured both in a distributed and in a centralized way, and we present the trade-offs between the two approaches. In Section 4, we present a modular network interface architecture, which is split into a kernel, providing core functionality, and a number of shells to extend functionality, e.g., wrappers to provide an interface to existing bus protocols, such as AXI or DTL. In this section, we also show how the NI allows NoC configuration using the NoC itself as opposed to via a separate control interconnect. In Section 5, we demonstrate the feasibility of our network interface design through a prototype implementation in a 0.13 $\mu$ m technology, and we conclude in Section 6.

## 2 NoC Services

As mentioned in the previous section, the communication services of the *Æthereal* NoC are defined to meet the following goals: (a) decouple computation (IP modules) from communication (NoC), (b) provide backward compatibility to existing bus protocols, (c) provide support for real-time communication, and (d) have a low-cost implementation.

Decoupling computation from communication is a key ingredient in managing the complexity of designing chips with billions of transistors, because it allows the IP modules and the interconnect to be designed independently [16, 24]. In NoCs, this decoupling is achieved by positioning the network services at the transport level [3, 21] or above in the ISO-OSI reference model [22]. At the transport level, the offered services are *end to end* between communicating IP modules, hiding, thus, the network internals, such as topology, routing scheme, etc.

Backward compatibility with existing protocols, such as AXI or DTL, is achieved by using a model based on *transactions* [23]. In a transaction-based model, there are two types of IP modules: masters and slaves. Masters initiate transactions by issuing requests, which can be further split in commands, and write data (corresponding to the address and write signal groups in AXI). Examples of commands are read, and write. One or more slaves receive and execute each transaction. Optionally, a transaction can also include a response issued by the slave to the master to return data or an acknowledgment of the transaction execution (corresponding to the read data and write response groups in AXI).

In the *Æthereal* NoC, all these signals are sequentialized in request and response *messages*, which are supplied to the NoC, where they are transported by means of *packets*. Sequentialization is performed to reduce the number of wires, increasing their utilization, and to simplify arbitration. Packetization is performed by the NI, and is thus transparent to the IP modules.

The *Æthereal* NoC offers its services on *connections*, which can be point to point (one master, one slave), multicast (one master, multiple slaves, all slaves executing each transaction), and narrowcast (one master, multiple slaves, a transaction is executed by only one slave) [23]. Connections are composed of unidirectional point-to-point *channels* (between a single master and a single slave). To each channel, *properties* are attached, such as guaranteed message delivery or not, in order or un-ordered message

delivery, and with or without timing guarantees. As a result, different properties can be attached to the request and response parts of a connection, or for different slaves within the same connection. Connections can be opened and closed at any time. Opening and closing of connections takes time, and is intended to be performed at a granularity larger than individual transactions.

Support for real-time communication is achieved by providing throughput, latency and jitter *guarantees*. In *Æthereal*, this is implemented by configuring connections as pipelined time-division-multiplexed circuits over the network. Time multiplexing is only possible when the network routers have a notion of synchronicity which allows slots to be reserved consecutively in a sequence of routers [13, 21]. This scheme [21] has smaller packet buffers, and, hence, has lower implementation cost compared to alternatives, such as rate-based packet switching [29], or deadline-based packet switching [20].

Throughput guarantees are given by the number of slots reserved for a connection. Slots correspond to a given bandwidth:  $B_i$ , and, therefore, reserving  $N$  slots for a connection results in a total bandwidth of  $N \times B_i$ . The latency bound is given by the waiting time until the reserved slot arrives and the number of routers data passes to reach its destination. Jitter is given by the maximum distance between two slot reservations.

Protocol stacks that are used in networks to implement communication services, require additional cost compared to buses. Protocol stacks are necessary in networks to manage the complexity of networks, and to offer differentiated services. The pressure to keep the protocol stack small is higher on-chip than off-chip, because the size of the IP modules attached to the NoC is relatively small. However, for NoCs, the protocol stacks can be reduced by exploiting the on-chip characteristics (e.g., no transfer errors, short wires) [23]. In the *Æthereal* NoC, we optimize the performance and minimize the cost of the protocol stack by implementing it in hardware, rather than in software. We support this claim in Section 5.

## 3 Network Configuration

Before the *Æthereal* NoC can be used by an application, it must be configured. NoC (re)configuration means opening and closing connections in the system. Connections are set up depending on the application or the mode the system is running. Therefore, we must be able to open and close connections while the system is running. (Re)configuration can be partial or total (some or all connections are opened/closed, respectively).

Opening a connection involves setting several registers, and allocating shared resources (for more details see Section 4). In the case of the current prototype of the *Æthereal* NoC, for each pair of one master and one slave of a connection, there are 5 and 3 registers written at the master and slave network interfaces, respectively. The shared resources consist of the slots allocated to the connections. These slots can be configured using either a distributed or a centralized model.

In the distributed case, a connection can be opened/closed from multiple network interface ports. Multiple configuration operations can be performed simultaneously, however, potential conflicts must also be solved (e.g., connection configurations initiated at two configuration ports may try to reserve the same slot in a router). Information about the slots is maintained in the routers, which also accept or reject a tentative slot allocation.

In a centralized system, there is only one place that performs NoC configuration. In such a case, the slot information can be

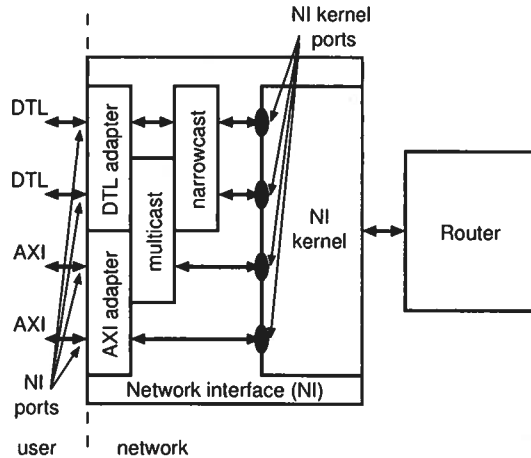


Figure 1. NI kernel and shells

stored in the configuration module instead of the routers, which simplifies the design, and, in the case of small NoCs, may even speed up configuration. For large NoCs, however, centralized configuration can introduce a bottleneck.

In the initial prototype of the Æthereal NoC, we opt for centralized configuration, because it is able to satisfy the needs of a small NoC (around 10 routers), and has a simpler design and lower cost. We use transactions to program the NoC, both for connection registers in the NIs, and for the slot information. We present details of how NoC configuration is performed in Section 4.

## 4 Network Interface Architecture

The network interface (NI) is the component that provides the conversion of the packet-based communication of the NoC to the higher-level protocol that IP modules use. We split the design of the network interface in two parts (see Figure 1): (a) the *NI kernel*, which implements the channels, packetizes messages and schedules them to the routers, implements the end-to-end flow control, and the clock domain crossing, and (b) the *NI shells*, which implement the connections (e.g., narrowcast, multicast), transaction ordering for connections, and other higher-level issues specific to the protocol offered to the IP.

### 4.1 NI Kernel Architecture

The NI kernel (see Figure 2) receives and provides messages, which contain the data provided by the IP modules via their protocol after sequentialization. The message structure may vary depending on the protocol used by the IP module. However, the message structure is irrelevant for the NI kernel, as it just sees messages as pieces of data to be transported over the NoC.

The NI kernel communicates with the NI shells via *ports*. At each port, point-to-point connections can be configured, their maximum number being selected at NI instantiation time. A port can have multiple connections to allow differentiated traffic classes, in which case there are also *connid* signals to select on which connection a message is supplied or consumed.

In the NI kernel, there are two message queues for each point-to-point connection (one source queue, for messages going to the NoC, and one destination queue, for messages coming from the NoC). Their size is also selected at the NI instantiation time. In our NI, queues are implemented using custom-made hardware fifos,

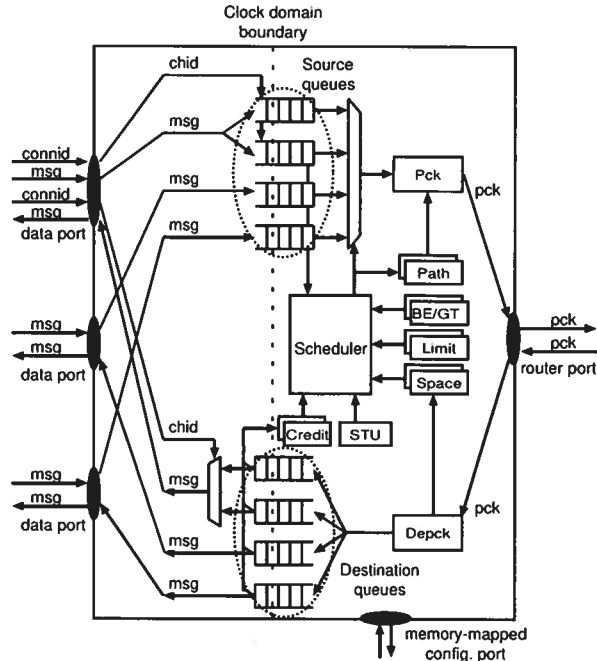


Figure 2. Network interface kernel

and are also used to provide the clock domain crossing between the network and the IP modules. Each port can, therefore, have a different clock frequency.

Each channel is configured individually. In a first prototype of the Æthereal NI, we can configure if a channel provides time guarantees (GT) or not (we call this best effort, BE), reserve slots for GT connections, configure the end-to-end flow control, and the routing information.

End-to-end flow control ensures that no data is sent unless there is enough space in the destination buffer to accommodate it. This is implemented using credits [26]. For each channel, there is a counter (*Space*) tracking the empty buffer space of the remote destination queue. This counter is initialized with the remote buffer size. When data is sent from the source queue, the counter is decremented. When data is consumed by the IP module at the other side, credits are produced in a counter (*Credit*) to indicate that more empty space is available. These credits are sent to the producer of data to be added to its *Space* counter. In the Æthereal prototype, we piggyback credits in the header of the packets for the data in the other direction to improve NoC efficiency. Note that at most *Space* data items can be transmitted before credits are received. We call the minimum between the data items in the queue and the value in the counter *Space*, the *sendable data*.

From the source queues, data is packetized (*Pck*) and sent to the NoC via a single link. A packet header consists of the routing information (NI address for destination routing, and path for source routing), remote queue id (i.e., the queue of the remote NI in which the data will be stored), and piggybacked credits.

There are multiple channels which may require data transmission, we implement a scheduler to arbitrate between them. The scheduler checks if the current slot is reserved for a GT channel. If the slot is reserved, is the GT channel has data which can be transmitted, and if there is space in the channel's destination buffer, then the channel is granted data transmission. Otherwise,

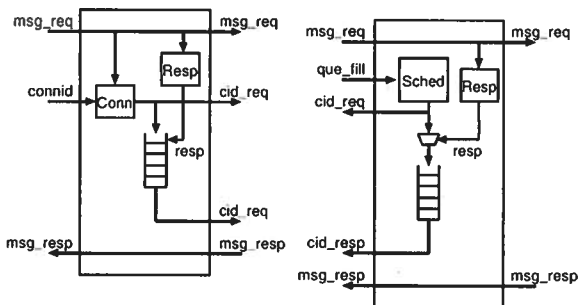


Figure 3. Narrowcast shell      Figure 4. Multi-connection shell

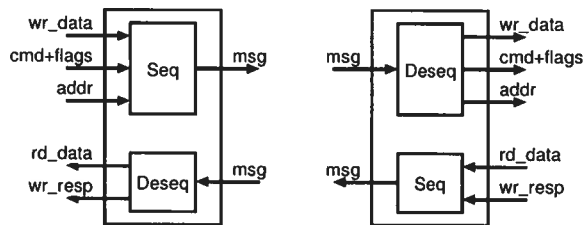


Figure 5. Master shell      Figure 6. Slave shell

the scheduler selects a BE channel with data and remote space using some arbitration scheme: e.g. round-robin, weighted round-robin, or based on the queue filling.

To optimize the NoC utilization, it is preferable to send longer packets. To achieve this, we implemented a configurable threshold mechanism, which skips a channel as long as the sendable data is below the threshold. This is applicable for both BE and GT channels. To prevent starvation at user/application level (e.g., due to write data being buffered indefinitely on which the IP module waits for an acknowledge), we also provide a *flush* signal for each channel (and a bit in the message header) to temporarily override the threshold. When the flush signal is high for a cycle, a snapshot of its source queue filling is taken, and as long as all the words in the queue at the time of flushing have not been sent, the threshold for that queue is bypassed.

A similar threshold is set for credit transmission. The reason is that, when there is no data on which the credits can be piggybacked, the credits are sent as empty packets, thus, consuming extra bandwidth. To minimize the bandwidth consumed by credits, a credit threshold is set, which allows credits to be transmitted only when their sum is above the threshold. Similarly to the data case, to prevent possible starvation, we provide a flush signal to force credits to be sent even when they are below their threshold.

As credits are piggybacked on packets, a queue becomes eligible for scheduling when either the amount of sendable data are above a first threshold, or when the amount of credits is above a second threshold. However, once a queue is selected, a packet containing the largest possible amount of credits and data will be produced. Note the amount of credits is bound by implementation to the given number of bits in the packet header, and packet have a maximum length to avoid links being used exclusively by a packet/channel, which would cause congestion.

On the outgoing path, packets are depacketized, credits are added to the counter Space, and data is stored in its corresponding queue, which is given by a queue id field in the header.

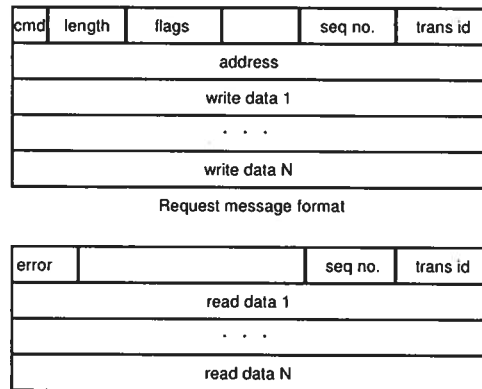


Figure 7. Message format examples

## 4.2 NI Shells Architectures

With the NI kernel described in the previous section, point-to-point connections (i.e., between one master and one slave) can be supported directly. These type of connections are useful in systems involving chains of modules communicating point to point with one another (e.g., video pixel processing [10]).

For more complex types of connections, such as narrowcast or multicast, and to provide conversions to other protocols, we add shells around the NI kernel. As an example, in Figure 1, we show a NI with two DTL and two AXI ports. All ports provide point-to-point connections. In addition to this, the two DTL ports provide narrowcast connections, and one DTL and one AXI port provide multicast connections. Note that these shells add specific functionality, and can be plugged in or left out at design time according to the requirements. NoC instantiation is simple, as we use an XML description to automatically generate the VHDL code for the NIs as well as for the NoC topology.

In Figure 3, we show an example of a narrowcast shell. Narrowcast connections are connections between one master and several slaves, where each transaction is executed by a single slave selected based on the address provided in the transaction [23]. Narrowcast connections provide a simple, low-cost solution for a single shared address space mapped on multiple memories. It implements the splitting/merging of data going to/coming from these memories.

We implement the narrowcast connection as a collection of point-to-point connections, one for each master-slave pair. Within a narrowcast connection, the slave for which the transaction is destined is selected based on the address (Conn block). The address range assigned to a slave is configurable in the narrowcast module. To provide in-order response delivery, the narrowcast must also keep a history of connection identifiers of the transactions including responses (e.g., reads, and acknowledged writes), and the length of these responses. In-order delivery per slave of request messages is already provided by the point-to-point connections.

When a slave using a connectionless protocol (e.g., DTL) is connected to a NI port supporting multiple connections, a multi-connection shell must be included to arbitrate between the connections. A multi-connection shell (see Figure 4) includes a scheduler to select connections from which messages are consumed, based e.g., on their filling. As for the narrowcast, the multi-connection shell has a connection id history for scheduling the responses.

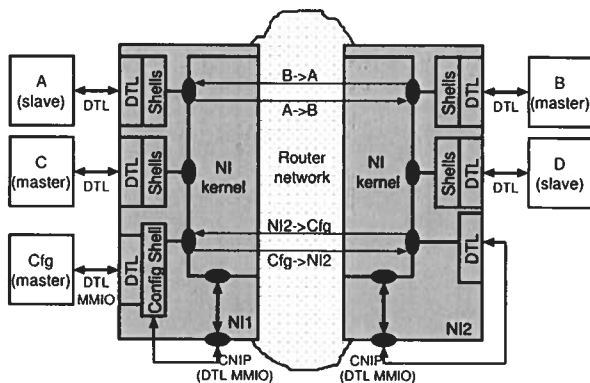


Figure 8. NI configuration

In Figures 5 and 6, we show a master and slave shells that implement a simplified version of a protocol such as AXI. The basic functionality of such a shell is to sequentialize commands and their flags, addresses, and write data in request messages, and to desequentialize messages into read data, and write responses. Examples of the message structures (i.e., after sequentialization) passing from NI shells and NI kernel are shown in Figure 7. In full-fledged master and slave shells, more blocks would be added to implement e.g., the unbuffered writes at the master side, and read linked, write conditional at the slave side.

### 4.3 NI Configuration

As mentioned in Section 3, in our prototype  $\text{\AA}$ ethereal NoC, we opt for centralized configuration. This means that there is a single configuration module that configures the whole NoC, and that slot tables can be removed from the routers. Consequently, only the NIs need to be configured when opening/closing connections.

NIs are configured via a configuration port (CNIP), which offers a memory-mapped view on all control registers in the NIs. This means that the registers in the NI are readable and writable by any master using normal read and write transactions.

Configuration is performed using the NoC itself (i.e., there is no separate control interconnect needed for NoC configuration). Consequently, the CNIPs are connected to the NoC like any other slave (see CNIP at NI2 in Figure 8). At the configuration module Cfg's NI, we introduce a configuration shell (*Config Shell*), which, based on the address configures the local NI (NI1), or sends configuration messages via the NoC to other NIs. The configuration shell optimizes away the need for an extra data port at NI1 to be connected to the NI1's CNIP.

In Figure 9, we show the necessary steps in setting up a connection between two modules (master B and slave A) from a configuration module (Cfg). Like for any other memory-mapped register, before sending configuration messages for configuring the B to A connection at NI2, a connection from Cfg to NI2's CNIP must be set up. This connection is opened in two steps corresponding to the request and response channels. First, the request channel to the NI2's CNIP is set up by writing the necessary registers in NI1 (Step 1 in Figure 9). Second, we use this channel to set up (via the NoC) the response channel from NI2's CNIP to Cfg (Step 2). The three shown messages are delivered and executed in order at NI2. The last of them also requests an acknowledgment message to confirm that the channel has been successfully set up.

After the configuration connection has been set up, the remote NI2 can be configured to set up a channel from B to A. For config-

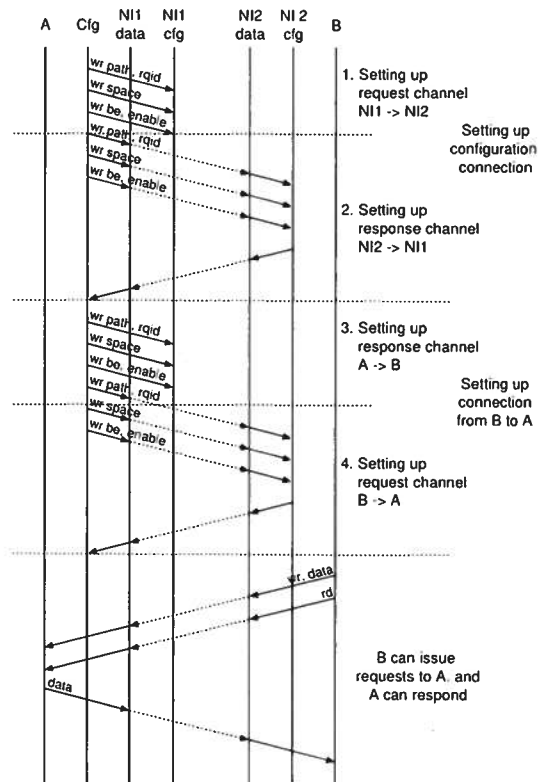


Figure 9. Connection configuration example

uring NI2 (B's NI), the previously set up configuration connection is used. For configuring NI1, the NI1's configuration port is accessed directly via *Config Shell*. First, the channel from the slave module A to the master module B is configured at NI1 (Step 3). Second, the channel from the master module B to the slave module A is configured (Step 4) through messages to NI2.

## 5 Implementation

In the previous section, we describe a prototype of a configurable NI architecture. In this section, we discuss the synthesized area and speed figures for the network interface components: NI kernel, narrowcast, multichannel and configuration shells, and master and slave shells for a simplified version of DTL.

We have synthesized an instance of a NI kernel with a STU of 8 slots, and 4 ports having 1, 1, 2, and 4 channels, respectively, with all queues being 32-bit wide and 8-word deep. The queues are area-efficient custom-made hardware fifos. We use these fifos instead of RAMs, because we need simultaneous access at all NI ports (possibly running at different speeds) as well as simultaneous read and write access for incoming and outgoing packets, which cannot be offered with a single RAM. Finally, for the small queues needed in the NI, multiple RAMs have a too large area overhead. Moreover, the hardware fifos implement the clock domain boundary allowing each NI port to run at a different clock frequency. The router side of the NI kernel runs at a frequency of 500 MHz, which matches our prototype router frequency [21], and delivers a bandwidth toward the router of 16 Gbit/s in each direction. The synthesized area for this NI-kernel instance is  $0.11 \text{ mm}^2$  in a  $0.13 \mu\text{m}$  technology.

Narrowcast and multi-connection shells have an area of  $0.004 \text{ mm}^2$  and  $0.007 \text{ mm}^2$ , corresponding to 4% and 6% of the NI kernel area. The DTL shells are very small,  $0.005 \text{ mm}^2$  and  $0.002 \text{ mm}^2$  for the master and slave ports, corresponding to 5% and 2% of the NI kernel area, respectively. (This is also due to the fact that not all of the DTL functionality has been implemented). The configuration shell, which provides a simplified DTL-MMIO interface to configure the NoC, has an area of  $0.01 \text{ mm}^2$ .

Summing up, for an example NI with 4 ports, one for configuration (one channel to which the configuration shell is attached), two masters (one offering narrowcast), and one slave (multichannel), the total area is  $0.11 + 0.01 + 2 \times 0.005 + 0.004 + 0.002 + 0.007 = 0.143 \text{ mm}^2$ .

The latency introduced by our current NI is 2 cycles in the DTL master shell (due to sequentialization, as part of packetization), 0 to 2 in the narrowcast and multicast shells (depending on the NI instance), and between 1 and 3 cycles in the NI kernels (as data needs to be aligned to a 3 word flit boundary), and 2 clock cycles for clock domain crossing. Additional delay is caused by the arbitration, but we do not include this in the NI latency overhead, as it needs to be performed anyway (also in the case of a bus, arbitration is performed).

The resulting latency overhead introduced by our NI is between 4 and 10 cycles, which is pipelined to maximize throughput. The latency overhead of a software implementation of the protocol is much larger (e.g., 47 instructions for packetization only [4]). A hardware implementation allows both legacy software and hardware task implementations to be used without change.

## 6 Conclusions

In this paper, we describe a network interface architecture which offers high-level services at a low cost. Our network interface provides a shared-memory abstraction, where communication is performed using read/write transactions. We offer, via connections, high-level services, such as transaction ordering, throughput and latency guarantees, and end-to-end flow control. These connections are configurable at runtime via a memory-mapped configuration port. We use the network to configure itself as opposed to using a separate control interconnect for network configuration.

Our network interface has a modular design, composed of kernel and shells. The NI kernel provides the basic functionality, including arbitration between channels, transaction ordering, end-to-end flow control, packetization, and a link protocol with the router. Shells implement (a) additional functionality, such as multicast and narrowcast connections, and (b) adapters to existing protocols, such as AXI or DTL. All these shells can be plugged in or left out at design time according to the needs. This is done using an XML description of the network, which is used to automatically generate the VHDL code for the network interfaces, as well as for the network topology.

We show an instance of our network interface, which shows that the cost of implementing our protocol stack in hardware is small ( $0.143 \text{ mm}^2$  in a  $0.13 \mu\text{m}$  technology, running at 500 MHz). Our hardware protocol stack implementation provides a very low protocol overhead of 4 to 6 cycles, which is much lower than a software stack implementation.

In conclusion, we provide an efficient network interface offering a shared-memory abstraction, high-level services (including guarantees), which allows runtime network configuration using the network itself.

## References

- [1] ARM. *AMBA AXI Protocol Specification*, June 2003.
- [2] L. Benini and G. De Micheli. Powering networks on chips. In *Proc. ISSS*, 2001.
- [3] L. Benini and G. De Micheli. Networks on chips: A new SoC paradigm. *IEEE Computer*, 35(1):70–80, 2002.
- [4] P. Bhojwani and R. Mahapatra. Interfacing cores with on-chip packet-switched networks. In *Proc. VLSI Design*, 2003.
- [5] E. Bolotin et al. QNoC: QoS architecture and design process for network on chip. *Journal of Systems Architecture*, 49, Dec. 2003.
- [6] T. Callahan and S. C. Goldstein. NIFDY: A low overhead, high throughput network interface. In *Proc. ISCA*, 1995.
- [7] A. Chien et al. Design challenges for high-performance network interfaces. *IEEE Computer*, 31(11):42–44, 1998.
- [8] D. J. Culler et al. *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufmann Publishers, 1999.
- [9] W. J. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *Proc. DAC*, 2001.
- [10] O. P. Gangwal et al. Understanding video pixel processing applications for flexible implementations. In *Proc. Euromicro DSD*, 2003.
- [11] S. Gonzalez Pestana et al. Cost-performance trade-offs in networks on chip: A simulation based approach. In *Proc. DATE*, 2004.
- [12] K. Goossens et al. Networks on silicon: Combining best-effort and guaranteed services. In *Proc. DATE*, 2002.
- [13] K. Goossens et al. Guaranteeing the quality of services in networks on chip. In J. Nurmi, H. Tenhunen, J. Isoaho, and A. Jantsch, editors, *Networks on Chip*, pages 61–82. Kluwer, 2003.
- [14] P. Guerrier and A. Greiner. A generic architecture for on-chip packet-switched interconnections. In *Proc. DATE*, 2000.
- [15] F. Karim et al. An interconnect architecture for networking systems on chip. *IEEE Micro*, 22(5), 2002.
- [16] K. Keutzer et al. System-level design: Orthogonalization of concerns and platform-based design. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 19(12):1523–1543, 2000.
- [17] S. Kumar et al. A network on chip architecture and design methodology. In *Proc. ISVLSI*, 2002.
- [18] OCP International Partnership. *Open Core Protocol Specification. 2.0 Release Candidate*, 2003.
- [19] Philips Semiconductors. *Device Transaction Level (DTL) Protocol Specification. Version 2.2*, July 2002.
- [20] J. Rexford. *Tailoring Router Architectures to Performance Requirements in Cut-Through Networks*. PhD thesis, Univ. Michigan, 1999.
- [21] E. Rijpkema et al. Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip. In *Proc. DATE*, 2003.
- [22] M. T. Rose. *The Open Book: A Practical Perspective on OSI*. Prentice Hall, 1990.
- [23] A. Rădulescu and K. Goossens. Communication services for networks on chip. In S. Bhattacharyya, E. Deprettere, and J. Teich, editors, *Domain-Specific Embedded Multiprocessors*. Dekker, 2003.
- [24] M. Sgroi et al. Addressing the system-on-a-chip interconnect woes through communication-based design. In *Proc. DAC*, 2001.
- [25] P. Steenkiste. A high-speed network interface for distributed-memory systems: Architecture and applications. *ACM Trans. on Computer Systems*, 15(1):75–109, 1997.
- [26] A. S. Tanenbaum. *Computer Networks*. Prentice Hall, 1996.
- [27] D. Wiklund and D. Liu. Socbus: switched network on chip for hard real time embedded systems. In *Proc. IPDPS*, 2003.
- [28] C. A. Zeferino, M. E. Kreutz, L. Carro, and A. A. Susin. A study on communication issues for systems-on-chip. In *Proc. SBCCI*, 2002.
- [29] H. Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proc. of the IEEE*, 83(10):1374–1396, 1995.

# Exhibit H

## Communication Services for Networks on Chip

Andrei Rădulescu and Kees Goossens

Philips Research Laboratories, Eindhoven, The Netherlands

Email: andrei.radulescu|kees.goossens@philips.com

### Abstract

Networks are emerging as a possible solution for future on-chip interconnects. In this chapter, we show how networks on chip (NoC) are similar to and differ from both off-chip networks (e.g., computer networks) and current on-chip interconnects (e.g., buses). We re-examine their communication services in the context of NoCs. To enable a clean separation between the NoC and IP blocks, we provide services that abstract from network implementations. We define a request-response transaction model similar to bus protocols, to make our approach backward compatible. To exploit the full power of NoCs, we also provide connection-oriented communication with differentiated services. Examples are bandwidth guarantees, transaction orderings, and end-to-end flow control.

*Key Words:* Networks on chip, on-chip buses, computer networks, communication services, protocol stack, transaction, connection

## I. Introduction

Networks on chip (NoC) have received considerable attention recently as a solution to the interconnect problem in highly-complex chips [3–5, 7–9, 15, 20, 23]. The reason is twofold. First, NoCs help resolve the electrical problems in new deep-submicron technologies, as they structure and manage global wires [3–5, 7, 8]. At the same time they share wires, lowering their number and increasing their utilization [7, 8]. NoCs can also be energy efficient and reliable [4], and are scalable compared to buses [9]. Second, NoCs also decouple computation from communication, which is essential in managing the design of billion-transistor chips [14, 23]. NoCs achieve this decoupling because they are traditionally designed using protocol stacks [22], which provide well-defined interfaces separating communication service usage from service implementation [5, 23].

Using networks for on-chip communication when designing systems on chip (SoC), however, raises a number of new issues that must be taken into account. This is because, in contrast to existing on-chip interconnects (e.g., buses, switches, or point-to-point wires), where the communicating modules are directly connected, in a NoC the modules communicate remotely via network nodes. As a result, interconnect arbitration changes from centralized to distributed, and issues like out-of order transactions, higher latencies, and end-to-end flow control must be handled either by the intellectual property block (IP) or by the network.

Most of these topics have been already the subject of research in the field of computer networks [25] and parallel machine interconnect networks [6]. However, on-chip networks have different properties (e.g., tighter link synchronization) and constraints (e.g., higher memory cost) leading to different design choices, which ultimately affect the network services.

In this chapter, we compare NoCs and off-chip networks showing both their similarities and differences. We also explore the differences between NoCs and existing on-chip interconnects. We present an interface which takes these issues into consideration. Our interface is aimed at being similar to a split-transaction bus interface, such as VCI [26], OCP [17] or DTL [18], to allow simple, low-cost wrappers to bus interfaces, and to allow backward compatibility with existing IPs. Our interface uses a request-response protocol that provides basic read and write operations. But it

extends bus interfaces to fully exploit the power of our NoC [8, 20, 21]. For example, it offers connection-based communication where end-to-end flow control and time-related guarantees (e.g., bounded latency) can be requested.

The chapter is organized as follows. In the next two sections we compare NoC properties with those of off-chip networks and buses, respectively. In Section IV, we define the services that we offer in our network. Finally, we present our conclusions.

## II. Networks Brought on Chip

Networks have been the subject of research for decades, both in the context of local and wide area networks (computer networks) [25], and as an interconnect for parallel machines [6]. Both are very much related to on-chip networks, and many of the results in those fields are also applicable on chip. However, NoC premises are different from off-chip networks, and, therefore, most of the network design choices must be re-evaluated.

NoCs differ from off-chip networks mainly in their constraints and synchronization. Typically, resource constraints are tighter on chip than off chip. Storage (i.e., memory) and computation resources are relatively more expensive, whereas the number of point-to-point links is larger on chip than off chip [7].

Storage is expensive, because general-purpose on-chip memory, such as RAMs, occupy a large area. Having the memory distributed in the network components in relatively small sizes is even worse, as the overhead area in the memory then becomes dominant.

For on-chip networks, computation too comes at a relatively high cost compared to off-chip networks. An off-chip network interface usually contains a dedicated processor to implement the part of the protocol stack, to relieve the host processor from the communication processing. Including a dedicated processor in a network interface may be not feasible on chip, as the size of the network interface will become comparable to or larger than the IP to be connected to the network. Moreover, running the protocol stack on the IP itself may also be not feasible, because often these IPs have one dedicated function only, and do not have the capabilities to

run a network protocol stack. A cost-effective solution would be to have a dedicated-hardware implementation of the protocol stack.

The number of wires and pins to connect network components is an order of magnitude larger on chip than off chip [7]. If they are not used massively for other purposes than NoC communication, they allow wide point-to-point interconnects (e.g., 300-bit links) [7,15]. This is not possible off-chip, where links are relatively narrower: 8-16 bits.

On-chip wires are also relatively shorter than off chip [7], allowing a much tighter synchronization than off chip. This allows a reduction in the buffer space in the routers because the communication can be done at a smaller granularity. In the current semiconductor technologies, wires are also fast and reliable, which allows simpler link-layer protocols (e.g., no need for error correction, or retransmission). This also compensates for the lack of memory and computational resources.

In the rest of the section, we list five network issues that have a direct impact on the NoC cost: reliable communication, deadlock, data ordering, network flow control and buffering strategy, and time-related guarantees. For each of them, we discuss the differences and similarities for on- and off-chip networks.

**Reliable communication.** A consequence of the tight on-chip resource constraints is that the network components (i.e., routers and network interfaces) must be fairly simple to minimize computation and memory requirements. Luckily, on-chip wires currently provide a reliable communication medium, which can help to avoid the considerable overhead incurred by off-chip networks for providing reliable communication. Data integrity can be provided at low cost at the data link layer. However, data loss also depends on the network architecture. In most computer networks data is simply dropped if congestion occurs in the network [6,25]. On-chip, dropping data may lead to a too costly implementation of reliable communication. We show below that a network that does not drop data can be a much lower-cost solution, at the peril of introducing the possibility of deadlock.

**Deadlock.** Computer network topologies have generally an irregular (possibly dynamic) structure, which can introduce buffer cycles. In such

topologies, packet dropping at the network nodes may be required to avoid deadlocks.

Deadlock can also be avoided without dropping data, for example by introducing constraints either in the topology or routing. Fat-tree topologies have already been considered for NoCs, where deadlock is avoided by bouncing back packets in the network in case of buffer overflow [9]. Tile-based approaches to system design [7, 15, 24] use mesh or torus network topologies, where deadlock can be avoided using, for example, a turn-model routing algorithm [6].

An alternative solution for deadlock in NoCs, which takes into consideration that modules connecting to the network are either masters (initiating requests and receiving responses), or slaves (receiving requests and sending back responses), is to maintain separate virtual networks (with separate buffers) for requests and responses [6].

**Data ordering.** In a network, data sent from a source to a destination may arrive out of order due to reordering in network nodes, following different routes, or retransmission after dropping. For off-chip networks out-of-order data delivery is typical. However, for NoCs where no data is dropped, data can be forced to follow the same path between a source and a destination (deterministic routing) with no reordering. This in-order data transportation requires less buffer space, and reordering modules are no longer necessary.

**Network flow control and buffering strategy.** Network flow control and buffering strategy have a direct impact on the memory utilization in the network. Wormhole routing requires only a flit buffer (per queue) in the router, whereas store-and-forward and virtual-cut-through routing require at least the buffer space to accommodate a packet [6]. Consequently, on chip, wormhole routing may be preferred over virtual-cut-through or store-and-forward routing. Similarly, input queuing may be a lower memory-cost alternative to virtual-output-queuing or output-queuing buffering strategies, because it has fewer queues. Dedicated (lower cost) fifo memory structures also enable on-chip usage of virtual-cut-through routing or virtual output queuing for a better performance [20]. However, using virtual-cut-through routing and virtual output queuing at

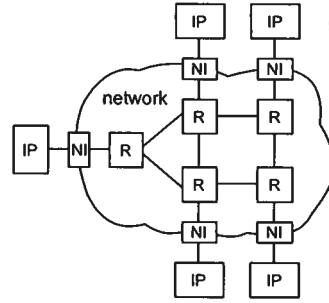


**Figure 1.** A network interconnect example

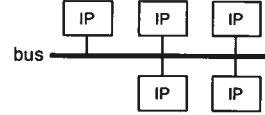


**Figure 2.** A bus interconnect example

the same time is still too costly [20].

**Time-related guarantees.** Off-chip networks typically use packet switching and offer best-effort services. Contention can occur at each network node, making latency guarantees very hard to offer. Throughput guarantees can still be offered using schemes such as rate-based switching [27] or deadline-based packet switching [19], but with high buffering costs.

An alternative to provide such time-related guarantees is to use time-division multiple access (TDMA) circuits, where every circuit is dedicated to a network connection. Circuits provide guarantees at a relatively low memory and computation cost. Network resource utilization is increased when the network architecture allows any left-over guaranteed bandwidth to be used by best-effort communication [10, 20, 21].

### III. From buses to NoCs

Introducing networks (Figure 1) as on-chip interconnects radically changes the communication as compared to direct interconnects such as buses or switches (Figure 2). This is because of the multi-hop nature of a network, where communication modules are not directly connected, but separated by one or more network nodes. This is in contrast with the prevalent existing interconnects (i.e., buses) where modules are directly connected. The implications of this change reside in the arbitration (which

must change from centralized to distributed), and in the communication properties (e.g., ordering, or flow control).

In this section, we list some of these topics, and outline the differences between NoCs and buses. We refer mainly to buses as direct interconnects, because currently they are the most used on-chip interconnect. Most of the bus characteristics also hold for other direct interconnects (e.g., switches [16]). Multilevel buses are a hybrid between buses and NoCs. For our purposes, depending on the functionality of the bridges, multilevel buses behave either like simple buses [2] or like NoCs.

**Programming Model.** The programming model of a bus typically consists of load and store operations which are implemented as a sequence of primitive bus transactions. Bus interfaces typically have dedicated groups of wires for command, address, write data, and read data [1, 12, 13, 17, 18, 26].

A bus is a resource shared by multiple IPs. Therefore, before using it, IPs must go through an arbitration phase, where they request access to the bus, and block until the bus is granted to them.

A bus transaction involves a request and possibly a response. Modules issuing requests are called masters, and those serving requests are called slaves. If there is a single arbitration for a request-response pair, the bus is called non-split. In this case, the bus remains allocated to the master of the transaction until the response is delivered, even when this takes a long time. Alternatively, in a split bus, the bus is released after the request to allow transactions from different masters to be initiated. However, a new arbitration must be performed for the response such that the slave can access the bus [11].

For both split and non-split buses, both communication parties have direct and immediate access to the status of the transaction. In contrast, network transactions are one-way transfers from an output buffer at the source to an input buffer at the destination that causes some action at the destination, the occurrence of which is not visible at the source [6]. The effects of a network transaction are observable only through additional transactions. A request-response type of operation is still possible, but requires at least two distinct network transactions. Thus, a bus-like transaction in a NoC will essentially be a split transaction.

**Transaction Ordering.** Traditionally, all transactions on a bus are ordered (cf. Peripheral VCI [26], AMBA [1], DTL [18], or CoreConnect PLB and OPB [12, 13]). This is possible at a low cost, because the interconnect, being a direct link between the communicating parties, does not reorder data. However, on a split bus, a total ordering of transactions on a single master may still cause performance penalties, when slaves respond at different speeds. To solve this problem, recent extensions to bus protocols allow transactions to be performed on connections. Ordering of transactions within a connection is still preserved, but between connections there are no ordering constraints (e.g., OCP [17], or Basic VCI [26]). A few of the bus protocols allow out-of-order responses per connection in their advanced modes (e.g., Advanced VCI [26]), but both requests and responses arrive at the destination in the same order as they were sent.

In a NoC, ordering becomes weaker. Global ordering can only be provided at a very high cost due to the conflict between the distributed nature of the networks, and the requirement of a centralized arbitration necessary for global ordering.

Even local ordering, between a source-destination pair, may be costly. Data may arrive out of order if it is transported over multiple routes. In such cases, to still achieve an in-order delivery, data must be labeled with sequence numbers and reordered at the destination before being delivered.

**Atomic Chains of Transactions.** An atomic chain of transactions is a sequence of transactions initiated by a single master that is executed on a single slave exclusively. That is, other masters are denied access to that slave, once the first transaction in the chain claimed it. This mechanism is widely used to implement synchronization mechanisms between master modules (e.g., semaphores).

On a bus, atomic operations can easily be implemented, as the central arbiter will either (a) lock the bus for exclusive use by the master requesting the atomic chain, or (b) not granting access to a locked slave. In the former case, the duration resources are locked is shorter because once a master has been granted access to a bus, it can quickly perform all the transactions in the chain (no arbitration delay is required for the subsequent transactions in the chain). Consequently, the locked slave and the bus can be opened up again in a short time. This approach is used in AMBA and CoreConnect.

In the latter case, the bus is not locked, and can still be used by other modules, however, at the price of a longer locking duration of the slave. This approach is used in VCI and OCP.

In a NoC, where the arbitration is distributed, masters do not know that a slave is locked. Therefore, transactions to a locked slave may still be initiated, even though the locked slave cannot accept them. Consequently, to prevent deadlock, these other transactions must be either dropped, or transactions in the atomic chain must be able to bypass them to be served. Moreover, the duration a module is kept locked is much longer in case of NoCs, because of the higher latency per transaction.

**Deadlock.** In buses, deadlock is generally not an issue. Deadlock can still occur at the application level (e.g., an atomic chain of transactions that locks the bus, but never unlocks it), but this is not caused by the interconnect itself.

In a network, deadlock becomes a more important issue, and special care has to be taken in the network design to avoid deadlock. Deadlock is mainly caused by cycles in the buffers. To avoid deadlock, either network nodes must drop packets when their buffers are filled, or routing must be cycle-free. In a NoC, we believe the latter is preferable, because of its lower cost in achieving reliable communication (see Section II).

A second cause of deadlock are atomic chains of transactions. The reason is that while a module is locked, the queues storing transactions may get filled with transactions outside the atomic transaction chain, blocking the access of the transaction in the chain to reach the locked module. If atomic transaction chains must be implemented (to be compatible with processors allowing this, such as MIPS), the network nodes should be able to filter the transactions in the atomic chain, or be allowed to drop those blocking them.

**Media Arbitration.** An important difference between buses and NoCs is in the medium arbitration scheme. In a bus, master modules request access to the interconnect, and the arbiter grants the access for the whole interconnect at once. Arbitration is *centralized* as there is only one arbiter component. It is also *global* as all the requests as well as the state of the interconnect are visible to the arbiter. Moreover, when a grant is

given, the complete path from the source to the destination is exclusively reserved.

In a non-split bus, arbitration takes place once when a transaction is initiated. As a result, the bus is granted for both request and response. In a split bus, requests and responses are arbitrated separately.

In a NoC arbitration is also necessary, as it is a shared interconnect. However, in contrast to buses, the arbitration is *distributed*, because it is performed in every router, and is based only on *local* information. Arbitration of the communication resources (links, buffers) is performed incrementally as the request or response advances [20].

**Destination Name and Routing.** For a bus, the command, address, and data are broadcasted on the interconnect. They arrive at every destination, only one of which activates based on the broadcasted address, and executes the requested command. This is possible because all modules are directly connected to the same bus.

In a NoC, it is not feasible to broadcast information to all destinations, because it must be copied to all routers and network interfaces. This floods the network with data. The address is better decoded at the source to find a route to the destination module. A transaction address has, therefore, two parts: (a) a destination identifier, and (b) an internal address at the destination.

**Latency.** Transaction latency is caused by two factors: (a) the access time to the bus, which is the time until the bus is granted, and (b) the latency introduced by the interconnect to transfer the data.

For a bus, where the arbitration is centralized, the access time is proportional to the number of masters connected to the bus. The transfer latency itself typically is constant and relatively low, because the modules are linked directly. However, the speed of transfer is limited by the bus speed, which is relatively low.

In a NoC, arbitration is performed at each router for the following link. The access time per router is small. Both end-to-end access time and transport time increase proportionally to the number of hops between master and slave. However, network links are unidirectional and point to point and, hence, can run at higher frequencies than buses, thus lowering the

latency.

From a latency prospective, using a bus or a network is a trade off between the number of modules connected to the interconnect (which affects access time), the speed of the interconnect, and the network topology.

**Data Format.** In most modern bus interfaces the data format is defined by separate wire groups for the transaction type, address, write data, read data, and return acknowledgments/errors (e.g., VCI, OCP, AMBA, DTL, or CoreConnect). This is used to pipeline transactions. For example, concurrently with sending the address of a read transaction, the data of a previous write transaction can be sent, and the data from an even earlier read transaction can be received. Moreover, having dedicated wire groups simplifies the transaction decoding; there is no need for a mechanism to select between different kinds of data sent over a common set of wires.

Inside a network, there is typically no distinction between different kinds of data. Data is treated uniformly and passed from one router to another. This is done to minimize the control overhead and buffering in routers. If separate wires would be used for each of the above-mentioned groups, separate routing, scheduling and queuing would be needed, and the cost of routers would increase proportionally.

In addition, in a network at each layer in the protocol stack, control information must be supplied together with the data (e.g., packet type, network address, or packet size). This control information is organized as an envelope around the data. That is, first a header is sent, followed by the actual data (payload), followed possibly by a trailer. Multiple such envelopes may be provided for the same data, each carrying the corresponding control information for each layer in the network protocol stack [6,25].

**Buffering and Flow Control.** Buffering data of a master (output buffering) is used both for buses and NoCs to decouple computation from communication. However, for NoCs output buffering is also needed to marshal data, which consists of (a) (optionally) splitting the outgoing data in smaller packets which are transported by the network, and (b) adding control information for the network around the data (packet header). To avoid output buffer overflow the master must not initiate transactions that generate more data than the currently available space.

Similarly to output buffering, input buffering is also used to decouple computation from communication. In a NoC, input buffering is also required to unmarshal data.

In addition, flow control for input buffers differs for buses and NoCs. For buses, the source and destination are directly linked, and, destination can therefore signal directly to a source that it cannot accept data. This information can even be available to the arbiter such that the bus is not granted to a transaction trying to write to a full buffer.

In a NoC, however, the destination of a transaction cannot signal directly to a source that its input buffer is full. Consequently, transactions to a destination can be started, possibly from multiple sources, after the destination's input buffer has filled up. Several policies can be adopted when an input buffer is full. One policy is not to accept additional incoming transactions, and to store them in the network. However, this approach can easily lead to network congestion, as the data could be eventually stored all the way to the sources, blocking the links in between. Another policy is to accept incoming transactions at a full destination, and drop some data in the input buffer. Congestion is avoided but data is lost, and this is undesirable.

To avoid input buffer overflow connections can be used, together with end-to-end flow control. At connection set up between a master and one or more slaves, buffer space is allocated at the network interfaces of the slaves, and the network interface of the master is assigned credits reflecting the amount of buffer space at the slaves. The master can only send data when it has enough credits for the destination slave(s). The slaves grant credits to the master when they consume data.

#### IV. The *Æthereal* Approach

As described in the previous two sections, NoCs have different properties from both existing off-chip networks and existing on-chip interconnects. As a result, existing protocols and service interfaces cannot be adopted directly to NoCs, but must take the characteristics of NoCs into account. For example, a protocol such as TCP/IP assumes the network is lossy, and includes significant complexity to provide reliable communication. Therefore, TCP/IP is not suitable in a NoC where we assume data

transfer reliability is already solved at a lower level. On the other hand, existing on-chip protocols such as VCI, OCP, AMBA, DTL, or CoreConnect are also not directly applicable. For example, they assume ordered transport of data: if two requests are initiated from the same master, they will arrive in the same order at the destination. This does not hold automatically for NoCs. Atomic chains of transactions and end-to-end flow control also need special attention in a NoC interface.

Our objectives when defining the services of our on-chip network (called *Æthereal*) are the following. First, the services abstract from the network internals as much as possible. This is a key ingredient in tackling the challenge of decoupling the computation from communication [14,23], which allows IPs (the computation part), and the interconnect (the communication part) to be designed independently from each other. As a consequence, our services are positioned at the transport layer in the ISO-OSI reference model [25], which is the first layer to be independent of the implementation of the network.

Second, we aim at a NoC interface as close as possible to a bus interface. NoCs can then be introduced non-disruptively: existing IPs, methodologies and tools can continue to be used with minor changes. As a consequence, we use a request-response interface similar to interfaces for split buses [1, 12, 13, 17, 18, 26].

Third, our interface extends traditional bus interfaces to fully exploit the power of NoCs. For example, we offer connection-based communication which does not only relax ordering constraints (as for buses), but also enables new communication properties, such as end-to-end flow control based on credits, or guaranteed throughput [8, 20, 21]. All these properties can be set for each connection individually.

#### A. The *Æthereal* Connection and Transaction Model

IPs interact with our network [8, 20, 21] at so-called network interfaces (NI). NIs provide NI ports (NIP) through which the communication services are accessed. As shown in Figure 3, a NI can have several NIPs to which one or more IPs (computation elements or memories, but not interconnection elements) can be connected. Similarly, an IP can be connected to more than

one NI and NIP.

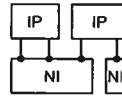


**Figure 3.** Examples of links between NIs and IPs.

Communication between NIPs is performed on *connections*. Connections are introduced to describe and identify communication with different properties, such as guaranteed throughput, bounded latency and jitter, ordered delivery, or flow control. For example, to distinguish and independently guarantee communication of 1Mbps and 25Mbps, two connections can be used. Two NIPs can be connected by multiple connections, possibly with different properties. Connections as defined here are similar to the concept of threads and connections from OCP and VCI. Where in OCP and VCI connections are used only to relax transaction ordering, we generalize from only the ordering property to include configuration of buffering and flow control, guaranteed throughput, and bounded latency per connection.

Æthereal connections must be *created* with the desired properties before being used. This may result in resource reservations inside the network (e.g., buffer space, or percentage of the link usage per time unit). If the requested resources are not available, the network will refuse the request. After usage, connections are *closed*, which leads to freeing the resources occupied by that connection.

To allow more flexibility in configuring connections and, hence, better resource allocation per connection, the outgoing and return parts of connections are configured independently. For example, a different amount of buffer space can be allocated in the NIPs at master and slaves, or different bandwidths can be reserved for requests and responses.

Depending on the requested services, the time to handle a connection (i.e., creating, closing, modifying services) can be short (e.g., creating/closing an unordered, lossy, best-effort connection) or significant (e.g., creating/closing a multicast guaranteed-throughput connection). Consequently, connections are assumed to be created, closed, or modified infrequently, coinciding e.g., with reconfiguration points, when the application requirements change.



**Figure 4.** Transaction composition.

Communication takes place on connections using *transactions*, consisting of a request and possibly a response. The request encodes an operation (e.g., read, write, flush, test and set, nop) and possibly carries outgoing data (e.g., for write commands). The response returns data as a result of a command (e.g., read) and/or an acknowledgment.

Connections involve at least two NIPs. Transactions on a connection are always started at one and only one of the NIPs, called the connection's *active* NIP (ANIP). All the other NIPs of the connection are called *passive* NIPs (PNIP).

There can be multiple transactions active on a connection at a time, but more generally than for split buses. That is, transactions can be started at the ANIP of a connection while responses for earlier transactions are pending. If a connection has multiple slaves, multiple transactions can be initiated towards different slaves. Transactions are also pipelined between a single master-slave pair for both requests and responses. In principle, transactions can also be pipelined within a slave, if the slave allows this.

A transaction is composed from the following messages (see Figure 4):

- A *command* message (CMD) is sent by the ANIP, and describes the action to be executed at the slave connected to the PNIP. Examples of commands are read, write, test and set, and flush. Commands are the only messages that are compulsory in a transaction. For NIPs that allow only a single command with no parameters (e.g., fixed-size address-less write), we assume the command message still exists, even if it is implicit (i.e., not explicitly sent by the IP).
- An *out data* message (OUTDATA) is sent by the ANIP following a command that requires data to be executed (e.g., write, multicast, and test-and-set).
- A *return data* message (RETDATA) is sent by a PNIP as a consequence of a transaction execution that produces data (e.g., read, and test-and-set).
- A *completion acknowledgment* message (RETSTAT) is an optional

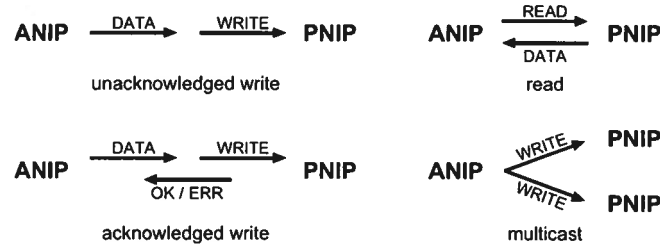


**Figure 5.** Transaction examples.

message which is returned by PNIP when a command has been completed. It may signal either a successful completion or an error. For transactions including both RETDATA and RETSTAT the two messages can be combined in a single message for efficiency. However, conceptually, they exist both: RETSTAT to signal the presence of data or an error, and RETDATA to carry the data. In bus-based interfaces RETDATA and RETSTAT typically exist as two separate signals [1, 12, 13, 17, 18, 26].

Messages composing a transaction are divided in *outgoing* messages, namely CMD and OUTDATA, and *response* messages, namely RETDATA, RETSTAT. Within a transaction, CMD precedes all other messages, and RETDATA precedes RETSTAT if present. These rules apply both between master and ANIP, and PNIP and slave. Examples of transactions are shown in Figure 5.

We classify connections as follows (see Figure 6):

- A *simple* connection is a connection between one ANIP and one PNIP.
- A *narrowcast* connection is a connection between one ANIP and one or more PNIPs, in which each transaction that the ANIP initiates is executed by exactly one PNIP. An example of the narrowcast connection is shown in Figure 7, where the ANIP performs transactions on an address space which is mapped on two memory modules. Depending on the transaction address, a transaction is executed on only one of these two memories.
- A *multicast* connection is a connection between one ANIP and one

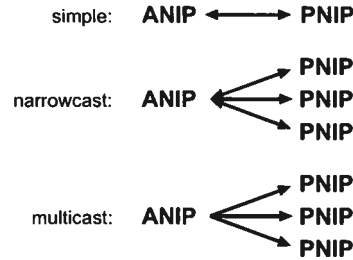


**Figure 6.** Connection types.

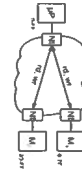


**Figure 7.** A narrowcast connection.

or more PNIPs, in which the sent messages are duplicated and each PNIP receives a copy of those messages. In a multicast connection no return messages are currently allowed, because of the large traffic they generate (i.e., one response per destination). It could also increase the complexity in the ANIP because individual responses from PNIPs must be merged into a single response for the ANIP. This requires buffer space and/or additional computation for the merging itself.

## B. Connection Properties

In this section we elaborate on the features that can be configured for a connection: guaranteed message integrity, guaranteed transaction completion, various transaction orderings, guaranteed throughput, bounded latency and jitter, and connection flow control.

**Data Integrity.** Data integrity means that the content of messages is not changed (accidentally or not) during transport. We assume that data integrity is already solved at a lower layer in our network, namely at the link layer, because in current on-chip technologies data can be transported uncorrupted over links. Consequently, our network interface always guarantees that messages are delivered uncorrupted at the destination.

**Transaction Completion.** A transaction without a response (e.g., a posted write) is said to be complete when it has been executed by the slave. As there is no response message to the master, no guarantee regarding transaction completion can be given.

A transaction with a response (e.g., an acknowledged write) is said to be complete when a RETSTAT message is received from the ANIP<sup>1</sup>. The transaction may either (a) be executed successfully, in which case a success RETSTAT is returned, (b) fail in its execution at the slave, in which case an execution error RETSTAT is returned, or (c) fail because of buffer overflow in a connection with no flow control, in which case it reports an overflow error. We assume that when a slave accepts a CMD requesting a response, the slave always generates the response.

In our network, routers do not drop data [21]. Therefore, messages are always guaranteed to be delivered at the NI. For connections with flow control, also NIs do not drop data. Thus, message delivery and, thus, transaction completion to the IPs is guaranteed automatically in this case.

However, if there is no flow control, messages may be dropped at the network interface in case of buffer overflow (see the paragraph on end-to-end flow control below). All of CMD, OUTDATA, and RETDATA may be dropped at the NI. To guarantee transaction completion, RETSTAT is not allowed to be dropped. Consequently, in the ANIPs enough buffer space must be provided to accommodate RETSTAT messages for all outstanding transactions. This is enforced by bounding the number of outstanding transactions.

**Transaction Ordering.** Across different connections no ordering of transactions is defined at the transport layer.

There are several points in a connection where order of transactions can be observed (see Figure 8): (a) the order in which the master presents CMD messages to the ANIP, (b) the order in which the CMDs are delivered to the slave by the PNIP, (c) the order in which the slave presents the responses to the PNIP, and (d) the order the responses are delivered to the master by the ANIP. Note that not all of (b), (c), and (d) are always present. Moreover, there are no assumptions about the order in which the slaves execute

---

<sup>1</sup> Recall that when data is received as a response (RETDATA), a RETSTAT (possibly implicit) is also received to validate the data.

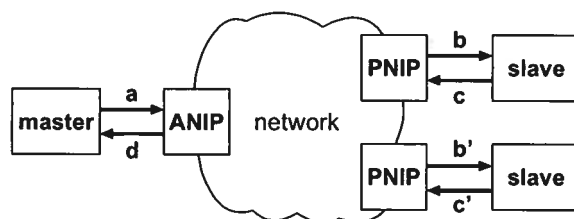


**Figure 8.** Message ordering is observable at a, b, c, and d.

transactions; we can only observe the order of the responses. We consider the order of the transaction execution by the slaves to be a system decision, and not a part of the interconnect protocol.

At both ANIP and PNIPs, outgoing messages belonging to different transactions on the same connection are allowed to be interleaved. For example, two write commands can be issued, and only afterwards their data. If the order of OUTDATA messages differs from the order of CMD messages, transaction identifiers must be introduced to associate OUTDATAS with their corresponding CMD.

Outgoing messages can be delivered by the PNIPs to the slaves (see Figure 8-b) as follows:

- *Unordered*, which imposes no order on the delivery of the outgoing messages of different transactions at the PNIPs.
- *Ordered locally*, where transactions must be delivered to each PNIP in the order they were sent (Figure 8-a), but no order is imposed across PNIPs. Locally-ordered delivery of the outgoing messages can be provided either by an ordered data transportation, or by reordering outgoing messages at the PNIP.
- *Ordered globally*, where transactions must be delivered in the order they were sent, across all PNIPs of the connection. Globally-ordered delivery of the outgoing part of transactions require a costly synchronization mechanism.

Transaction response messages can be delivered by the slaves to the PNIPs (see Figure 8-c) as follows:

- *Ordered*, when RETDATA and RETSTAT messages are returned in

the same order as the CMDs were delivered to the slave (Figure 8-b).

- *Unordered*, otherwise.

When responses are unordered, there has to be a mechanism to identify the transaction to which a response belongs. This is usually done using tags attached to messages for transaction identifications (similar to tags in VCI).

Response messages can be delivered by the ANIP to the master (see Figure 8-d) as follows:

- *Unordered*, which imposes no order on the delivery of responses. Here also, tags must be used to associate responses with their corresponding CMDs.
- *Ordered locally*, where RETDATA and RETSTAT messages of transactions for a single slave are delivered in the order the original CMDs were presented by the master to the ANIP (Figure 8-a). Note that there is no ordering imposed for transactions to different slaves within the same connection.
- *Globally ordered*, where all responses in a connection are delivered to the master in the same order as the original CMDs. When transactions are pipelined on a connection, then globally-ordered delivery of responses requires reordering at the ANIP.

All  $3 \times 2 \times 3 = 18$  combinations between the above orderings are possible. Out of these, we define and offer the following two.

An *unordered* connection is a connection in which no ordering is assumed in any part of the transactions. As a result, the responses must be tagged to be able identify to which transaction they belong. Implementing unordered connections has low cost, however, they may be harder to use, and introduce the overhead of tagging.

An *ordered* connection is defined as a connection with *local* ordering for the outgoing messages from PNIPS to slaves (Figure 8-b), *ordered* responses at the PNIPS (Figure 8-c), and *global* ordering for responses at the ANIP (Figure 8-d). We choose local ordering for the outgoing part because the global ordering has a too high cost, and has few uses. The ordering of responses is selected to allow a simple programming model with no tagging. Global ordering at the ANIP is possible at a moderate cost, because

all the reordering is done locally in the ANIP.

A user can emulate connections with global ordering of outgoing and return messages at the PNIPs using non-pipelined acknowledged transactions, at the cost of high latency.

**Connection latency, throughput, and jitter.** In our network, throughput can be reserved for connections in a time-division multiple access (TDMA) fashion, where bandwidth is split in fixed-size slots on a fixed time frame. Bandwidth, as well as bounds on latency and jitter, can be guaranteed when slots are reserved. They are all defined in multiples of the slots. throughput, latency and jitter can all be configured independently for the request and response parts of a connection.

Fully guaranteed-throughput connections (i.e., providing throughput guarantees on both request and return parts of the connection) can overbook resources in some cases. For example, when an ANIP opens a guaranteed-throughput read connection, it must reserve slots for the read command messages, and for the read data messages. The ratio between the two can be very large (e.g., 1:100), which leads either to a large number of slots, or bandwidth being wasted for the read command messages.

To resolve this problem, the request part of a connection can be best effort, while the response can have guaranteed throughput (or vice versa). For the example mentioned above, one can use best effort read messages, and guaranteed-throughput read-data messages. No global connection guarantees can be offered in this case, but the overall throughput can be higher and more stable than in the case of using only best-effort traffic.

**Connection flow control.** As mentioned earlier, our network guarantees that messages are delivered to the NI. Messages sent from one of the NIPs are not immediately visible at the other NIP, because of the multi-hop nature of networks. Consequently, handshakes over a network would allow only a single message be transmitted at a time. This limits the throughput on a connection and adds latency to transactions. To overcome this problem, and achieve a better network utilization, the messages must be pipelined. In this case, if the data is not consumed at the PNIP at the same rate it arrives, either flow control must be introduced to slow down the producer, or data may be lost because of limited buffer space at the consumer

NI.

We introduce end-to-end flow control at the level of connections, which requires buffer space to be associated with connections. End-to-end flow control ensures that messages are sent over the network only when there is enough space in the NIP's destination buffer to accommodate them.

End-to-end flow is optional (i.e., to be requested when connections are opened) and can be configured independently for the outgoing and return paths. When no flow control is provided, messages are dropped when buffers overflow. Multiple policies of dropping messages are possible, as in off-chip networks. Possible scenarios include: (a) the oldest message is dropped (milk policy), or (b) the newest message is dropped (wine policy) [25].

We opt for a credit-based flow control. Credits are associated with the empty buffer space at the receiver NI. The sender's credit is lowered as data is sent. When the PNIP delivers data to the slave, credits are granted to the sender. If the sender's credit is not sufficient to send some data, the NI at the sender stalls the sending.

### C. Use Cases

To illustrate the need for differentiated services on connections, we consider in this section two examples of traffic. We describe the properties they would use over an *Æ*thereal connection to meet their traffic requirements.

Video processing streams typically require a lossless, in-order video stream with guaranteed throughput, but possibly allow corrupted samples. An *Æ*thereal connection for such a stream would require the necessary throughput, ordered transactions, and flow control. If the video stream is produced by the master, only write transactions are necessary. In this case, with a flow-controlled connection there is no need to also require transaction completion, because messages are never dropped, and the write command and its data are always delivered at the destination. Data integrity is always provided by our network, even though it may be not necessary in this case.

Another example is that of cache updates which require uncorrupted, lossless, low-latency data transfer, but ordering and guaranteed throughput

are less important. In this case, a connection would not require any time related guarantees, because even though an average low latency is required, a guarantee on low latency is not critical. Low latency can be obtained even with a best-effort connection. The connection would also require flow control and guaranteed transaction completion to ensure lossless transactions. However, no ordering is necessary, because this is not important for cache updates, and allowing out of order transaction can reduce the response time.

## V. Conclusions

In this chapter, we compare networks on chip (NoC) to off-chip networks (e.g., computer networks) and existing on-chip interconnects (e.g., buses). We show that NoCs have many similarities with off-chip networks. However, they also differ, especially in their resource constraints. For example on a chip, memory and computation resources are more expensive, while there are more wires. This makes NoC architectures different from off-chip networks, and requires rethinking of network services.

We also compare NoCs to existing on-chip interconnects, such as buses and switches. By directly connecting IP blocks, existing on-chip interconnects can offer tight coupling between masters and slaves, and global arbitration. In NoCs, masters and slaves are completely decoupled, and the arbitration is distributed over the network nodes. This make it harder to provide guarantees, such as bandwidth lower bounds, and transaction orderings.

We define a set of NoC services that abstract from the network details. Using these services in IP design decouples computation and communication. We use a request-response transaction model to be close to existing on-chip interconnect protocols. This eases the migration of current IPs to NoCs. To fully utilize the NoC capabilities, such as high bandwidth and transaction concurrency, our services provide connection-oriented communication. Connections can be configured independently with different properties. These properties include transaction completion, various transaction orderings, bandwidth lower bounds, latency and jitter upper bounds, and flow control.

Our NoC services are a prerequisite for service-based system design which makes applications independent of NoC implementations, makes designs more robust, and enables architecture-independent quality-of-service strategies.

### References

1. ARM. *AMBA Specification. Rev. 2.0*, 1999.
2. ARM. *Multi-Layer AHB. Overview*, 2001.
3. J. Bainbridge and S. Furber. CHAIN: A delay-insensitive chip area interconnect. *IEEE Micro*, 22(5), 2002.
4. L. Benini and G. De Micheli. Powering networks on chips. In *ISSS*, 2001.
5. L. Benini and G. De Micheli. Networks on chips: A new SoC paradigm. *IEEE Computer*, 35(1):70–80, 2002.
6. D. J. Culler, J. P. Singh, and A. Gupta. *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufmann Publishers, 1999.
7. W. J. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *DAC*, 2001.
8. K. Goossens, J. van Meerbergen, A. Peeters, and P. Wielage. Networks on silicon: Combining best-effort and guaranteed services. In *DATE*, 2002.
9. P. Guerrier and A. Greiner. A generic architecture for on-chip packet-switched interconnections. In *DATE*, 2000.
10. P. J. Having. *Mobile Multimedia Systems*. PhD thesis, University of Twente, 2000.
11. J. Hennessy and D. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 1995.
12. IBM. *32-bit On-Chip Peripheral Bus. Rev. 2.1*, 2001.
13. IBM. *32-bit Processor Local Bus. Rev. 2.9*, 2001.
14. K. Keutzer, S. Malik, A. R. Newton, J. M. Rabaey, and A. Sangiovanni-Vincentelli. System-level design: Orthogonalization of concerns and platform-based design. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 19(12):1523–1543, 2000.

15. S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Öberg, Tiensyrjä, and A. Hemani. A network on chip architecture and design methodology. In *ISVLSI*, 2002.
16. J. A. Leijten, J. L. van Meerbergen, A. H. Timmer, and J. A. Jess. Prophid, a data-driven multi-processor architecture for high-performance DSP. In *ED&TC*, 1997.
17. OCP International Partnership. *Open Core Protocol Specification*, 2001.
18. Philips. *DTL Protocol Specification. Rev. 2.1*, 2001.
19. J. Rexford. *Tailoring Router Architectures to Performance Requirements in Cut-Through Networks*. PhD thesis, Univ. Michigan, 1999.
20. E. Rijpkema, K. Goossens, A. Rădulescu, J. Dielissen, J. van Meerbergen, P. Wielage, and E. Waterlander. Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip. In *DATE*, 2003.
21. E. Rijpkema, K. Goossens, and P. Wielage. A router architecture for networks on silicon. In *PROGRESS*, Oct. 2001.
22. M. T. Rose. *The Open Book: A Practical Perspective on OSI*. Prentice Hall, 1990.
23. M. Sgori, M. Sheets, K. Keutzer, S. Malik, J. Rabaey, and A. Sangiovanni-Vincentelli. Addressing the system-on-a-chip interconnect woes through communication-based design. In *DAC*, 2001.
24. P. Stravers and J. Hoogerbrugge. Homogeneous multiprocessing and the future of silicon design paradigms. In *VLSI-TSA*, 2001.
25. A. S. Tanenbaum. *Computer Networks*. Prentice Hall, 1996.
26. VSI Alliance. *Virtual Component Interface Standard*, 2000.
27. H. Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proc. of the IEEE*, 83(10):1374–1396, 1995.

# Exhibit I

(19) World Intellectual Property  
Organization  
International Bureau



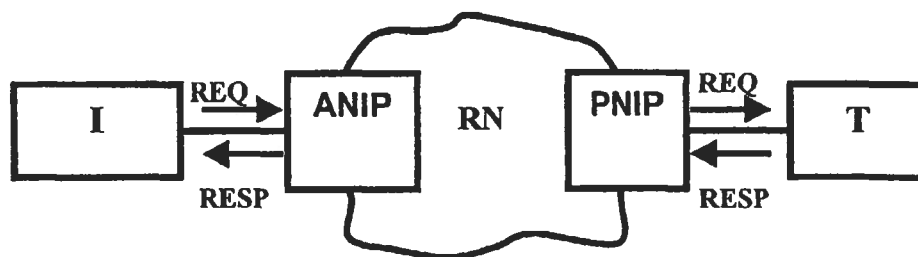
(43) International Publication Date  
22 April 2004 (22.04.2004)

PCT

(10) International Publication Number  
**WO 2004/034176 A2**

- (51) International Patent Classification<sup>7</sup>: **G06F** RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.
- (21) International Application Number: PCT/IB2003/004414 (84) Designated States (regional): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).
- (22) International Filing Date: 7 October 2003 (07.10.2003)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data: 02079196.8 8 October 2002 (08.10.2002) EP
- (71) Applicant (for all designated States except US): KONINKLIJKE PHILIPS ELECTRONICS N.V. [NL/NL]; Groenewoudseweg 1, NL-5621 BA Eindhoven (NL).
- (72) Inventors; and
- (75) Inventors/Applicants (for US only): RADULESCU, Andrei [RO/NL]; c/o Prof. Holstlaan 6, NL-5656 AA Eindhoven (NL). GOOSSENS, Kees, G., W. [NL/NL]; c/o Prof. Holstlaan 6, NL-5656 AA Eindhoven (NL).
- (74) Agent: DE JONG, Durk, J.; c/o Prof. Holstlaan 6, NL-5656 AA Eindhoven (NL).
- (81) Designated States (national): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VC, VN, YU, ZA, ZM, ZW, ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).
- Declaration under Rule 4.17:  
— as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii)) for the following designations AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VC, VN, YU, ZA, ZM, ZW, ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG)
- Published:  
— without international search report and to be republished upon receipt of that report
- For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: INTEGRATED CIRCUIT AND METHOD FOR ESTABLISHING TRANSACTIONS



(57) Abstract: An integrated circuit comprising a plurality of processing modules (M; I; S; T) and a network (N; RN) arranged for providing at least one connection between a first and at least one second module is provided. Said connections comprises a set of communication channels each having a set of connection properties. Said connection supports transactions comprising outgoing messages from the first module to the second module and return messages from the second module to the first module. The connection properties of the different communication channels of said connection can be adjusted independently. Therefore, the utilization of the resources of a network on chip is more efficient, since the connection between modules can be efficiently adapted to their actual requirement, such that the connection is not over dimensioned and unused network resources can be assigned to other connections.

WO 2004/034176 A2

## Integrated circuit and method for establishing transactions

## FIELD OF THE INVENTION

The invention relates to an integrated circuit having a plurality of processing modules and a network arranged for providing connections between processing modules and a method for exchanging messages in such an integrated circuit.

## 5 BACKGROUND OF THE INVENTION

Systems on silicon show a continuous increase in complexity due to the ever increasing need for implementing new features and improvements of existing functions. This is enabled by the increasing density with which components can be integrated on an integrated circuit. At the same time the clock speed at which circuits are operated tends to  
10 increase too. The higher clock speed in combination with the increased density of components has reduced the area which can operate synchronously within the same clock domain. This has created the need for a modular approach. According to such an approach the processing system comprises a plurality of relatively independent, complex modules. In conventional processing systems the systems modules usually communicate to each other via  
15 a bus. As the number of modules increases however, this way of communication is no longer practical for the following reasons. On the one hand the large number of modules forms a too high bus load. On the other hand the bus forms a communication bottleneck as it enables only one device to send data to the bus. A communication network forms an effective way to overcome these disadvantages.

20 Networks on chip (NoC) have received considerable attention recently as a solution to the interconnect problem in highly-complex chips. The reason is twofold. First, NoCs help resolve the electrical problems in new deep-submicron technologies, as they structure and manage global wires. At the same time they share wires, lowering their number and increasing their utilization. NoCs can also be energy efficient and reliable and are  
25 scalable compared to buses. Second, NoCs also decouple computation from communication, which is essential in managing the design of billion-transistor chips. NoCs achieve this decoupling because they are traditionally designed using protocol stacks, which provide well-defined interfaces separating communication service usage from service implementation.

## CONFIRMATION COPY

Using networks for on-chip communication when designing systems on chip (SoC), however, raises a number of new issues that must be taken into account. This is because, in contrast to existing on-chip interconnects (e.g., buses, switches, or point-to-point wires), where the communicating modules are directly connected, in a NoC the modules communicate remotely via network nodes. As a result, interconnect arbitration changes from centralized to distributed, and issues like out-of order transactions, higher latencies, and end-to-end flow control must be handled either by the intellectual property block (IP) or by the network.

Most of these topics have been already the subject of research in the field of local and wide area networks (computer networks) and as an interconnect for parallel machine interconnect networks. Both are very much related to on-chip networks, and many of the results in those fields are also applicable on chip. However, NoC's premises are different from off-chip networks, and, therefore, most of the network design choices must be reevaluated. On-chip networks have different properties (e.g., tighter link synchronization) and constraints (e.g., higher memory cost) leading to different design choices, which ultimately affect the network services.

NoCs differ from off-chip networks mainly in their constraints and synchronization. Typically, resource constraints are tighter on chip than off chip. Storage (i.e., memory) and computation resources are relatively more expensive, whereas the number of point-to-point links is larger on chip than off chip. Storage is expensive, because general-purpose on-chip memory, such as RAMs, occupy a large area. Having the memory distributed in the network components in relatively small sizes is even worse, as the overhead area in the memory then becomes dominant.

For on-chip networks computation too comes at a relatively high cost compared to off-chip networks. An off-chip network interface usually contains a dedicated processor to implement the protocol stack up to network layer or even higher, to relieve the host processor from the communication processing. Including a dedicated processor in a network interface is not feasible on chip, as the size of the network interface will become comparable to or larger than the IP to be connected to the network. Moreover, running the protocol stack on the IP itself may also be not feasible, because often these IPs have one dedicated function only, and do not have the capabilities to run a network protocol stack.

The number of wires and pins to connect network components is an order of magnitude larger on chip than off chip. If they are not used massively for other purposes than

NoC communication, they allow wide point-to-point interconnects (e.g., 300-bit links). This is not possible off-chip, where links are relatively narrower: 8-16 bits.

On-chip wires are also relatively shorter than off chip allowing a much tighter synchronization than off chip. This allows a reduction in the buffer space in the routers  
5 because the communication can be done at a smaller granularity. In the current semiconductor technologies, wires are also fast and reliable, which allows simpler link-layer protocols (e.g., no need for error correction, or retransmission). This also compensates for the lack of memory and computational resources.

Reliable communication: A consequence of the tight on-chip resource  
10 constraints is that the network components (i.e., routers and network interfaces) must be fairly simple to minimize computation and memory requirements. Luckily, on-chip wires provide a reliable communication medium, which can help to avoid the considerable overhead incurred by off-chip networks for providing reliable communication. Data integrity can be provided at low cost at the data link layer. However, data loss also depends on the  
15 network architecture, as in most computer networks data is simply dropped if congestion occurs in the network.

Deadlock: Computer network topologies have generally an irregular (possibly dynamic) structure, which can introduce buffer cycles. Deadlock can also be avoided, for example, by introducing constraints either in the topology or routing. Fat-tree topologies have  
20 already been considered for NoCs, where deadlock is avoided by bouncing back packets in the network in case of buffer overflow. Tile-based approaches to system design use mesh or torus network topologies, where deadlock can be avoided using, for example, a turn-model routing algorithm. Deadlock is mainly caused by cycles in the buffers. To avoid deadlock, routing must be cycle-free, because of its lower cost in achieving reliable communication. A  
25 second cause of deadlock are atomic chains of transactions. The reason is that while a module is locked, the queues storing transactions may get filled with transactions outside the atomic transaction chain, blocking the access of the transaction in the chain to reach the locked module. If atomic transaction chains must be implemented (to be compatible with processors allowing this, such as MIPS), the network nodes should be able to filter the  
30 transactions in the atomic chain.

Data ordering: In a network, data sent from a source to a destination may arrive out of order due to reordering in network nodes, following different routes, or retransmission after dropping. For off-chip networks out-of-order data delivery is typical. However, for NoCs where no data is dropped, data can be forced to follow the same path

between a source and a destination (deterministic routing) with no reordering. This in-order data transportation requires less buffer space, and reordering modules are no longer necessary.

Network flow control and buffering strategy: Network flow control and buffering strategy have a direct impact on the memory utilization in the network. Wormhole routing requires only a flit buffer (per queue) in the router, whereas store-and-forward and virtual-cut-through routing require at least the buffer space to accommodate a packet. Consequently, on chip, wormhole routing may be preferred over virtual-cut-through or store-and-forward routing. Similarly, input queuing may be a lower memory-cost alternative to virtual-output-queuing or output-queuing buffering strategies, because it has fewer queues. Dedicated (lower cost) FIFO memory structures also enable on-chip usage of virtual-cut-through routing or virtual output queuing for a better performance. However, using virtual-cut-through routing and virtual output queuing at the same time is still too costly.

Time-related guarantees: Off-chip networks typically use packet switching and offer best-effort services. Contention can occur at each network node, making latency guarantees very hard to offer. Throughput guarantees can still be offered using schemes such as rate-based switching or deadline-based packet switching, but with high buffering costs. An alternative to provide such time-related guarantees is to use time-division multiple access (TDMA) circuits, where every circuit is dedicated to a network connection. Circuits provide guarantees at a relatively low memory and computation cost. Network resource utilization is increased when the network architecture allows any left-over guaranteed bandwidth to be used by best-effort communication.

Introducing networks as on-chip interconnects radically changes the communication when compared to direct interconnects, such as buses or switches. This is because of the multi-hop nature of a network, where communication modules are not directly connected, but separated by one or more network nodes. This is in contrast with the prevalent existing interconnects (i.e., buses) where modules are directly connected. The implications of this change reside in the arbitration (which must change from centralized to distributed), and in the communication properties (e.g., ordering, or flow control).

An outline the differences of NoCs and buses will be given below. We refer mainly to buses as direct interconnects, because currently they are the most used on-chip interconnect. Most of the bus characteristics also hold for other direct interconnects (e.g., switches). Multilevel buses are a hybrid between buses and NoCs. For our purposes, depending on the functionality of the bridges, multilevel buses either behave like simple

buses or like NoCs. The programming model of a bus typically consists of load and store operations which are implemented as a sequence of primitive bus transactions. Bus interfaces typically have dedicated groups of wires for command, address, write data, and read data. A bus is a resource shared by multiple IPs. Therefore, before using it, IPs must go through an arbitration phase, where they request access to the bus, and block until the bus is granted to them.

A bus transaction involves a request and possibly a response. Modules issuing requests are called masters, and those serving requests are called slaves. If there is a single arbitration for a pair of request-response, the bus is called non-split. In this case, the bus remains allocated to the master of the transaction until the response is delivered, even when this takes a long time. Alternatively, in a split bus, the bus is released after the request to allow transactions from different masters to be initiated. However, a new arbitration must be performed for the response such that the slave can access the bus.

For both split and non-split buses, both communication parties have direct and immediate access to the status of the transaction. In contrast, network transactions are one-way transfers from an output buffer at the source to an input buffer at the destination that causes some action at the destination, the occurrence of which is not visible at the source. The effects of a network transaction are observable only through additional transactions. A request-response type of operation is still possible, but requires at least two distinct network transactions. Thus, a bus-like transaction in a NoC will essentially be a split transaction.

Transaction Ordering: Traditionally, on a bus all transactions are ordered (cf. Peripheral VCI, AMBA, or CoreConnect PLB and OPB. This is possible at a low cost, because the interconnect, being a direct link between the communicating parties, does not reorder data. However, on a split bus, a total ordering of transactions on a single master may still cause performance penalties, when slaves respond at different speeds. To solve this problem, recent extensions to bus protocols allow transactions to be performed on connections. Ordering of transactions within a connection is still preserved, but between connections there are no ordering constraints (e.g., OCP, or Basic VCI). A few of the bus protocols allow out-of-order responses per connection in their advanced modes (e.g., Advanced VCI), but both requests and responses arrive at the destination in the same order as they were sent.

In a NoC, ordering becomes weaker. Global ordering can only be provided at a very high cost due to the conflict between the distributed nature of the networks, and the requirement of a centralized arbitration necessary for global ordering. Even local ordering,

between a source-destination pair, may be costly. Data may arrive out of order if it is transported over multiple routes. In such cases, to still achieve an in-order delivery, data must be labeled with sequence numbers and reordered at the destination before being delivered. The communication network comprises a plurality of partly connected nodes.

- 5 Messages from a module are redirected by the nodes to one or more other nodes. To that end the message comprises first information indicative for the location of the addressed module(s) within the network. The message may further include second information indicative for a particular location within the module, such as a memory, or a register address. The second information may invoke a particular response of the addressed module.

- 10 Atomic Chains of Transactions: An atomic chain of transactions is a sequence of transactions initiated by a single master that is executed on a single slave exclusively. That is, other masters are denied access to that slave, once the first transaction in the chain claimed it. This mechanism is widely used to implement synchronization mechanisms between master modules (c.g., semaphores). On a bus, atomic operations can easily be  
15 implemented, as the central arbiter will either (a) lock the bus for exclusive use by the master requesting the atomic chain, or (b) know not to grant access to a locked slave. In the former case, the time resources are locked is shorter because once a master has been granted access to a bus, it can quickly perform all the transactions in the chain (no arbitration delay is required for the subsequent transactions in the chain). Consequently, the locked slave and the  
20 bus can be opened up again in a short time. This approach is used in AMBA and CoreConnect. In the latter case, the bus is not locked, and can still be used by other modules, however, at the price of a longer locking time of the slave. This approach is used in VCI and OCP.

- 25 In a NoC, where the arbitration is distributed, masters do not know that a slave is locked. Therefore, transactions to a locked slave may still be initiated, even though the locked slave cannot accept them. Consequently, to prevent deadlock, these other transactions in the atomic chain must be able to bypass them to be served. Moreover, the time a module is locked is much longer in case of NoCs, because of the higher latency per transaction.

- Media Arbitration: An important difference between buses and NoCs is in the  
30 medium arbitration scheme. In a bus, master modules request access to the interconnect, and the arbiter grants the access for the whole interconnect at once. Arbitration is centralized as there is only one arbiter component, and global as all the requests as well as the state of the interconnect are visible to the arbiter. Moreover, when a grant is given, the complete path from the source to the destination is exclusively reserved. In a non-split bus, arbitration takes

place once when a transaction is initiated. As a result, the bus is granted for both request and response. In a split bus, requests and responses are arbitrated separately.

In a NoC arbitration is also necessary, as it is a shared interconnect. However, in contrast to buses, the arbitration is distributed, because it is performed in every router, and is based only on local information. Arbitration of the communication resources (links, buffers) is performed incrementally as the request or response advances .

Destination Name and Routing: For a bus, the command, address, and data are broadcasted on the interconnect. They arrive at every destination, of which one activates based on the broadcasted address, and executes the requested command. This is possible because all modules are directly connected to the same bus. In a NoC, it is not feasible to broadcast information to all destinations, because it must be copied to all routers and network interfaces. This floods the network with data. The address is better decoded at the source to find a route to the destination module. A transaction address will therefore have two parts: (a) a destination identifier, and (b) an internal address at the destination.

Latency: Transaction latency is caused by two factors: (a) the access time to the bus, which is the time until the bus is granted, and (b) the latency introduced by the interconnect to transfer the data. For a bus, where the arbitration is centralized the access time is proportional to the number of masters connected to the bus. The transfer latency itself typically is constant and relatively fast, because the modules are linked directly. However, the speed of transfer is limited by the bus speed, which is relatively low.

In a NoC, arbitration is performed at each router for the following link. The access time per router is small. Both end-to-end access time and transport time increase proportionally to the number of hops between master and slave. However, network links are unidirectional and point to point, and hence can run at higher frequencies than buses, thus lowering the latency. From a latency prospective, using a bus or a network is a trade off between the number of modules connected to the interconnect (which affects access time), the speed of the interconnect, and the network topology.

Data Format: In most modern bus interfaces the data format is defined by separate wire groups for the transaction type, address, write data, read data, and return acknowledgments/ errors (e.g., VCI, OCP, AMBA, or CoreConnect). This is used to pipeline transactions. For example, concurrently with sending the address of a read transaction, the data of a previous write transaction can be sent, and the data from an even earlier read transaction can be received. Moreover, having dedicated wire groups simplifies the transaction decoding; there is no need for a mechanism to select between different kinds of

data sent over a common set of wires. Inside a network, there is typically no distinction between different kinds of data. Data is treated uniformly, and passed from one router to another. This is done to minimize the control overhead and buffering in routers. If separate wires would be used for each of the above-mentioned groups, separate routing, scheduling, and queuing would be needed, increasing the cost of routers.

In addition, in a network at each layer in the protocol stack, control information must be supplied together with the data (e.g., packet type, network address, or packet size). This control information is organized as an envelope around the data. That is, first a header is sent, followed by the actual data (payload), followed possibly by a trailer. Multiple such envelopes may be provided for the same data, each carrying the corresponding control information for each layer in the network protocol stack .

Buffering and Flow Control: Buffering data of a master (output buffering) is used both for buses and NoCs to decouple computation from communication. However, for NoCs output buffering is also needed to marshal data, which consists of (a) (optionally) splitting the outgoing data in smaller packets which are transported by the network, and (b) adding control information for the network around the data (packet header). To avoid output buffer overflow the master must not initiate transactions that generate more data than the currently available space. Similarly to output buffering, input buffering is also used to decouple computation from communication. In a NoC, input buffering is also required to unmarshal data.

In addition, flow control for input buffers differs for buses and NoCs. For buses, the source and destination are directly linked, and, destination can therefore signal directly to a source that it cannot accept data. This information can even be available to the arbiter, such that the bus is not granted to a transaction trying to write to a full buffer.

In a NoC, however, the destination of a transaction cannot signal directly to a source that its input buffer is full. Consequently, transactions to a destination can be started, possibly from multiple sources, after the destination's input buffer has filled up. If an input buffer is full, additional incoming transitions are not accepted, and stored in the network. However, this approach can easily lead to network congestion, as the data could be eventually stored all the way to the sources, blocking the links in between.

To avoid input buffer overflow connections can be used, together with end-to-end flow control. At connection set up between a master and one or more slaves, buffer space is allocated at the network interfaces of the slaves, and the network interface of the master is assigned credits reflecting the amount of buffer space at the slaves. The master can

only send data when it has enough credits for the destination slave(s). The slaves grant credits to the master when they consume data.

#### SUMMARY OF THE INVENTION

5           It is an object of the invention to provide an integrated circuit and a method for exchanging messages in an integrated circuit with a more effective usage of the properties of the network.

          This object is achieved by an integrated circuit according to claim 1 and a method for exchanging messages according to claim 7.

10           Therefore, an integrated circuit comprising a plurality of processing modules M; I; S; T and a network N; RN arranged for providing at least one connection between a first and at least one second module is provided. Said connections comprises a set of communication channels each having a set of connection properties. Said connection supports transactions comprising outgoing messages from the first module to the second  
15   module and return messages from the second module to the first module. The connection properties of the different communication channels of said connection can be adjusted independently.

          Therefore, the utilization of the resources of a network on chip is more efficient, since the connection between modules can be efficiently adapted to their actual  
20   requirement, such that the connection is not over dimensioned and unused network resources can be assigned to other connections.

          The invention is based on the idea to allow connection channels of a connection with different connection properties.

          According to an aspect of the invention, said integrated circuit comprises at  
25   least one communication managing means CM for managing the communication between different modules; and at least one resource managing means RM for managing the resources of the network N.

          According to a further aspect of the invention, said first module M; I issues a request for a connection to at least one of said second modules to said communication  
30   managing means CM. Said communication managing means CM forwards the request for a connection with communication channels each having a specific set of connection properties to said resource managing means (RM). Said resource managing means RM determines whether the requested connection based on said communication channels with said specific connection properties is available, and responds the availability of the requested connection

to said communication managing means CM. A connection between the first and second module is established based on the available properties or available network resources required to implement the properties of said communication channels of said connection.

According to a still further aspect of the invention, said communication  
5 managing means CM rejects establishing a connection based on the available connection properties when the available connection properties are not sufficient to perform the requested connection between said first and second module. The connection properties require that some network resources are implemented, e.g. a throughput requires slot reservation and flow control requires buffering. Therefore, a connection requiring some  
10 properties is opened or not depending on the availability of these resources. Accordingly, the communication manager CM has some control over the minimum requirements for a connection.

According to a further aspect of the invention, said communication managing means CM issues a request to reset the connection between said first and second module,  
15 when said modules have successfully performed their transactions, so that the network resources can be used again for other connections.

According to still a further aspect of the invention, said integrated circuit comprises at least one network interface means NI, associated to each of said modules, for managing the communication between said modules and said network N. Hence, the modules  
20 can be designed independent from the network and can therefore be re-used.

The invention also relates to a method for exchanging messages in an integrated circuit comprising a plurality of modules as described above. The messages between the modules are exchanged over connections via a network. Said connections comprises a set of communication channels each having a set of connection properties. Said  
25 connection through the network supports transactions comprising outgoing messages from the first module to the second module and return messages from the second module to the first module. The network manages the outgoing messages in a way different from the return messages, i.e. the connection channels can be configured independently.

Further aspects of the invention are described in the dependent claims.

30 These and other aspects of the invention are apparent from and will be elucidated with reference to the embodiment(s) described hereinafter.

## BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 shows a System on chip according to a first embodiment,

Fig. 2 shows a System on chip according to a second embodiment, and

Fig. 3 shows a System on chip according to a third embodiment.

## DESCRIPTION OF THE PREFERRED EMBODIMENTS

5           The following embodiments relate to systems on chip, i.e. a plurality of modules on the same chip communicate with each other via some kind of interconnect. The interconnect is embodied as a network on chip NOC. The network on chip may include wires, bus, time-division multiplexing, switch, and/or routers within a network. At the transport layer of said network, the communication between the modules are performed over  
10 connections. A connection is considered as a set of channels, each having a set of connection properties, between a first module and at least one second module. For a connection between a first module and a single second module, the connection comprises two channel, namely one from the first module to the second channel, i.e. the request channel, and a second from the second to the first module, i.e. the response channel. The request channel is reserved for  
15 data and messages from the first to the second, while the response channel is reserved for data and messages from the second to the first module. However, if the connection involves one first and N second modules,  $2*N$  channels are provided. The connection properties may include ordering (data transport in order), flow control (a remote buffer is reserved for a connection, and a data producer will be allowed to send data only when it is guaranteed that  
20 space is available for the produced data), throughput (a lower bound on throughput is guaranteed), latency (upper bound for latency is guaranteed), the lossiness (dropping of data), transmission termination, transaction completion, data correctness, priority, or data delivery.

Fig. 1 shows a System on chip according to a first embodiment. The system comprises a master module M, two slave modules S1, S2. Each module is connected to a  
25 network N via a network interface NI, respectively. The network interfaces NI are used as interfaces between the master and slave modules M, S1, S2 and the network N. The network interfaces NI are provided to manage the communication of the respective modules and the network N, so that the modules can perform their dedicated operation without having to deal with the communication with the network or other modules. The network interfaces NI can  
30 send requests such as read rd and write wr between each other over the network.

Fig. 2 shows a system on chip according to a second embodiment. The system comprises two modules, namely an initiator I and a target T, a router network RN, and two network interfaces ANIP, PNIP between the modules and the router network RN. The network interfaces provide two network interface ports NIP (one request and one response

port) through which the modules communicate with the router network RN or other modules via the router network RN. The network interface has one or more ports where modules can be connected. Two different type of ports are available, namely the active network interface port ANIP, which is connected to masters and the passive network interface port PNIP, which is connected to slave. The communication between the initiator module I and the target module T is based on request-response transactions, where the master, i.e. the initiator module I, initiates a transaction by placing a request, possibly with some data or required connection properties. The request REQ is delivered to the slave, i.e. the target module T, via the active network interface port ANIP, the network RN and the passive network interface port PNIP. The request is executed by the target module T and data is returned as a response RESP if necessary or required. This response RESP may include data and/or acknowledgement for the master or initiator module I.

The modules as described in Fig. 1 and 2 can be so-called intellectual property blocks IPs (computation elements, memories or a subsystem which may internally contain interconnect modules) that interact with network at said network interfaces NI. NIs provide NI ports NIP through which the communication services are accessed. A NI can have several NIPs to which one or more IPs can be connected. Similarly, an IP can be connected to more than one NI and NIP.

The communication over the network is performed by the network interfaces on connections. Connections are introduced to describe and identify communication with different properties, such as guaranteed throughput, bounded latency and jitter, ordered delivery, or flow control. For example, to distinguish and independently guarantee communication of 1Mbs and 25Mbs, two connections can be used. Two NIPs can be connected by multiple connections, possibly with different properties. Connections as defined here are similar to the concept of threads and connections from OCP and VCI. Where in OCP and VCI connections are used only to relax transaction ordering, we generalize from only the ordering property to include among others configuration of buffering and flow control, guaranteed throughput, and bounded latency per connection.

Fig. 3 shows a system on chip according to a third embodiment. The system of the third embodiment is based on the system according to the second embodiment and additionally comprises a communication manager CM and a resource manager RM. Here, the communication manager CM and the resource manager RM are arranged in the router network RN, but they can also be arranged in one or some of the network interfaces NI.

If the initiator module I needs to read or to write data from/to the target module T, a request REQ for a connection with the target module T is issued. This request REQ is send to a request port of the active network interface ports ANIP of the network interface associated to said initiator I. This request can contain information regarding an  
5 identification of at least one target module, i.e. a target ID, as well regarding the properties of the connection between the initiator module I and the target module T. The properties of the connection between the two modules may depend on the direction, i.e. the properties of the request channel can be different of the properties of the response channel.

The communication manager CM request a connection with a set of properties  
10 between two modules from the resource manager RM, after receiving the request REQ from the active network interface ports ANIP, i.e. the properties of a connection, e.g. throughput, flow control, are to be requested when asking for a connection setup. The resource manager RM allocates the necessary resources and enquires if such a connection based on these resources is possible. The properties require resources to be implemented (e.g., throughput  
15 requires slot reservations, flow control requires buffering). Therefore, a connection requiring some properties is opened or not depending on the availability of these resources. The availability of the connection properties correspond to the ability of the network to fulfill or provide the resources for the connection properties identified in the connection setup request. The allocation of the resources can be preformed in two ways. First, the resource manager  
20 RM may contain a property table with entries for all the different properties of the channels and connections. Alternatively, one central property table can be provided containing all different properties of the network, i.e. the table can either be central or distributed.

After enquiring of the available connection properties, the resource manager RM reserves the available connection i.e. the required resources, and responds to the  
25 communication manager CM which connection is available, i.e. which connection properties or the required resources thereof are available for the desired channels. Optionally, the communication manager CM can accept the connection with the available properties, but on the other hand the communication manager CM may also refuse the offered connection if the available properties or the resources thereof are not acceptable. If the communication  
30 manager CM refuses the offered connection, it sends an error message to the initiator module I (i.e. via the ANIP) and requests the resource module RM to release the reservation of said connections. Otherwise, the resource manager RM sets the connection properties, and establishes a connection with accepted properties between said initiator and target modules. After said two modules I, T have performed the transactions as requested by the initiator

module, the communication manager CM issues a request to said resource manager RM to reset the connection or the connection properties.

The connections according to the embodiments of the invention must be first created or established with the desired properties before being used. This may result in  
5 resource reservations inside the network (e.g., buffer space, or percentage of the link usage per time unit). If the requested resources are not available, the network RN will refuse the request. After usage, connections are closed, which leads to freeing the resources occupied by that connection.

To allow more flexibility in configuring connections, and, hence, better  
10 resource allocation per connection, the outgoing and return parts of connections can be configured independently. For example, a different amount of buffer space can be allocated in the NIPs at the master and slaves, or different bandwidths can be reserved for requests and responses.

An example for the use of differential properties for the outgoing and return  
15 parts is described as follows. Guaranteed-throughput connections can overbook resources in some cases. For example, when an ANIP opens a guaranteed-throughput read connection, it must reserve slots for the read command messages, and for the read data messages. The ratio between the two can be very large (e.g., 1:100), which leads either to a large number of slots, or bandwidth being wasted for the read command messages.

To solve this problem, the connection properties of the request and response  
20 parts of a connection can be configured independently for all of throughput, latency and jitter. Consequently, the connection properties of request part of a connection can be best effort, while the connection properties of response can have guaranteed throughput (or vice versa). For the example mentioned above, we can use best effort read messages, and guaranteed-  
25 throughput read-data messages. No global connection guarantees can be offered in this case, but the overall throughput can be higher and more stable than in the case of using only best-effort traffic.

A connection on which only read commands are executed for large blocks of data can be considered as further example. In such a case, if flow control is implemented, a  
30 network interface PNIP associated to a slave S, T would require a small buffer, while the network interface ANIP associated to a master M, I would require a large buffer. In an alternative example a guaranteed throughput is required. Typically, a guaranteed throughput are provided by reserving fixed-sized slots from a slot table, where a bandwidth value is associated to each slot, so that the bandwidth values are summed if more slots are allocated.

In such a scheme there will be a minimum bandwidth that can be allocated. Therefore, allocating bandwidth only for read commands would be an inefficient use of the bandwidth as read commands would only use a small fraction. The return data responses would probably use enough bandwidth to justify the slow reservation. To prevent such an inefficient  
5 bandwidth utilization, the request part of the connection can be set up as best effort (i.e. no throughput guarantees) and guaranteed throughput only for the response part. Accordingly, no overall guarantees will be offered but for a reasonable loaded network, such a connection will still perform well. For a network with a low load this solution may be even faster as best-effort data can be transferred faster (e.g. there is no need to wait for the assigned bandwidth  
10 slot).

Depending on the requested services, the time to handle a connection (i.e., creating, closing, modifying services) can be short (e.g., creating/closing an unordered, lossy, best-effort connection) or significant (e.g., creating/closing a multicast guaranteed-throughput connection). Consequently, connections are assumed to be created, closed, or  
15 modified infrequently, coinciding e.g. with reconfiguration points, when the application requirements change.

Communication takes place on connections using transaction, consisting of a request and possibly a response. The request encodes an operation (e.g., read, write, flush, test and set, nop) and possibly carries outgoing data (e.g., for write commands). The response  
20 returns data as a result of a command (e.g., read) and/or an acknowledgment. Connections involve at least two NIPs. Transactions on a connection are always started at one and only one of the NIPs, called the connection's active NIP (ANIP). All the other NIPs of the connection are called passive NIPs (PNIP).

There can be multiple transactions active on a connection at a time, but more  
25 generally than for split buses. That is, transactions can be started at the ANIP of a connection while responses for earlier transactions are pending. If a connection has multiple slaves, multiple transactions can be initiated towards different slaves. Transactions are also pipelined between a single master-slave pair for both requests and responses. In principle, transactions can also be pipelined within a slave, if the slave allows this.

30 A transaction can be composed from the following messages:

- A command message (CMD) is sent by the ANIP, and describes the action to be executed at the slave connected to the PNIP. Examples of commands are read, write, test and set, and flush. Commands are the only messages that are compulsory in a transaction. For NIPs that allow only a single command with no parameters (e.g., fixed-size address-less

write), we assume the command message still exists, even if it is implicit (i.e., not explicitly sent by the IP).

- An out data message (OUTDATA) is sent by the ANIP following a command that requires data to be executed (e.g., write, multicast, and test-and-set).
- 5 - A return data message (RETDATA) is sent by a PNIP as a consequence of a transaction execution that produces data (e.g., read, and test-and-set).
- A completion acknowledgment message (RETSTAT) is an optional message which is returned by PNIP when a command has been completed. It may signal either a successful completion or an error. For transactions including both RETDATA and RETSTAT
- 10 the two messages can be combined in a single message for efficiency. However, conceptually, they exist both: RETSTAT to signal the presence of data or an error, and RETDATA to carry the data. In bus-based interfaces RETDATA and RETSTAT typically exist as two separate signals.

Messages composing a transaction are divided in outgoing messages, namely

15 CMD and OUTDATA, and response messages, namely RETDATA, RETSTAT. Within a transaction, CMD precedes all other messages, and RETDATA precedes RETSTAT if present. These rules apply both between master and ANIP, and PNIP and slave.

Connections can be classified as follows:

- A simple connection is a connection between one ANIP and one PNIP.
- 20 - A narrowcast connection is a connection between one ANIP and one or more PNIPs, in which each transaction that the ANIP initiates is executed by exactly one PNIP. An example of the narrowcast connection, where the ANIP performs transactions on an address space which is mapped on two memory modules. Depending on the transaction address, a transaction is executed on only one of these two memories.
- 25 - A multicast connection is a connection between one ANIP and one or more PNIPs, in which the sent messages are duplicated and each PNIP receives a copy of those messages. In a multicast connection no return messages are currently allowed, because of the large traffic they generate (i.e., one response per destination). It could also increase the complexity in the ANIP because individual responses from PNIPs must be merged into a
- 30 single response for the ANIP. This requires buffer space and/or additional computation for the merging itself.

Connecting properties that can be configured for a connection are as follows: guaranteed message integrity, guaranteed transaction completion, various transaction orderings, guaranteed throughput, bounded latency and jitter, and connection flow control.

Data integrity means that the content of messages is not changed (accidentally or not) during transport. We assume that data integrity is already solved at a lower layer in our network, namely at the link layer, because in current on-chip technologies data can be transported uncorrupted over links. Consequently, our network interface always guarantees  
5 that messages are delivered uncorrupted at the destination.

A transaction without a response (e.g. a posted write) is said to be complete when it has been executed by the slave. As there is no response message to the master, no guarantee regarding transaction completion can be given.

A transaction with a response (e.g. an acknowledged write) is said to be  
10 complete when a RETSTAT message is received from the ANIP. Recall that when data is received as a response (RETDATA), a RETSTAT (possibly implicit) is also received to validate the data. The transaction may either be executed successfully, in which case a success RETSTAT is returned, fail in its execution at the slave, and then an execution error RETSTAT is returned, or fail because of buffer overflow in a connection with no flow  
15 control, and then it reports an overflow error. We assume that when a slave accepts a CMD requesting a response, the slave always generates the response.

In the network, routers do not drop data, therefore, messages are always guaranteed to be delivered at the NI. For connections with flow control, also NIs do not drop data. Thus, message delivery and, thus, transaction completion to the IPs is guaranteed  
20 automatically in this case.

However, if there is no flow control, messages may be dropped at the network interface in case of buffer overflow. All of CMD, OUTDATA, and RETDATA may be dropped at the NI. To guarantee transaction completion, RETSTAT is not allowed to be dropped. Consequently, in the ANIPs enough buffer space must be provided to accommodate  
25 RETSTAT messages for all outstanding transactions. This is enforced by bounding the number of outstanding transactions.

Now the ordering requirements between different transactions within a single connection are described. Over different connections no ordering of transactions is defined at the transport layer.

30 There are several points in a connection where the order of transactions can be observed: (a) the order in which the master module M, I presents CMD messages to the ANIP, (b) the order in which the CMDs are delivered to the slave module T, S by the PNIP, (c) the order in which the slave module T, S presents the responses to the PNIP, and (d) the order the responses are delivered to the master by the ANIP. Note that not all of (b), (c), and

(d) are always present. Moreover, there are no assumptions about the order in which the slaves execute transactions; only the order of the responses can be observed. The order of the transaction execution by the slaves is considered to be a system decision, and not a part of the interconnect protocol.

5                   At both ANIP and PNIPs, outgoing messages belonging to different transactions on the same connection are allowed to be interleaved. For example, two write commands can be issued, and only afterwards their data. If the order of OUTDATA messages differs from the order of CMD messages, transaction identifiers must be introduced to associate OUTDATAs with their corresponding CMD.

10                   Outgoing messages can be delivered by the PNIPs to the slaves (see b) as follows:

- Unordered, which imposes no order on the delivery of the outgoing messages of different transactions at the PNIPs.
- Ordered locally, where transactions must be delivered to each PNIP in the  
15                   order they were sent (a), but no order is imposed across PNIPs. Locally-ordered delivery of the outgoing messages can be provided either by an ordered data transportation, or by reordering outgoing messages at the PNIP.
- Ordered globally, where transactions must be delivered in the order they were sent, across all PNIPs of the connection. Globally-ordered delivery of the outgoing part of  
20                   transactions require a costly synchronization mechanism.

                    Transaction response messages can be delivered by the slaves to the PNIPs (c) as Ordered, when RETDATA and RETSTAT messages are returned in the same order as the CMDs were delivered to the slave (b), or as Unordered, otherwise. When responses are unordered, there has to be a mechanism to identify the transaction to which a response  
25                   belongs. This is usually done using tags attached to messages for transaction identifications (similar to tags in VCI).

                    Response messages can be delivered by the ANIP to the master (see d) as follows:

- Unordered, which imposes no order on the delivery of responses. Here, also,  
30                   tags must be used to associate responses with their corresponding CMDs.
- Ordered locally, where RETDATA and RETSTAT messages of transactions for a single slave are delivered in the order the original CMDs were presented by the master to the ANIP. Note that there is no ordering imposed for transactions to different slaves within the same connection.

- Globally ordered, where all responses in a connection are delivered to the master in the same order as the original CMDs. When transactions are pipelined on a connection, then globally-ordered delivery of responses requires reordering at the ANIP.

All  $3 \times 2 \times 3 = 18$  combinations between the above orderings are possible.

5 Out of these, we define and offer the following two. An unordered connection is a connection in which no ordering is assumed in any part of the transactions. As a result, the responses must be tagged to be able identify to which transaction they belong. Implementing unordered connections has low cost, however, they may be harder to use, and introduce the overhead of tagging.

10 An ordered connection is defined as a connection with local ordering for the outgoing messages from PNIPs to slaves, ordered responses at the PNIPs, and global ordering for responses at the ANIP. We choose local ordering for the outgoing part because the global ordering has a too high cost, and has few uses. The ordering of responses is selected to allow a simple programming model with no tagging. Global ordering at the ANIP  
15 is possible at a moderate cost, because all the reordering is done locally in the ANIP. A user can emulate connections with global ordering of outgoing and return messages at the PNIPs using non-pipelined acknowledged transactions, at the cost of high latency.

In the network, throughput can be reserved for connections in a time-division multiple access (TDMA) fashion, where bandwidth is split in fixed-size slots on a fixed time  
20 frame. Bandwidth, as well as bounds on latency and jitter can be guaranteed when slots are reserved. They are all defined in multiples of the slots.

As mentioned earlier, the network guarantees that messages are delivered to the NI. Messages sent from one of the NIPs are not immediately visible at the other NIP, because of the multi-hop nature of networks. Consequently, handshakes over a network  
25 would allow only a single message be transmitted at a time. This limits the throughput on a connection and adds latency to transactions. To solve this problem, and achieve a better network utilization, the messages must be pipelined. In this case, if the data is not consumed at the PNIP at the same rate it arrives, either flow control must be introduced to slow down the producer, or data may be lost because of limited buffer space at the consumer NI.

30 End-to-end flow control may be introduced at the level of connections, which requires buffer space to be associated with connections. End-to-end flow control ensures that messages are sent over the network only when there is enough space in the NIP's destination buffer to accommodate them.

End-to-end flow is optional (i.e., to be requested when connections are opened) and can be configured independently for the outgoing and return paths. When no flow control is provided, messages are dropped when buffers overflow. Multiple policies of dropping messages are possible, as in off-chip networks. Possible scenarios include: (a) the oldest message is dropped (milk policy), or (b) the newest message is dropped (wine policy).

One example for flow control is a credit-based flow control. Credits are associated with the empty buffer space at the receiver NI. The sender's credit is lowered as data is sent. When the PNIP delivers data to the slave, credits are granted to the sender. If the sender's credit is not sufficient to send some data, the NI at the sender stalls the sending.

To illustrate the need for differentiated services on connections, some examples of traffic are described below. Video processing streams typically require a loss less, in-order video stream with guaranteed throughput, but possibly allow corrupted samples. An connection for such a stream would require the necessary throughput, ordered transactions, and flow control. If the video stream is produced by the master, only write transactions are necessary. In such a case, with a flow-controlled connection there is no need to also require transaction completion, because messages are never dropped, and the write command and its data are always delivered at the destination. Data integrity is always provided by our network, even though it may be not necessary in this case.

Another example is that of cache updates which require uncorrupted, loss less, low-latency data transfer, but ordering and guaranteed throughput are less important. In such a case, a connection would not require any time related guarantees, because a low latency, even if preferable, is not critical. Low latency can be obtained even with a best-effort connection. The connection would also require flow control and guaranteed transaction completion to ensure loss less transactions. However, no ordering is necessary, because this is not important for cache updates, and allowing out of order transaction can reduce the response time.

A set of NoC services is defined that abstract from the network details. Using these services in IP design decouples computation and communication. A request-response transaction model is used to be close to existing on-chip interconnect protocols. This eases the migration of current IPs to NoCs. To fully utilize the NoC capabilities, such as high bandwidth and transaction concurrency, connection-oriented communication are provided. Connections can be configured independently with different properties. These properties include transaction completion, various transaction orderings, bandwidth lower bounds, latency and jitter upper bounds, and flow control.

The provision of such network on chip NoC services is a prerequisite for service-based system design which makes applications independent of NoC implementations, makes designs more robust, and enables architecture-independent quality-of-service strategies.

5           In other situations there may be different types of flow control (e.g. you never want to lose write commands, but don't mind losing read data). If a module can do both read and write commands, it may be important that write transactions always succeed (e.g. when writing to an interrupt controller), but that read transactions are not critical because they can be retried (so the CMD of the read transaction is dropped and the read never executed, or the  
10   RETDATA is dropped after the read has been executed. Another example is that if you know that writes always succeed if they are delivered, a flow-controlled connection is requested. Acknowledgements are not necessary in that case. Without flow control acknowledgements are compulsory, complicating the master and causing additional traffic.

          In the integrated circuit according to the invention the decision to drop  
15   messages or not is not decided per transaction but for the outgoing and return parts of connection as a whole. For example all outgoing messages having the format reads+address or writes+address+data) may be guaranteed lossless, while for all return messages (whether read data, write acknowledgements) packets may be dropped.

A connection could be opened as follows:

20   connid = open (             
                    nofc/fc,  
                    outgoing unordered/local/global,  
                    outgoing buffer size,  
                    return unordered/local/global,  
25                   return buffer size);

i.e. all outgoing messages have certain properties, and all return messages have certain properties. Where fc represents flow control and nofc represents no flow control.

          An alternative solution for deadlock in NoCs, which takes into consideration  
30   that modules connecting to the network are either masters (initiating requests and receiving responses), or slaves (receiving requests and sending back responses), is to maintain separate virtual networks (with separate buffers) for requests and responses.

          According to an embodiment of the invention, a method for exchanging messages in an integrated circuit comprising a plurality of modules requests for a connection

with specific properties between two modules, decides if said requested connection with said properties between said two modules is possible, responds the connection with available properties, establishes a connection with said properties between said two modules, and performs transactions between said two modules. Additionally, the available connection may  
5 be accepted and the properties of said connection may be reset.

According to a further embodiment of the invention, the network has a first mode wherein a message is transferred within a guaranteed time interval, and a second mode wherein a message is transferred as fast as possible with the available resources, wherein the outgoing transaction is a read message, requesting the second module to send data to the first  
10 module, wherein the return transaction is the data generated by the second module upon this request, and wherein the outgoing transaction is transferred according to the second mode, and the return transaction is transferred according to the first mode. For acknowledged write transaction the write command and the outgoing data for the master use guaranteed throughput, and acknowledgment from the slaves uses best effort. Except for time-related  
15 guarantees, there is also a distinction on the buffering in the above examples. For data messages there is potentially more buffering allocated than for commands and acknowledgments. Consequently, for a read transaction buffers for the return part would be larger than those for the outgoing part. For the acknowledged write buffers the outgoing part are larger, and those for acknowledgments are smaller.

According to a further embodiment, it is possible to allocate different  
20 bandwidths for different channels. However, there are also limitations. A slot table is used, which contains a number of slots in a time window. Bandwidth is reserved allocating these slots to connections. For example, if a table with 100 slots for a time frame of  $1\mu\text{s}$  is used, each slot will be allocated for  $1/100$  from  $1\mu\text{s} = 10\text{ns}$ . If the network provides  $1\text{Gb/s}$  per link,  
25 the bandwidth per slot will be  $1/100$  from  $1\text{Gb/s} = 10\text{Mb/s}$ . Therefore, only multiple of  $10\text{Mb/s}$  for guaranteed throughput traffic can be allocated. For a read command generating long bursts, allocating the minimum bandwidth of  $10\text{Mb/s}$  would be probably too much, as it will use only a small fraction of it. The bandwidth can indeed be used by best-effort traffic, however, not by other guaranteed throughput traffic. As a result, not all the traffic for which  
30 guarantees are needed may fit in the slot table. An alternative is to use more slots, but this increases the cost of the router. Accordingly, a best effort command may be a better solution.

According to still a further embodiment of the invention, a connection between a master and two slave is provided, wherein different properties are assigned to the different channels from the master to the slaves. One of these slaves is a fast memory and the

other one is a slow memory. A higher throughput is assigned to the channel connecting the master and the fast memory.

As described above, NoCs have different properties from both existing off-chip networks and existing on-chip interconnects. As a result, existing protocols and service  
5 interfaces cannot be adopted directly to NoCs, but must take the characteristics of NoCs into account. For example, a protocol such as TCP/IP assumes the network is lossy, and includes significant complexity to provide reliable communication. Therefore, it is not suitable in a NoC where we assume data transfer reliability is already solved at a lower level. On the other  
10 hand, existing on-chip protocols such as VCI, OCP, AMBA, or CoreConnect are also not directly applicable. For example, they assume ordered transport of data: if two requests are initiated from the same master, they will arrive in the same order at the destination. This does not hold automatically for NoCs. Atomic chains of transactions and end-to-end flow control also need special attention in a NoC interface.

The objectives when defining network services are the following. First, the  
15 services should abstract from the network internals as much as possible. This is a key ingredient in tackling the challenge of decoupling the computation from communication, which allows IPs (the computation part), and the interconnect (the communication part) to be designed independently from each other. As a consequence, the services are positioned at the transport layer in the OSI reference model, which is the first layer to be independent of the  
20 implementation of the network. Second, a NoC interface should be as close as possible to a bus interface. NoCs can then be introduced non-disruptively: with minor changes, existing IPs, methodologies and tools can continue to be used. As a consequence, a request-response interface is used, similar to interfaces for split buses. Third, the interface extends traditional bus interfaces to fully exploit the power of NoCs. For example, connection-based  
25 communication which does not only relax ordering constraints (as for buses), but also enables new communication properties, such as end-to-end flow control based on credits, or guaranteed throughput. All these properties can be set for each connection individually.

It should be noted that the above-mentioned embodiments illustrate rather than limit the invention, and that those skilled in the art will be able to design many alternative  
30 embodiments without departing from the scope of the appended claims. In the claims, any reference signs placed between parentheses shall not be construed as limiting the claim. The word "comprising" does not exclude the presence of elements or steps other than those listed in a claim. The word "a" or "an" preceding an element does not exclude the presence of a plurality of such elements. In the device claim enumerating several means, several of these

means can be embodied by one and the same item of hardware. The mere fact that certain measures are recited in mutually different dependent claims does not indicate that a combination of these measures cannot be used to advantage.

- Furthermore, any reference signs in the claims shall not be construed as
- 5 limiting the scope of the claims.

## CLAIMS:

1. Integrated circuit comprising a plurality of processing modules (M; I; S; T) and a network (N; RN) arranged for providing at least one connection between a first and at least one second module,  
wherein the at least one connection comprises a set of communication  
5 channels each having a set of connection properties, the connection properties of the different communication channels of said connection being adjustable independently,  
wherein said connection supports transactions comprising outgoing messages from the first module to the second module and/or return messages from the second module to the first module.
- 10
2. Integrated circuit according to claim 1, further comprising:  
at least one communication managing means (CM) for managing the  
communication between different modules; and  
15 at least one resource managing means (RM) for managing the resources of the network (N).
3. Integrated circuit according to claim 2, wherein  
said first module (M; I) is adapted to issue a request (REQ) for a connection  
20 with at least one of said second modules to said communication managing means (CM),  
said communication managing means (CM) is adapted to forward said request (REQ) for a connection with communication channels each having a specific set of connection properties to said resource managing means (RM),  
said resource managing means (RM) is adapted to determine whether the  
25 requested connection based on said communication channels with said specific connection properties are available, and to respond the availability of the requested connection to said communication managing means (CM),  
wherein a connection between the first and second module is established based on the available properties of said communication channels of said connection.

4. Integrated circuit according to claim 2 or 3, wherein  
said communication managing means (CM) is adapted to reject establishing a  
connection based on the available connection properties when the available connection  
5 properties are not sufficient to perform the requested connection between said first and  
second module (M, I, S, T).
5. Integrated circuit according to claim 2, 3 or 4, wherein  
said communication managing means (CM) is adapted to request a reset of the  
10 connection between said first and second module(M, I, S, T), when said modules have  
successfully performed their transactions.
6. Integrated circuit according to claim 2, 3, 4 or 5, further comprising:  
at least one network interface means (NI), associated to each of said modules,  
15 for managing the communication between said modules and said network (N).
7. Method for exchanging messages in an integrated circuit comprising a  
plurality of modules, the messages between the modules being exchanged over connections  
via a network,  
20 wherein said connections comprises a set of communication channels each  
having a set of connection properties, any communication channel being independently  
configurable,  
wherein said connection through the network supports transactions comprising  
outgoing messages from the first module to the second module and/or return messages from  
25 the second module to the first module.

1/1

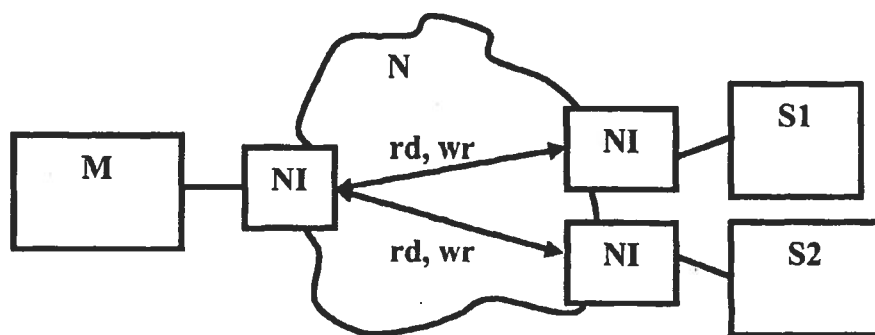


FIG. 1

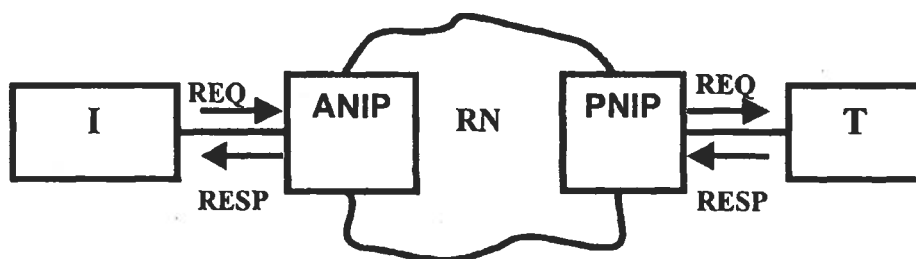


FIG. 2

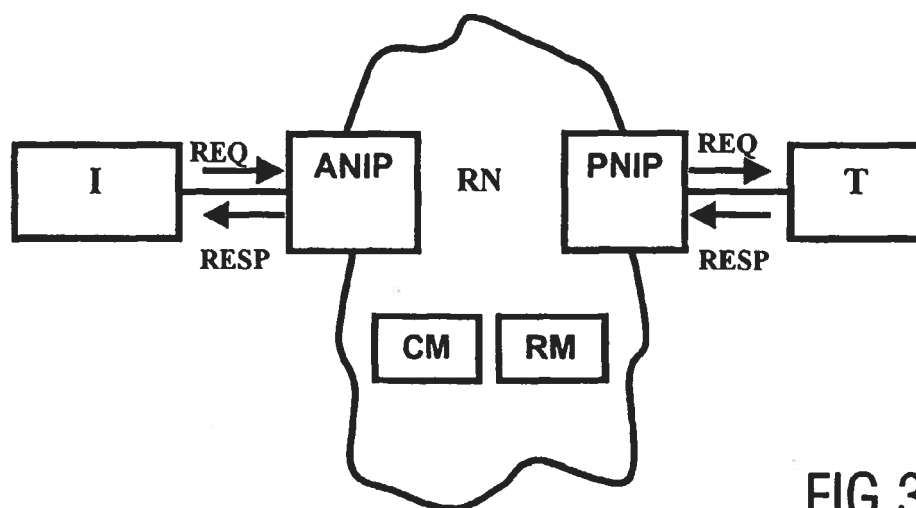


FIG. 3

# Exhibit J

# Networks on Silicon: Blessing or Nightmare?

Paul Wielage and Kees Goossens

Philips Research Laboratories, Eindhoven, The Netherlands

{paul.wielage,kees.goossens}@philips.com

## Abstract

Continuing VLSI technology scaling raises several deep submicron (DSM) problems like relatively slow interconnect, power dissipation and distribution, and signal integrity. Those problems are encountered particularly on long wires for global interconnect. As clock frequencies increase, scaled wires become relatively slower, and on-chip communication will be the limiting performance factor of future chips. We explain why efficiently sharing of the wires for long distance communication is the solution to this problem. We introduce networks on silicon (NoS), that route packets over shared (semi)-global wires. NoS performance is expected to be high, but comes at a cost. Balancing the performance and cost of a NoS is a major challenge, and we believe busses still have a role play.

## 1 Technology trend

VLSI technology scaling has long followed Moore's law. No fundamental barriers have been identified that invalidate this law for at least another decade [12]. Moore's law predicts that chips in 2010 will count over 4 billion transistors, operating in the multi-GHz range. This abundance of transistors will make very complex *systems on silicon* (SoS) possible.

However, challenges at all abstraction levels of design will have to be addressed before such SoSs will become a reality. The three most important deep submicron (DSM) challenges, related to all abstraction levels, are: substantial wire delay, controlling power delivery and dissipation, and assuring signal integrity.

Until recently, on-chip wiring was cheap. Consequently architectural models have been employed that relied on low-latency communication to globally share expensive computational resources. Global wire delay stays at best constant under technology scaling and hence these wires become effectively slower compared to a gate delay. For example, for 130 nm technology the reachable distance of a repeated global signal in a clock cycle is no more than the length of a

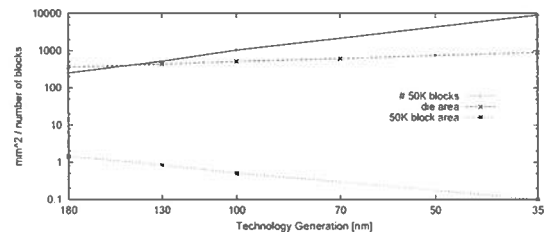


Figure 1. The number of 50k blocks for future process technologies.

chip [4]. For 50 nm technology, crossing a chip with highly optimized interconnect takes between six and ten clock-cycles, clearly invalidating the low-latency assumption of today. Hence we must move to system-level architectures that scale with technology.

A feasible template for a future-proof architecture is constructed from processing nodes that do not grow in complexity with technology. Instead, as technology scales, the number of these processing nodes on the chip grows. An on-chip communication network then combines these nodes into a SoS [4].

Various publications show that the spanning wires in blocks of 50k gates scale with technology [4, 13]. This means that the aforementioned DSM issues can be handled by CAD tools, assuming their evolutionary improvement. Figure 1 shows the exponentially increasing amount of such 50k blocks for a large die in subsequent technologies; in 35 nm this number is approximately ten thousand (adapted from [13] and [4]). It remains to find a communication architecture that allows a SoS composed of these blocks co-operate efficiently.

## 2 Networks on silicon are inevitable

Given the growing demand for and impact of interconnect on system cost and performance, it is worthwhile to optimize the utilization of wires. Ad-hoc global wiring struc-

tures often lead to a huge number of wires with an average usage as low as 10% in time [2]. To control cost in this scenario, the wire packing density must be very high, which is not beneficial for the power and delay characteristics. Efficient mechanisms for sharing (semi)-global wires must solve this cost-performance dilemma.

In deep submicron technologies, (semi)-global wires need special attention for power, signal-integrity, and performance reasons. In the discussion below we show how special circuit techniques can handle these issues. Such techniques only work, however, when embedded in dedicated communication IP, which provides a more abstract interface.

Power is an issue for global interconnect because it costs more energy to send a bit of information over longer wires. To reduce the communication delay, the energy consumption increases due to bigger drivers. Employing low-swing signaling for the global wires saves up to a factor four in power for these wires [15]. Implementing low-swing signaling requires special circuit techniques.

Signal integrity is hampered increasingly by growing capacitive and inductive coupling between wires. Capacitive noise coupling is the result of the large aspect ratio of wires in DSM technologies. Inductive noise coupling becomes more of a problem due to the decreasing transition times. IR drop<sup>1</sup> in the supply distribution increasingly contributes to the noise. The most effective way to make a connection robust against noise is application of differential signaling [7]. Differential signaling improves both the generation of and sensitivity to noise.

The signal propagation delay of an uninterrupted wire grows quadratically with its length; hence from a certain length onwards it is advantageous to partition the wire in segments with repeaters in between. The repeater insertion technique improves bandwidth and latency but at the cost of higher power consumption. Wire delay can be reduced by fat wires with a lower resistance per unit length at the cost of lower wire density. Such wires behave like lossy transmission lines and require drivers with a resistance matched to the transmission line.

As a result, we believe that all inter-block communication will be implemented by hard-macro transmitters and receivers, employing low-swing differential signaling, with well-controlled interconnect instead of ad-hoc drivers handled by standard place-and-route tools. In this way, communication links can be realized with predictable performance and DSM robustness.

Currently, the prevalent on-chip interconnects are busses [1]. In a bus architecture, devices share a single transmission medium to communicate. At a given time,

<sup>1</sup>Supply voltage drops are caused by high currents (I) flowing through the resistance (R) of the supply network. Since the supply voltage reduces under scaling IR drop worsens.

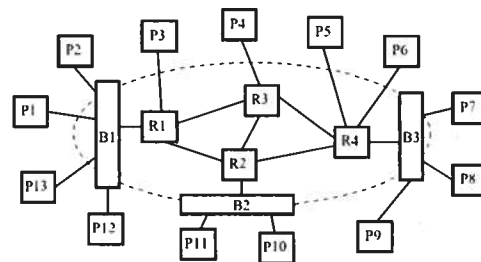


**Figure 2. Structural view of a network on silicon consisting of processing nodes (P) and nodes supporting communication (R, B).**

only one device has access to the shared medium. An arbitration mechanism is required to order simultaneous accesses. Such functionality is typically performed by a centralized bus arbiter. The performance of a shared-medium bus scales badly. For an increasing number of bus clients (i) individual clients get less bandwidth on average, and (ii) increased capacitive loads and wire length decrease the total bandwidth.

A solution that pairs scalable communication performance and minimal interconnect cost is expected from *networks on silicon* (NoS) where the SoS is considered as a network of components [2, 3, 1]. Figure 2 illustrates the hardware architecture of this concept. The outer components (marked P) exclusively perform processing and storage functions, whereas the inner components (marked B and R) form the NoS and cater to communication needs of the outer components. The basic building blocks of a NoS are routers (R).

A router forwards data from its input ports to its output ports in a concurrent fashion. To that end, a router of arity  $N$  contains a  $N \times N$  switch matrix. Data packets make their way through the network based on the routing information in their headers. A link between two routers is implemented by a point-to-point connection. The links typically span medium to long distances ranging from several to over more than twenty millimeters. The actual length depends on the chosen topology of the network. For a mesh topology the links are relatively short, for a torus which is a mesh with wrap-around connections, some links have a length of half the edge of the chip. Links can be optimized for bandwidth, latency, power, or a combination of these, depending on performance requirements.

### 3 NoS requirements

An important characteristic of a future system-level architecture is the separation between computation and com-

munication. A NoS allows the computational blocks to communicate with one other via a uniform interface. A uniform interface is advantageous because (i) it frees the core developer from having to make assumptions about the system in which the core will be used, and (ii) does not constrain the development of newer communication architectures by detailed interfacing requirements of particular legacy SoC components [6]. Several on-chip bus standards are evolving to realize this goal, most notably VCI, put forward by VSIA [14], and more recently, the Open Core Protocol [10].

The fundamental aim of a NoS is to provide flexible and efficient communication between the thousands of IP blocks in a system, with performance guarantees. In a typical SoS, the communication demands of different IP blocks show large variations. For example, data rates may be constant (e.g. digital video) or variable (e.g. compressed video). The importance of latency and jitter also varies greatly. Finally, the data granularity may range from single words to large blocks. A NoS should be able to offer different services to different clients. Each service class must be implemented efficiently, using a shared uniform infrastructure.

A high utilization of the network comes at a price. When the network starts to saturate, throughput and latency will show huge variations, which is not acceptable in real-time applications. Hence, the network should also provide guarantees, like loss-less data transport, minimal bandwidth, and bounded latency. The way packets are buffered and scheduled in routers, and the effects on performance guarantees has been the subject of intense research. Fundamentally, sharing and guarantees are conflicting, and efficiently combining guaranteed traffic with best-effort traffic is hard [11]. Although best-effort services are cheaper than guaranteed services we believe that the latter are essential because they enable compositional and scalable integration of the IP blocks [5]. It is up to the IP integrator at design time, and up to the application at run time, to make a trade off.

## 4 Performance and cost analysis of NoSs

The vision of previous sections is that the design of future SoSs will allow IP blocks to be plugged in at will to minimize communication costs, but without today's problems like timing closure. In this section we investigate the cost implications of system design based on a NoS. We hope the vision comes at acceptable cost. We hope that the overall cost of a NoS, including the full protocol stack to use it, turn out to be acceptable such that the integration blessings of NoSs do not change into a cost nightmare.

### 4.1 Performance

The aggregate bandwidth of a router is the product of the bandwidth per port,  $BW_{port}$ , the arity of the router (number of ports),  $N$ , and a utilization factor,  $\alpha \leq 1$  corresponding to the router arbitration scheme.

$$BW_{router} = \alpha N BW_{port} \quad (1)$$

We discuss each in turn. The bandwidth per port is determined by the bandwidth of the link and the router data path. In short:

$$BW_{port} = B \min(BW_{wire}, BW_{router\_data\_path}) \quad (2)$$

where  $B$  is the width of the data path. The combined bandwidth of the  $B$  wires of a link is a function of the layout characteristics (e.g. total length), chosen signaling technique, and the budgets for power, delay, and area. A first-order expression for the bandwidth of a repeated global wire optimized for power-delay is

$$BW_{wire} = \frac{1}{3 \cdot 2 \cdot \frac{2}{3} \cdot FO4} \text{ (bits/sec)} \quad (3)$$

where FO4 is the delay of an inverter driving four equally sized inverters [4]. In a 100 nm technology, this yields 5 Gb/s per wire under worst-case environmental conditions. Notice that the bandwidth of repeated global wires scales with technology because such wires allow (wave) pipelining at the segments.

Running the router data path at 5 GHz is not feasible. An aggressive but realistic frequency is 1.25 GHz corresponding the clock frequency of 50k gates blocks [4]. The critical function in the data path is the  $N \times N$  switch. For  $N$  up to 20 it meets the 1.25 GHz data rate, using  $N$  1-out-of- $N$  multiplexors. The relaxed demand on the wires of the link can be used to reduce power dissipation and area.

The utilization factor,  $\alpha$ , reflects the effectiveness of the router to resolve contention on the links. The queuing strategy, the queue sizes, and the schedule algorithm all strongly influence  $\alpha$ . Accordingly, many queuing policies and scheduling algorithms have been presented in the literature. For example,  $\alpha = 0.59$  for infinite fifo input queues with uniform and independent traffic. (Virtual) output queuing gives  $\alpha = 1$  under the same conditions, but at the cost of larger queues and a more complex scheduling algorithm [8]. Static scheduling techniques like (time-division-multiplexed) circuit switching can also improve the utilization factor.

Hence, in 100 nm technology, the bandwidth of a 32 bit router port is approximately 5 GByte/sec.

### 4.2 Cost

Three main components contribute to the area cost of a router: the switch, the control logic, and the packet queues.

The switch allows  $N$  simultaneous connections from the  $N$  inputs to the  $N$  outputs which results in  $B$  arrays of  $N \times N$  wires, giving rise to an  $O(N^2)$  area cost.

The control logic of a router is made up of the switch-matrix schedule unit and other configuration logic. The delay of a schedule cycle varies greatly per algorithm (for example, for virtual output queuing from  $O(1)$  to  $O(N^{5/2})$  [9]); it is important for two reasons. First, it determines the lower bound for latency that a flit<sup>2</sup> incurs to traverse the router. Second, it affects the size of the queues. The longer a schedule cycle, the more data arrive, given a fixed bandwidth of a port  $BW_{port}$ . This leads to deeper queues, and higher area cost.

The three aforementioned queuing strategies require queues of size  $O(N)$  to  $O(N^2)$  flits. Scheduling algorithms perform better with deeper queues, with a decreasing return.

Besides routers, a significant amount of area is consumed by so-called *network interfaces* (NI) modules. These modules translate the IP transactions for a given connection to packets that are sent over the network, and vice versa. Packets can be sent once the payload has been completely accepted by the NI. Hence, the buffers must be dimensioned such that, at least a complete packet for every simultaneously active connection can be stored.

The trade off between utilization  $\alpha$  and the cost is a complex one, but of importance to the viability of NoSs.

## 5 The future role of busses

In sections 1 and 2 we have argued that NoSs are essential to solve SoS integration in a scalable fashion. While Section 4.2 raised some general cost issues, we will now more concretely consider the trade off between busses and NoSs. *Will packet-switched NoSs completely replace current busses in future SoSs, or will a hybrid approach emerge?* We believe that shared busses may have a role to play in first-level communication (B in Figure 2) for the following reasons.

First, typical IP blocks underutilize the bandwidth capacity of an individual router port. All router ports offer the same bandwidth that is inherent to the architecture, whereas the bandwidth requirements of IP blocks varies greatly. A shared memory module needs typically much higher (peak) bandwidth than a streaming peripheral device. Single word transfers, variable bit rates, bursty IO, and much lower clock rates for IP blocks than for the NoS further waste bandwidth. This means that the communication needs of a number of IP blocks can be aggregated using a bus before the capacity of a network link is reached.

Second, network interfaces are more expensive (in terms of area) than a bus adaptor. Using a bus as a first-level traf-

<sup>2</sup>Flit stands for flow control digit, the atomic portion of data handled per schedule cycle. A packet is decomposed in flits.

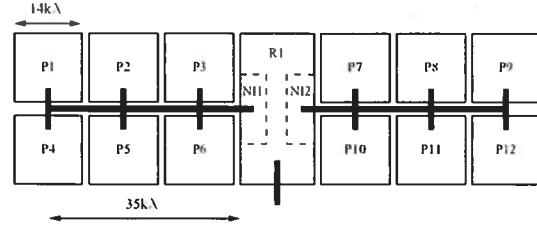


**Figure 3. A shared-medium bus seems a cost-effective way to connect the IP to the packet-switched network.**

fic concentrator, trading bus adaptors for network interfaces thus reduces the overall cost of IP-NoS interfacing. We expect that the overhead of a bus and its network interface are outweighed.

Finally, the number of routers is reduced significantly when busses are used as the first-level interconnect. Routers are larger than busses due to their packet queues and more complex scheduling. We give an example below.

An example of the heterogeneous communication architecture is depicted in Figure 3. A router of arity three surrounded by twelve IP blocks is shown. Two shared-medium busses, each connected to six 50k gates IP blocks, communicate with the router via two network interfaces. These have two functions: first they schedule the transactions on the bus, and second they give the bus clients access to the packet-switched network. The third port of the router provides communication to the remainder of the network. Figure 4 shows an architecture using only routers. Now three routers of arity five and one of arity four are needed.

The suggested shared-medium bus has a length of  $35k\lambda$ , where  $\lambda$  is half of the length of a minimal transistor. Global wires of this length will not be the bottle-neck of bus performance.<sup>3</sup>

The feasibility of hybrid NoSs hinges on the right implementation of the busses. First, they must be shared wires, as opposed to switches. Second, their arbitration must be combined, or at least compatible with, the scheduling taking place in the network interfaces, to offer uniform end-to-end network services.

We see a future for hybrid NoSs, with first-level communication over a shared-medium bus, and the higher levels using a packet-switched network. Perhaps a packet-switched network can be seen as a distributed and scalable implementation of a logical bridge that connects all the local busses of the SoS. Deciding how many IP blocks can use a local bus

<sup>3</sup>Minimum-delay wire segments have a length of  $28k\lambda$ , wire segments optimized for power-delay product have a length of  $48k\lambda$ . These lengths scale with technology as the edge of 50k blocks [4].

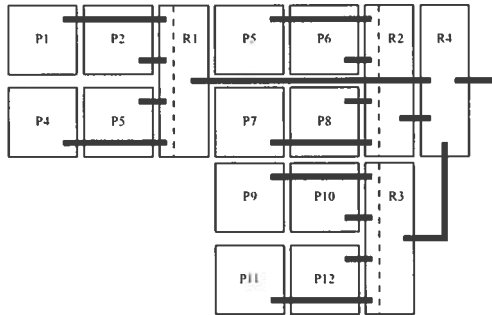


**Figure 4. IP to IP communication based on a homogeneous router network.**

before connecting to the router network is a question that must be answered foremost.

## 6 Conclusion

We have argued in Section 1 that future systems on silicon (SoS) will be composed of large numbers of processing nodes (or IP blocks). Each processing node is relatively small (50k gates) to scale with technology, and can be handled by CAD tools, assuming their evolutionary improvement. The interconnect and communication between these blocks then becomes an essential function in itself (Section 2), leading to networks on silicon (NoS). A NoS is based on packet switching to flexibly share link capacity between the network clients, and to provide pluriform communication services over a uniform infrastructure. Both efficiency, provided by best-effort traffic, and predictable performance, such as guaranteed throughput and latency, are important (Section 3). Efficiently combining them is a challenge. Section 4 showed that the performance of a NoS depends on many factors, but is expected to be high. The cost of a NoS can be stated in terms of area (routers, network interfaces), utilization of wires, and speed (latency). They can be traded off against one another, but also, perhaps more interestingly, against the cost of busses. A hybrid NoS using shared-wire busses to communicate locally, and accumulating traffic for a core router network is a promising architecture that deserves to be investigated.

## References

- [1] L. Benini and G. D. Micheli. Networks on chips: A new SoC paradigm. *IEEE Computer*, 35(1):70–80, Jan. 2002.
- [2] W. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *Proc. of DAC*, 2001.
- [3] P. Guerrier and A. Greiner. A generic architecture for on-chip packet-switched interconnections. In *Proc. of DATE*, 2000.
- [4] R. Ho, K. W. Mai, and M. A. Horowitz. The future of wires. *Proceedings of the IEEE*, 89(4), April 2001.
- [5] K. Goossens, J. van Meerbergen, A. Peeters and P. Wielage. Networks on silicon: Combining best-effort and guaranteed services. In *Proc. of DATE*, 2002.
- [6] K. Lahiri, A. Raghunathan, and S. Dey. Evaluation of the traffic-performance characteristics of system-on-chip communication architectures. In *Proc. of Int. Conf. on VLSI Design*, 2001.
- [7] Y. Massoud, J. Kawa, D. MacMillen, and J. White. Modeling and analysis of differential signaling for minimizing inductive cross-talk. In *Proc. of DAC*, 2001.
- [8] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand. Achieving 100% throughput in an input-queued switch. *IEEE Transactions on Communications*, 47(8), August 1999.
- [9] N. W. McKeown. *Scheduling Algorithms for Input-Queued Cell Switches*. PhD thesis, University of California, Berkeley, 1995.
- [10] Open core protocol specification version 1.0. <http://www.sonicsinc.com>, 1999.
- [11] J. Rexford and K. G. Shin. Support for multiple classes of traffic in multicomputer routers. In *Proceedings of the Parallel Computer Routing and Communication Workshop*, pages 116–130, May 1994. Lecture Notes in Computer Science 853.
- [12] S. Rusu. Trends and challenges in vlsi technology scaling towards 100nm (invited paper). In *Proc. of ESSCIRC*, 2001.
- [13] D. Sylvester and K. Keutzer. Impact of small process geometries on microarchitectures in systems on a chip. *Proceedings of the IEEE*, 89(4), April 2001.
- [14] On chip bus attributes specification 1 OCB 1 1 0. on-chip bus DWG. <http://www.vsi.org/library/specs/summary.htm>.
- [15] H. Zhang, V. George, and J. M. Rabaey. Low-swing on-chip signaling techniques: Effectiveness and robustness. *IEEE Transactions on VLSI*, 8(3), June 2000.

# Exhibit K

# Bringing Communication Networks On Chip: Test and Verification Implications

Bart Vermeulen

John Dielissen

Kees Goossens

Calin Ciordas†

Philips Research Laboratories  
Prof. Holstlaan 4, 5656 AA Eindhoven  
The Netherlands

† Eindhoven University of Technology

{Bart.Vermeulen, John.Dielissen, Kees.Goossens}@philips.com, C.Ciordas@tue.nl

## Abstract

In this article we present test and verification challenges for system chips that utilise on-chip networks. These systems on a chip (SOCs) and networks on a chip (NOCs) are introduced, where the NOC is exemplified by Philips's ÆTHEREAL NOC architecture. We discuss existing test and verification methods for SOCs and NOCs, and show the particular advantages of using a NOC both for testing and verifying the network, and for testing and verifying the other components of the SOC. This paper is concluded with our experiences with NOCs and a description of on-going work within Philips in this emerging field.

## 1 Introduction

High-performance networking requires dedicated hardware with tremendous computational and communication performance. Network components such as network interfaces and routers are complex systems that are built in a modular fashion by combining many application-specific integrated circuits (ASICs). With increasing packet throughput ASIC performance must increase. Moreover, trends towards differentiated services and higher quality of service require additional performance. Examples are more discerning packet classification, traffic shaping, network management, debug, and so on. To address these issues networking ASICs must become more versatile and programmable, often evolving towards *network processors*.

To indicate that network processors and ASICs are complex systems in themselves, they are usually named *systems on a chip* or SOCs. The number of components in a SOC is growing rapidly, and the communication infrastructure on a single SOC is a major concern. In fact, on-chip interconnect will increasingly be implemented as a *network on a chip* (NOC), complete with network interfaces, routers, and packet or circuit switching. Although the distances over which communication takes place differ by many orders of magnitude, the fields of on-chip networking and computer networking are clearly related.

In Section 2 we show why and how NOCs are used to implement SOC communication needs and we illustrate why their implementation is different compared to computer networks. Section 3 outlines Philips' ÆTHEREAL NOC, which is one such a solution.

In Section 4 we consider how a SOC, constructed from many hardware blocks (called *cores*) and a NOC can be tested for manufacturing defects. Section 5 describes the functional verification of a SOC, the verification of a NOC, and how a NOC can aid in the verification of a SOC. In Section 6 we present how an application, mapped onto a SOC with a NOC, can be verified. We end this paper in Section 7 with conclusions and highlight future work for the development of the Philips ÆTHEREAL NOC.

## 2 Networks on Chip

To manage the complexity of designing a SOC containing multiple cores, design teams are adopting core re-use methodologies. These methodologies allow cores, once they have been designed, to be re-used in multiple SOCs. As a result, the complexity of building a complete SOC shifts from the design of the individual cores to the design of the communication architecture connecting the cores. To prevent the design of this communication architecture from becoming the bottleneck for the design of future SOCs, this communication architecture itself has to be compositional and scalable.

A single broadcast medium, such as Amba and Silicon Backplane [1] buses, can already no longer deliver the required global bandwidth and latency for current SOCs. Switches, such as multi-layer Amba and Prophid, provide some relief, but are ultimately not scalable. Mirroring computer networks, a trend can be observed towards using networks of routers with circuit or packet switching to implement on chip communication [2, 3]. Of particular interest are SOCs for networking applications that themselves use NOCs. Examples of SOCs with an on-chip mesh of packet-switched routers to implement a single-chip switch are given in [4, 5]. Karim et al. [6] show how a network processor for OC-768 uses a hybrid circuit/packet-switched NOC.

Although computer networks and on-chip networks share many requirements, there are also a number of differences, which can lead to different trade offs [2, 7], and hence different architectures. Examples are:

- Quality of service beyond best-effort traffic is probably more important in SOCs for consumer electronics than for Internet services, due to their embedded, real-time, and often safety-critical nature [7]. Consumer applications have to be robust and require predictable performance.
- The conditions on a chip are currently more stable than off chip. On-chip routers are considered either faulty or correct, and hence the network topology is static (and not upgradable) after the chip leaves the factory.
- Routers and network interfaces of a NOC are more resource constrained than those in computer networks because they are intended for main-stream consumer products. As a consequence the chip area must be minimised leading to few and shallow buffers, fast and simple arbitration, limited traffic shaping, etc.
- On-chip communication links are relatively short compared to those in computer networks. Pipelining or transmission-line effects are therefore absent. This advantage partially offsets the severe resource constraints: buffers *can* be small due to the tight synchronisation between routers, and buffer overflow can be prevented by using flow control.
- In contrast to computer networks, inter-router wires are relatively abundant in NOCs [2]. Links can be wide, and their utilisation is probably less important.

Within Philips, the need to manage present-day and future on-chip communication demands spurred the development of the *ÆTHEREAL* NOC. Details of its architecture are presented in the next section.

## 3 The *Æthereal* Network on Chip

### 3.1 The *Æthereal* Architecture

The Philips *ÆTHEREAL* NOC [7, 8] addresses the communication needs for consumer-electronics SOCs with real-time requirements, as are for example used in digital video set-top boxes. Figure 1 shows an example SOC, consisting of cores and an *ÆTHEREAL* NOC.

Cores communicate with each other using the NOC, and include memories ( $M_i$ ), programmable or dedicated processors ( $P_i$ ), and external memory interfaces ( $MI_i$ ). The NOC consists of routers ( $R_i$ ) and network

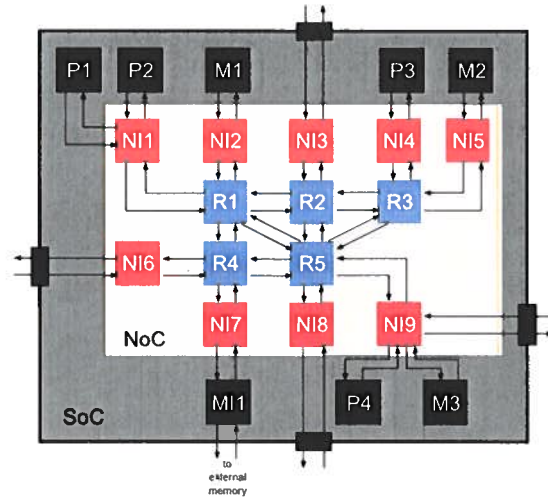


Figure 1: Example SoC with an ÆTHEREAL NoC.

interfaces ( $NI_i$ ), which are linked by non-pipelined wires. These routers and NIs are described in more detail in the following subsections.

### 3.2 The Æthereal Router

Conceptually, the ÆTHEREAL router module consist of two independent routers, see Figure 2(a).

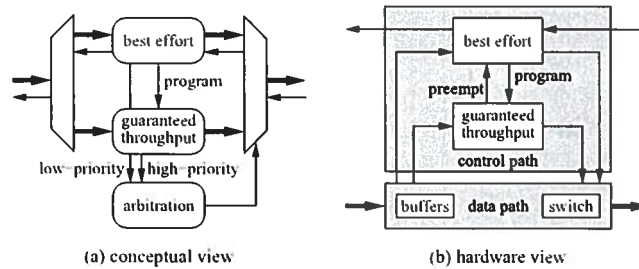


Figure 2: Two views of the combined GT-BE router.

The *best-effort* (BE) router offers uncorrupted, lossless (flow-controlled), ordered data transport. The *guaranteed-throughput* (GT) router adds hard throughput and latency guarantees over a finite time interval. The GT router fulfils our real-time service requirement, and combining it with a BE router ensures efficient resource utilisation.

#### 3.2.1 The Guaranteed-Throughput Router

To offer not merely statistical, but also hard guaranteed latency the network and hence the routers must be lossless. Besides this easy-to-solve property, also contention must be eliminated or quantified to be able to provide a guarantee. Rate-based and deadline-based scheduling offer guarantees [9], but they require deep output or priority queues. These queues are too costly for on-chip use. A low-cost alternative is to avoid contention completely, by scheduling packets at the network edge such that they are never in the same place, or are never there at the same time, or a combination. The ÆTHEREAL GT router uses a combination of both with time-division-multiplexed pipelined circuits. Every router and network interface block contains

a slot table  $T(s, o) = i$ , defining for a given slot  $s$  from which input  $i$  output  $o$  takes its data, if available. For this approach to work, all NOC blocks must share a common notion of time to ensure that their slot tables remain aligned. This is feasible in NOCs by using mesochronous clocking (synchronous clocks with constant skew), or asynchronous hand shaking, as described by Öberg in chapter 8 of [7]. BE packets are used to program the slot tables to set up and tear down GT connections, akin to ATM. This is shown in Figure 2(a) by the arrow labelled “program.” Router programming packets follow the same route as the connection they program. Slot allocations can be computed and programmed at run time in a distributed manner, or can be (pre-)computed off-line and then configured at run time.

### 3.2.2 The Best-Effort Router

The BE router is a classical input-queued wormhole router, and it uses round-robin arbitration for fairness. Data packets are never reordered in the router, and because deterministic routing is used, ordering is preserved end to end. Programming packets are shunted to a programming module in the router, and spliced in the data stream after they have programmed the slot table.

### 3.2.3 The Combined Router

As is shown in Figure 2(b) the control paths of the BE and GT routers are separate, yet interrelated. Moreover, the arbitration unit (including link-level flow control for the BE router) of Figure 2(a) has been merged with the BE router itself. The data path, mainly consisting of the switch matrix, is shared. In computer-network router architectures the buffers of BE and GT traffic would be stored in a shared RAM. For the small amount of buffering in on-chip routers (in our case, 3 words per GT queue, and 24 words per BE queue) using either RAMs or register-file memories would be very area inefficient. By using dedicated GT and BE hardware fifos (labelled GQ and BQ in Figure 3), the area of the router is reduced by two-thirds.

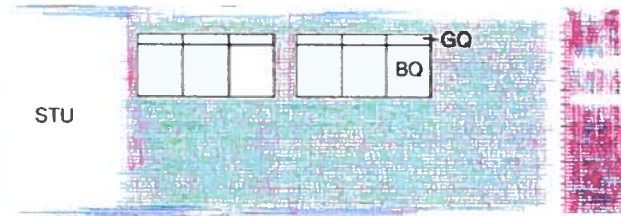


Figure 3: Lay-out of a combined GT-BE router.

We synthesised an arity 5 router with, a BE queue depth of 24 words of 32 bits, and a 256 slots table (labelled STU) in a 0.12 micron technology. The lay-out is shown in Figure 3. It has an aggregate bandwidth of  $5 \times 500 \text{ MHz} \times 32 \text{ bit} = 80 \text{ Gbit/s}$ . The area of the router is  $0.26 \text{ mm}^2$  in a 0.12 micron CMOS process using 6 metal layers.

## 3.3 The Æthereal Network Interface

The network interface is the bridge between a core and a router, where in general more cores can be connected to one network interface. It implements end-to-end flow control, admission control and traffic shaping, connection set up and tear down, and transaction reordering. Like the router, it contains a slot table, but it has dedicated hardware fifos per connection.

## 4 Manufacturing Test

Like all other SOC, an NOC-based SOC has to be tested for manufacturing defects. The NOC can be considered as just another core of a SOC, but it is also special in two ways: (1) it is often composed of many identical sub-cores (routers and network interfaces), and (2) it occupies a privileged, central position in the SOC, by virtue of its interconnecting role. In this section we explore the most relevant options for efficiently and effectively testing of a SOC with a NOC.

### 4.1 SOC Manufacturing Test

With the design of a scalable and modular SOC comes the issue of testing for manufacturing defects. Over the last years, several advances have been made in SOC testing. IEEE P1500 [10] is standardising a core-based test approach. In this approach, cores are wrapped in a test wrapper to allow easy testing of that core in a SOC. A so-called Test Access Mechanism (TAM) is used in conjunction with test wrappers around the core to transport test data to and from a core-under-test. Combined they allow application-independent test access to all on-chip cores. An example of this core-based test approach is shown in Figure 4.

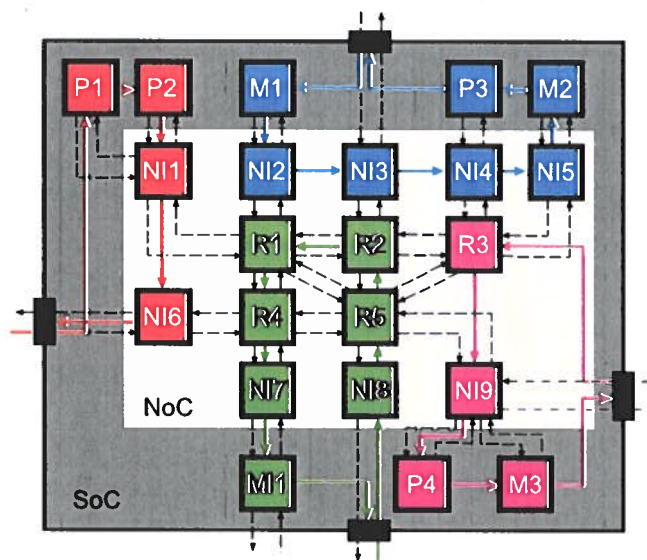


Figure 4: Default core-based testing

In test mode, the SOC cores are distributed over four TAMs connected to the four I/O interfaces of the chip. In Figure 4, these four TAMs are indicated with the colours red, green, blue, and purple. During scan test all TAMs are used in parallel to minimise the total test time. A disadvantage of this method is that it requires additional wires to be added to the design to form the TAMs. In network chips, adding these TAMs on top of an already large number of interconnects might cause wire congestion problems during the layout phase of the design.

Built-in Self Test (BIST) is another popular approach to test mainly regular logic blocks, such as for example on-chip memories. Developments have been made to extend the use of BIST to other cores than memories as well. Techniques such as random test pattern generation, in combination with test point insertion or bit-flipping [11] are used to test these blocks efficiently.

As test pattern set one will typically use the traditional stuck-at fault test patterns, complemented by both delay fault test patterns and possibly IDDQ or  $\Delta$ IDDQ test patterns to meet the test quality requirements.

## 4.2 Testing the NOC

Given the NOC's regular and hierarchical structure, it makes sense to adopt a core-based test approach. If we know the function of the core-under-test, i.e. a NOC we can utilise this knowledge to modify the standard core-based test approach to obtain a better-suited test. Figure 5(a) shows a possible core-based approach for an SOC with an NOC.

The blocks that make up the NOC are tested first. If the NOC contains a fault, the entire SOC can be sent off for diagnosis, without testing it further in the production line. This leads to a reduction in test time, and consequently a savings in test cost.

The concept of test re-use can be taken one step further. While testing the NOC, all identical blocks (e.g. all routers) can reuse the same test data. This test data set is broadcast and applied to all routers at the same time. The responses can be compared against each other and any mismatches sent off-chip. This allows for a very efficient pass-fail decision on the NOC. Note that each block still needs to be made uniquely accessible for diagnostic purposes.

Timing tests are also very important for testing a NOC because: (1) all clock boundaries between cores are inside the network interface, and (2) the NOC is spread over the entire SOC and has therefore many long wires. These interconnection wires are more vulnerable to timing errors and cross-talk than others. Sending delay fault test patterns with a high fault coverage over all communication links therefore increases the overall fault coverage. When NOCs become very large, additional test strategies such as those applied in FPGAs can also be included, as for example described by Ubar et al. in chapter 7 of [7].

## 4.3 Testing a SOC through its NOC

After the NOC has been structurally tested, the network can be used to functionally transport test data to and from the cores in a very flexible way. Figure 5(b) shows how the functionality of the NOC is used during the test of the other blocks in the SOC. The NOC is now considered a known-correct block that can be used to transport data from the device pins to the core-under-test and back. An advantage of the re-use of the NOC functionality is that no new TAM wires need to be added to the design, as the existing NOC wires are re-used. The flexibility of the NOC also enables the simultaneous distribution of test data to multiple identical cores. The test data itself can come from off chip, or from an internal BIST module.

When a SOC has structural errors, there are three possibilities: (1) the SOC is thrown away, (2) redundant hardware present on the SOC can replace the faulty cores (repair), or (3) the SOC is sold as a lower-performance SOC. In cases (2) and (3), the NOC's flexibility is used to an advantage. During the manufacturing test, the error information has to be collected and subsequently permanently stored inside the SOC.

## 5 Silicon Verification

Silicon verification involves checking whether the *hardware design* is correct, assuming it has been manufactured correctly. This process is often referred to as silicon debug, as the goal is to try to find any remaining design errors that might have slipped through the pre-silicon verification stages. In this section, we first discuss silicon debug of SOCs and of communication networks. We then look in more detail at the options for verifying a NOC as part of a SOC, and how a NOC can be used for the verification of the other cores of the SOC.

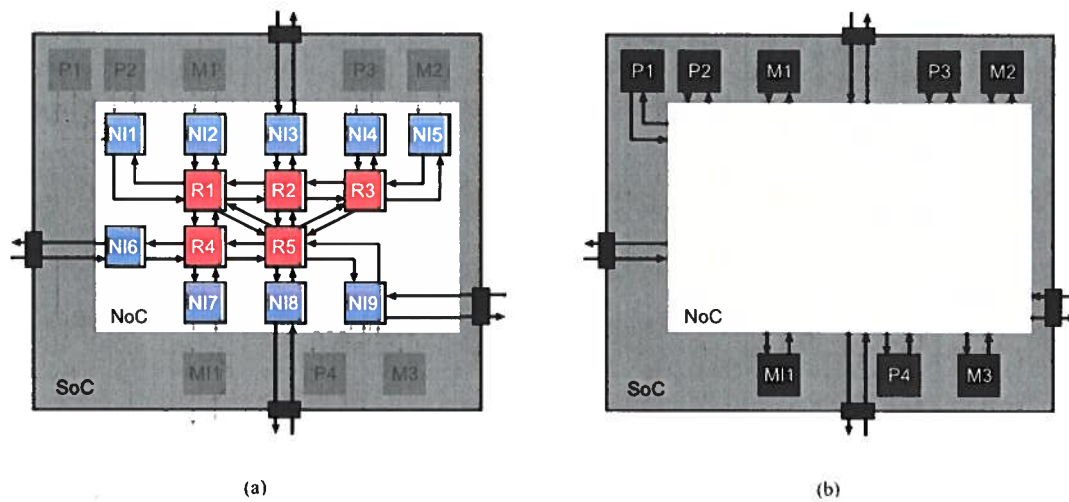


Figure 5: Testing an SoC with a NoC

## 5.1 SOC Silicon Verification

Many companies have adopted design-for-debug strategies to allow prototype silicon to be efficiently and effectively verified [12, 13, 14]. The methods they use can be split into two complementary categories: those that change the configuration of the hardware of the chip to access debug data (intrusive) and those that can acquire debug data in parallel to the functional hardware (non-intrusive).

- **Intrusive debug**

This category covers all debug methods that impact the application before debug data can be examined. These methods add on-chip breakpoint modules to the design of the SoC. These breakpoint modules interrupt the execution of the chip, after an internal sequence of events has been detected. All functional processing is at that point frozen. Various methods are then applied to access internal data. Commonly used access paths included system bus read and writes, TAP-based DMA access, and TAP-based scan-chain access [14].

- **Non-intrusive debug**

This category covers those debug methods that allow examination of debug data in real-time by streaming data for debug to an on-chip memory or out of the chip via a set of dedicated chip pins. Examples include the JTAG and the IEEE-ISTO 5001 NEXUS standard. These methods add hardware to the design that only observes the functionality of the chip, operating completely in parallel to it. This allows the application to run at the actual operating frequencies. As this debug architecture is completely separated (apart from probe points) for the functional hardware, care must be taken to keep the associated area cost within acceptable limits.

A hybrid solution is often chosen, combining these two methods, depending on the specific debug requirements.

## 5.2 Network Verification

In contrast to the previous section, network verification is about in-field testing. In the prior work on network verification two major areas can be recognized. First of all, a lot of work has been done on network errors: malfunctioning routers, routers that drop out of the network, links that are broken, error detection, etc., and recovery procedures for all these cases. We assume the on-chip *ÆTHERREAL* network to be very stable, and therefore network errors are not considered further.

The second major area of network verification focuses on bandwidth bottleneck detection and latency monitoring. This can be done either actively or passively. Active monitoring methods involve probing the network with test packets, in order to get round trip latency, peak bandwidth or available bandwidth. The problem with these methods is that they can introduce significant amounts of additional traffic in the network. On the internet, this intrusiveness of the debug traffic can be easily reduced by temporarily increasing the bandwidth with, for example, additional routers. Since this solution cannot be applied on a SOC, the intrusiveness cannot be easily removed, and hence complicates SOC debug.

Passive measurement methods execute performance measurements using special probe devices or probes added to routers, switches or hosts. The measured data is cached. This cached data can be either streamed to a central entity or shared within a local domain. How to best apply the lessons learned in computer networks to the *ÆTHEREAL* NOCs is an on-going research activity within Philips.

### 5.3 Verifying the NoC

In addition to using the standard verification techniques, an *ÆTHEREAL* NOC is verifying using special events that have been defined inside the network blocks. With these events, conditions such as incorrect configuration, incorrect topology (e.g. two ports of the router are switched), incorrect initialization, and reset errors can be detected and used to verify applications.

Examples of events in the NI block are: connection opened/closed, data for a connection received, data sent on a connection, and a certain data value appeared on a connection. Examples of events in the router block are: data with a certain path is passing, data in a queue for more than  $n$  cycles, incorrect path, and conflicts in reservation. These events can be sent either actively (self-initiating) or passively (polling). Within *ÆTHEREAL*, the approach is taken to temporary log events on-chip with a local time-stamp, and later, stream them off chip for analysis. Special debugger software uses the event information to graphically represents the state of the network at different levels of detail. This provides the user of the debugger software with very useful debug data. Furthermore the Codebook approach [15] is applied to the off-chip data to correlate the generated events and isolate the root cause of a particular problem.

When deciding to use the on-chip network for transport the events, a choice has to be made to make the implementation either completely independent of the NOC, or to over-dimension the existing network. Note that the latter approach leads to intrusiveness.

### 5.4 Verifying a SOC through its NOC

Verification of the SOC contains two parts: verification of the NOC, which is described in the previous section, and verification of the cores. In general the access to the cores is a problem. However, when using a NOC this access is simplified. In Figure 6 one of the possibilities is shown.

A processing block P3 receives data from another processing block P1 via the NOC. If in the verification of P3 a situation has to be reproduced, it is necessary to also repeat P1 and its predecessors. If data from P1 to P3 is first tapped off and streamed to memory M2, P1 needs not be executed during the repetition of P3 in following iterations. When upon replay P3 repeats its fault, the data can easily be streamed off chip and compared with the behavior of for example a FPGA model or a simulation model. Note that the exact timing of the data is lost during replay. This might lead to the disappearance of the problem, or the introduction of a new problem. Nevertheless, the disappearance of the problem hints at a timing-related problem, which also aids in debugging the application. When the timing, captured at P3, is enforced via specific reservations in the NOC, it is even possible to eliminate this time-intrusiveness. Experiments with varying timing behaviour of the input data can also help to locate the problem.

Beside transporting functional data to a core via the NOC, it is also possible to transport other data. By recording the entire state of a core in an embedded memory, it is possible to quickly restore this state via the NOC prior to replay.

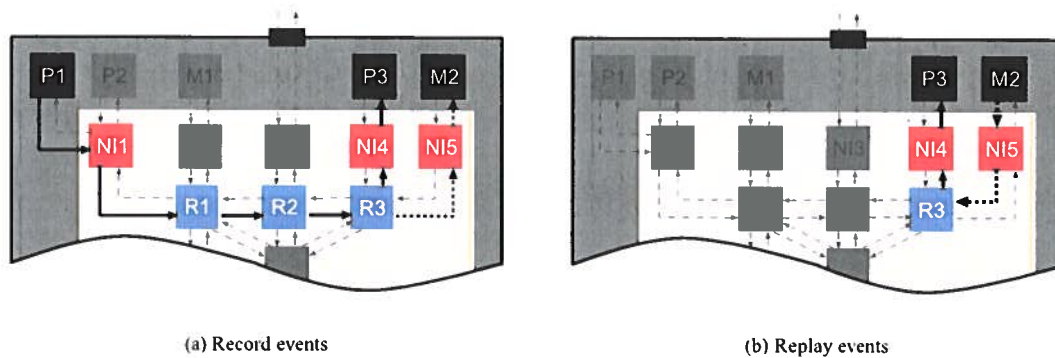


Figure 6: Recording a stream for easy replay in the receiving processing unit.

In this section we have shown that the NOC introduced new options for locating a problem during the verification of the cores. This type of verification is done only once per SOC design. In contrast, verification of an application that is mapped onto a SOC has to be conducted more often, and this is the topic of the next section.

## 6 Application Verification

Although the basic functionality of the SOC is verified using techniques from the previous sections, this by no means implies that the hardware and application software together will also run correctly. Examples of problems that we might detect only after we have mapped an application onto the SOC are:

- a processing core that writes into another core's memory space and thereby corrupts its operation,
- an incompatible format is used to exchange data between cores (e.g. endianness), and
- deadlines are not met because constraints were not passed to the network.

These bugs will become more difficult to find due to the increasing complexity of the SOC itself. Many cores run in parallel and the status of the system can no longer be related to a single program counter, or to traffic on a single bus. It is even possible that some of the cores are executing multi-threaded software and that these cores continue with those processes that can still consume or produce data. Lack of bandwidth, for example due to network congestion or functional processing spread, can cause an application to execute the processes in a different order. Some of the observed problems can be caused by the use of the NOC. However, there are also new opportunities for verifying the complete application using the NOC e.g., breakpoints, spying functional signals, and performance analysis.

- *Breakpoints* in the network can help to analyse the state of the SOC in more detail, as breakpoints can stop (part of) the application by either gating (some of) the on-chip clocks or by putting (part of) the cores in an idle mode. Once (part of) the application is stopped, the on-chip communication architecture can be used to access important registers and memories. Although in principle the  $\text{\textit{\text{ETHEREALS}}}$  NOC has a global notion of time and can therefore be stopped in relation to this global clock, this is by no means trivial for NOCs in general. Furthermore stopping a SOC is even more complex than stopping a NOC, as all cores can run at their own frequency. This is a topic of on-going research within Philips.
- *Spying functional data* in a local area network can be done relatively easy by plugging a network analyser on that link. This network analyser will monitor all data and process it into required debug or performance information. In a NOC, it is not possible to plug in a similar network analyser because

the data wires of the link are extremely difficult to probe. The two most important aspects we are interested in regarding a link are (1) the possible congestion on the link, and (2) the data from one of the connections that traverses the link. To achieve the first aspect, it is possible to add congestion monitors to the hardware design that can generate breakpoint events. To solve the second aspect, the NI can be configured such that the data is duplicated and sent along a separate debug channel. To reduce the tremendous amount of data that is generated during this functional spying, watchpoints can be introduced. Such a watch-point only gives an indication if a certain value has passed and should be generated inside the NI because, in general, the routers have no knowledge about the data.

- *Performance analysis* data and communication statistics, such as link utilisation, latency, and jitter, are important when debugging an application. One way to gather statistics on latency is to let both the sending and receiving NI blocks generate a event. From the sending NI, the moment at which the data is written in its queue is valuable data. At the receiving NI, this is either the time it arrives, or the time the core retrieves it. The person debugging the application should decide which one is most useful. This is a technique that is also very common in the verification of networks, as was described in Section 5.2

The latter two debug techniques lead to real-time generated data, which can be viewed as network events. These events are an addition to the network events defined in section 5.3, and they can be handled similarly.

## 7 Conclusions and future work

Today and in the future, SOCs will be used to implement high-performance networking applications. One of the issues to be addressed for these SOCs is their on-chip data communication. In this paper we presented the Philips ÆTHEREAL NOC for future-generation SOCs.

With the integration of a network on a SOC come additional test and verification requirements. Fortunately we can still use the wealth of test and verification methods that have already been successfully used in the past for either other existing SOCs and the much-larger communication networks, such as LANs or the Internet. The integrated network also provides new, and complementary possibilities to test and verify the SOC, and with it the SOC application. As shown in this paper, there are plenty of options for meeting a particular SOC's test and verification requirements.

In the initial phase of the ÆTHEREAL project, key elements of future investigation were defined, which include the challenge of stopping a SOC when cores run on their own frequency. Due to dynamic run-time effects (voltage drops, cross-talk, alpha particles, etc.), and their growing size NOCs evolve to computer networks. How to apply the verification lessons learned in computer networks to NOCs in a cost-effective way is another interesting (research) challenge that is currently under investigation within Philips.

In the future, a NOC will most likely become a similar commodity as its bigger brother the Internet, and no doubt equally successful.

## Acknowledgements

The authors like to thank their colleagues Harald Vranken and Jos Huisken and the anonymous reviewers for their valuable feedback on an early draft of this article.

## References

- [1] Drew Wingard. MicroNetworks-based integration for SOCs. In *Design Automation Conference*, 2001.
- [2] William J. Dally and Brian Towles. Route packets, not wires: On-chip interconnection networks. In *Design Automation Conference*, pages 684–689, June 2001.

- [3] Luca Benini and Giovanni De Micheli. Networks on chips: A new SoC paradigm. *IEEE Computer*, 35(1):70–80, 2002.
- [4] David Whclihan and Herman Schmit. Memory optimization in single chip network switch fabrics. In *Design Automation Conference*, June 2002.
- [5] HyperChip Inc. Cell-based switch fabric architecture. World International Property Organization Patent number WO 02/098066 A2, December 2002.
- [6] Faraydon Karim, Anh Nguyen, Sujit Dey, and Ramesh Rao. On-chip communication architecture for OC-768 network processors. In *Design Automation Conference*, June 2001.
- [7] Axel Jantsch and Hannu Tenhunen, editors. *Networks on Chip*. Kluwer, 2003.
- [8] E. Rijpkema, K. G. W. Goossens, A. Rădulescu, J. Dielissen, J. van Meerbergen, P. Wielage, and E. Waterlander. Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip. In *Proceedings of Design Automation and Test Conference in Europe*, March 2003.
- [9] Hui Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proceedings of the IEEE*, 83(10):1374–96, October 1995.
- [10] IEEE P1500 Web Site. <http://grouper.ieee.org/groups/1500/>.
- [11] H. Vranken, F. Meister, and H.-J. Wunderlich. Combining deterministic logic bist with test point insertion. In *Proceedings European Test Workshop*, pages 105–110, May 2002.
- [12] Don Douglas Josephson, Steve Poehlmann, and Vincent Govan. Debug Methodology for the McKinley Processor. In *Proceedings IEEE International Test Conference (ITC)*, pages 451–460, Baltimore, MD, October 2001.
- [13] H. Hao and R. Avra. Structured design-for-debug - the SuperSPARC-II methodology and implementation. In *Proceedings IEEE International Test Conference (ITC)*, pages 175–183, Washington, D.C., October 1995.
- [14] Bart Vermeulen, Tom Waayers, and Sandeep Goel. Core-based Scan Architecture for Silicon Debug. In *Proceedings IEEE International Test Conference (ITC)*, pages 638–647, Baltimore, MD, October 2002.
- [15] S. A. Yemini, S. Kliger, E. Mozes, Y Yemini, and D. Ohsie. High speed and robust event correlation. *IEEE Communications Magazine*, pages 82–90, May 1996.

# Exhibit L

# Concepts and Implementation of the Philips Network-on-Chip

John Dielissen, Andrei Rădulescu, Kees Goossens, and Edwin Rijpkema  
Philips Research Laboratories, Eindhoven, The Netherlands  
Email: john.dielissen@philips.com

## Abstract

*SoC communication infrastructures, such as the  $\mathcal{A}$ etheral network on chip (NoC), will play a central role in integrating IPs with diverse communication requirements. To achieve a compositional and predictable system design, it is essential to reduce uncertainties in the interconnect, such as throughput and latency. In our NoC, these uncertainties are eliminated by providing guaranteed throughput and latency services. Our NoC consists of routers and network interfaces. The routers provide reliable data transfer. The network interfaces implement, via connections, high-level services, such as transaction ordering, throughput and latency guarantees, and end-to-end flow control. The network interfaces also implement adapters to existing on-chip protocols, such as AXI, OCP and DTL, to seamlessly connect existing IP modules to the NoC. These services are implemented in hardware to achieve high speed, and low area. Our NoC provides run-time reconfiguration. We show that in the  $\mathcal{A}$ etheral NoC, this is achieved by using the NoC itself, instead of an additional control network. We present an instance of a 6-port router with an area of  $0.175\text{mm}^2$  after layout, and a network interface with 4 IP ports having a synthesized area of  $0.172\text{mm}^2$ . Both the router and the network interface are implemented in  $0.13\mu\text{m}$  technology, and run at 500 MHz.*

## 1 Introduction

As systems on chip (SoC) grow in complexity, the traditional on-chip interconnects, such as buses and switches, cannot be used anymore, due to their limited scalability. Networks on chip (NoC) scale better, and, therefore, they are a solution to large SoCs [2–5, 7, 9–11, 14].

NoCs offer well-defined interfaces [2, 8, 14, 16], decoupling computation from communication, and easing design. It has been shown that NoCs can provide interfaces to existing on-chip communication protocols, such as AXI [1], OCP [12], DTL [13], thus, enabling reuse of existing IP modules [8, 15].

A disadvantage of large interconnects in general (e.g., buses with bridges, or NoCs) is that they introduce uncertainties (e.g., due to contention). Applications also introduce uncertainties as they become more dynamic and heterogeneous. All these complicate integration, especially in hard real-time systems (e.g., video), as the user expects the resulting system to be predictable.

In the  $\mathcal{A}$ etheral NoC, we advocate the use of differentiated services and the use of guaranteed communication to eliminate uncertainties in the interconnect, and to ease integration [8]. We allow differentiated services by offering communication services on connections that can be configured individually for different services. Examples of properties that can be configured on a connection are

throughput and latency that can be configured to have no guarantees (i.e., best effort) or guaranteed for a particular bound. By providing guarantees, our NoC offers predictable communication, which is a first step in designing a predictable system.

In the next section, we present the  $\mathcal{A}$ etheral NoC, which offers both guaranteed and best-effort services. The NoC consists of routers and network interfaces. Our routers, described in Section 2.1, use input queuing, wormhole routing, link-level flow control and source routing. It has two traffic classes for the GT and BE data. For GT, time slots are reserved such that no contention occurs, while for BE, we use a round-robin arbitration to solve contention. The network interfaces, described in Section 2.2, have a modular design, composed of kernel and shells. The NI kernel provides the basic functionality, including arbitration between connections, ordering, end-to-end flow control, packetization, and a link protocol with the router. Shells implement (a) additional functionality, such as multicast and narrowcast connections, and (b) adapters to existing protocols, such as AXI or DTL. All these shells can be plugged in or left out at instantiation time according to the needs to optimize area cost.

The network connections are configurable at runtime via a memory-mapped configuration port. In Section 3, we show how the network is used to configure itself as opposed to using a separate control interconnect for network configuration.

## 2 Concepts of the Network

The network on chip as is exemplified by Figure 1, consists of two components: the routers and the network interfaces (NI). The routers can be randomly connected amongst themselves and to the network interfaces (i.e., there are no topology constraints). Note that in principle there can be multiple links between routers. The routers transport packets of data from one NI to another. The NIs are responsible for packetization/depacketization, for implementing the connections and services, and for offering a standard interface (e.g., AXI or OCP) to the IP modules connected to the NoC.

The  $\mathcal{A}$ etheral NoC provides both best-effort and guaranteed services (e.g., latency or throughput). To implement guarantees, we use *contention-free routing*, which is based on a time-division-multiplexed circuit-switching approach, where one or more circuits are set up for a connection [14]. This requires a logical notion of synchronicity, where all routers and NIs are in the same slot. Circuits are created by reserving consecutive slots in consecutive routers/NIs. This is, the circuits are pipelined, in the sense that if a circuit is set from router  $R$  to router  $R'$ , and slot  $s$  is reserved at router  $R$ , then slot  $s + 1$  must be reserved at router  $R'$ . On these circuits, data received in one slot will be forwarded to the next router/NI in the next slot. By setting up circuits, we ensure that data is transported without contention. In this way throughput

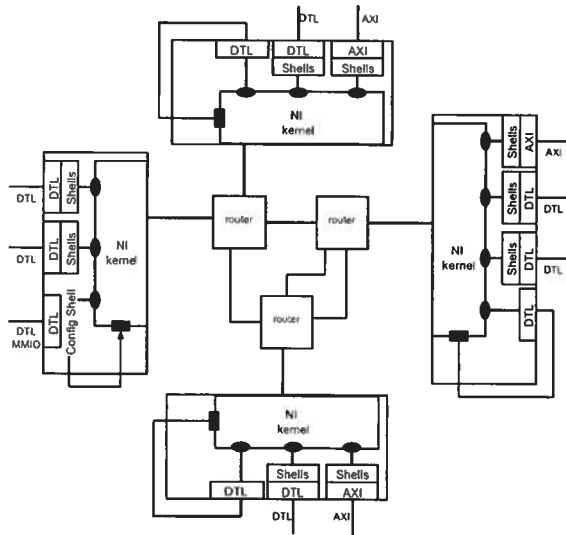


Figure 1. network example

and latency are guaranteed. We call this guaranteed traffic as guaranteed throughput (GT) data, as opposed to the best-effort (BE) data, for which no throughput guarantees are given.

As mentioned above, circuits are set up by reserving slots. These slots are reserved such that no more than one GT data is scheduled at the same time on an output port of a router or NI. BE data is transferred on the slots that are not used by the GT data: either the slots are unreserved, or the slots are reserved, but not used. BE data can be delayed because of the higher priority GT data, or because of contention on the ports.

In the following sections, we describe in detail the router and NI architectures.

## 2.1 Router Architecture

Routers send data from one network interface to the other by means of packets. Such a packet consists of one or more flits, where a flit is the minimal transmission unit. As a transmission scheme we use wormhole routing, because of the low cost (the buffer capacity can be less than the length of a packet) and low latency (the router can start forwarding the first flit of a packet without waiting for the tail). To reduce queuing capacity of a router, and thus the area, input queuing is used, as shown in Figure 2.

We select source routing as an addressing scheme, because it allows topology independence, while at the same time has a low cost: no expensive (programmable) lookup tables are needed in the router. In source routing the path on which the packet travels is included in the header of a packet. In *Æthereal*, this path is a list of destination ports, from which each router on the path removes the first element for its own use.

In the *Æthereal* network guarantees are given by statically calculating the GT schedule. In this way conflicts at the destination ports at each router can be avoided. In fact a pipelined circuit switching network is set-up. Since the network is distributed, also the circuit switching configuration, being the time at which GT-packets arrive and to which destination port they have to go, has to be distributed. In earlier versions of the *Æthereal* router [14], this was done in "local" slot tables. When programming a GT-

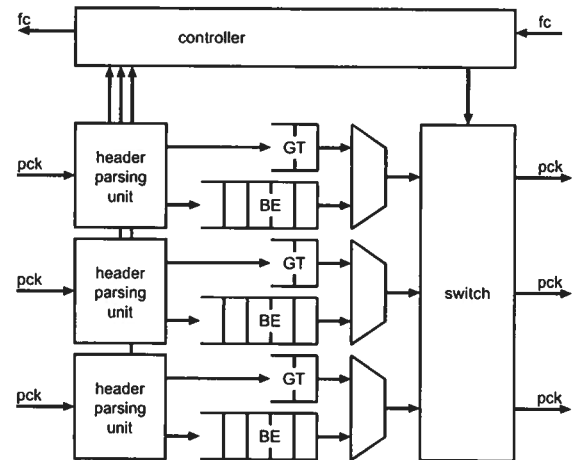


Figure 2. Router architecture

connection, all slot tables on the path are consulted to avoid conflicts in the schedule. In this way distributed programming is enabled, which is essential for large networks. However, for the next years we expect the network to be small and a centralized programming scheme is chosen, for which no "local" slot table is needed. The area cost of the "local" slot table is quite high because first of all, the table itself costs area (approximately 25% of the total area of a router with 6 bidirectional ports and 256 slots), and second the programming unit, and the connected additional port on the router have to be provided (an additional 25%). In this paper we present a NoC with centralized configuration. We include the switching configuration (the path) in the packet header, and, as a consequence the slot tables are removed from the routers.

Besides the path, the header also contains information for the Network interface, which is explained in section 2.2.1. As explained, a network packet is build up of one or more network flits. For the current *Æthereal* network, the flit size is chosen 3 to optimize the data clock frequency and control frequency. The information of the type of flit is annotated in the first element of the sideband information, being the id. The format of the flits is shown in Figure 3. The figure shows that the flit contains a header and 2 payload words. Only the first flit of a packet has a header and as a consequence, the next flits can have 3 payload words. Note that when packets consist of multiple flits, the overhead of the header is reduced. The amount of valid words in the flit is stored in the size field. The end of the packet is notified by the eop flag in the sideband information. As an example, Figure 4 shows how a packet, containing 10 payload words can be build up.

GT and BE-flits are semantically the same, but they are handled differently by the scheduler: GT-flits are always scheduled for the next cycle. The BE-flits are scheduled to the remaining destination ports according a round-robin schedule. Once a first flit of a BE-packet is send to a certain destination port, than port remains locked until the packet is finished: the port does not schedule BE-flits from the other input ports. In this way the interleaving of BE-packets is avoided, which makes implementation simple and cheap. Note that BE-packets can still be interleaved with GT-packets. For GT-flits the interleaving amongst themselves has to be avoided in the static schedule.

The router has a controller and a data path elements. In the data path, the input messages, from either routers or network interfaces

|      |           |     |      |
|------|-----------|-----|------|
| id   | credit    | qid | path |
| size | payload 0 |     |      |
| eop  | payload 1 |     |      |

**Figure 3. Network flits**

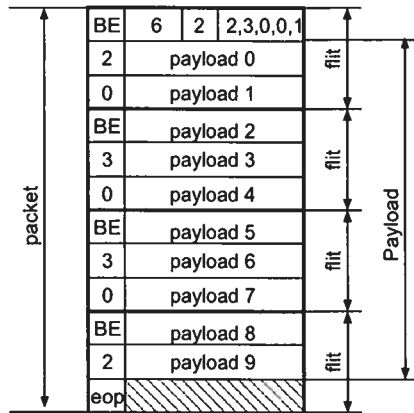


**Figure 4. BE-packet example**

are parsed by the header-parsing units (hpu). These units, shown in Figure 2, remove the first element for the path, send the parsed flits into GT or BE queues and notify the controller that there is a packet. The controller schedules flits for the next cycle. After scheduling the GT-flits, the remaining destination ports can serve the BE-flits. In the case of conflicts (e.g. two BE-flits address the same destination), a round-robin arbitration scheme is applied. The controller sets the switches in the right direction for the duration of the next flit cycle. Furthermore the read commands will be given to the fifo's.

To avoid overflow in the BE-input queues, a link-level flow control scheme is implemented. Each router is initialized with the amount of free space in the connected routers and network interfaces. Every time a flit is sent to a next router, the free space counter corresponding to that destination port is decremented. When a router schedules a flit for the next slot, it signals its predecessor that the free space counter can be incremented. Since GT-packets follow a pipelined circuit, a GT-flit is always sent to the next router in the next cycle, and therefore link-level flow control can be omitted.

### 2.1.1 Implementation

We synthesized and layouted in a  $0.13\mu\text{m}$  technology a prototype router with 6 bidirectional ports, and BE input queues of 32-bit wide and 24-word deep each (see Figure 5). In the floorplan the area-efficient custom-made hardware fifos, that we use for the BE and GT queues, are clearly visible. The design is fully testable using the well known scan-chain test method, and power stripes are included. The total area of the router sums up to  $0.175\text{mm}^2$ . The router runs at a frequency of 500 MHz, and delivers a bandwidth of 16 Gbit/s per link in each direction.

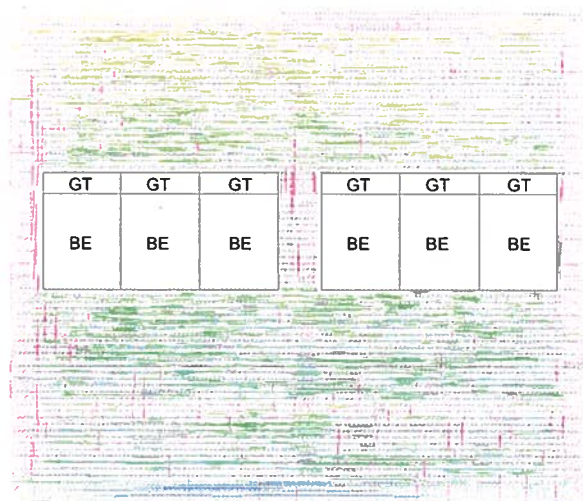


**Figure 5. Layout of a router with 6 bidirectional ports**

## 2.2 Network Interface Architecture

The network interface (NI) is the component that provides the conversion of the packet-based communication of the network to the higher-level protocol that IP modules use. We split the design of the network interface in two parts: (a) the *NI kernel*, which packetizes messages and schedules them to the routers, implements the end-to-end flow control, and the clock domain crossing, and (b) the *NI shells*, which implement the connections (e.g., narrowcast, multicast), transaction ordering, and other higher-level issues specific to the protocol offered to the IP. We describe the architectures of the NI kernel and the NI shells in the next two sections, and the results for their implementation in Section 2.2.3.

### 2.2.1 NI Kernel Architecture

The NI kernel (see Figure 6) receives and provides messages, which contain the data provided by the IP modules via their protocol after sequentialization. The message structure may vary depending on the protocol used by the IP module. However, the message structure is irrelevant for the NI kernel, as it just sees messages as pieces of data that must be transported over the NoC.

The NI kernel communicates with the NI shells via *ports*. At each port, peer-to-peer connections can be configured, their number being selected at NI instantiation time. A port can have multiple connection to allow differentiated traffic classes (e.g., best effort, or guaranteed throughput), in which case there are also *connid* signals to select on which connection a message is supplied or consumed.

For each connection, there are two message queues (one source queue, for messages going to the network, and one destination queue, for messages coming from the network) in the NI kernel. Their size is also selected at the NI instantiation time. Queues provide the clock domain crossing between the network and the IP modules. Each port can, therefore, have a different frequency.

Each channel is configured individually. In a first prototype of the  $\text{\AA}$ threal network interface, we can configure if a channel is

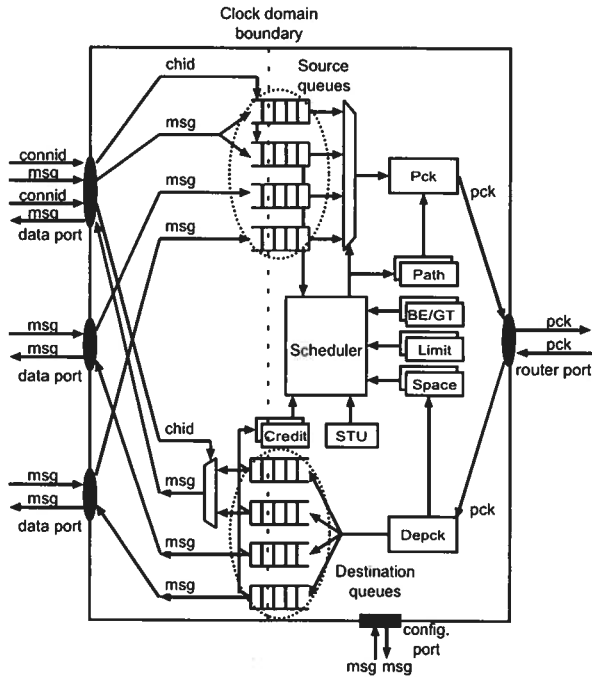


Figure 6. Network interface kernel

best effort (BE) or providing timing guarantees (GT), reserve slots in the latter case, configure the end-to-end flow control, and the routing information.

End-to-end flow control ensures that no data is sent unless there is enough space in the destination buffer to accommodate it. This is implemented using credits [17]. For each channel, there is a counter (*Space*) tracking the empty buffer space of the remote destination queue. This counter is configured with the remote buffer size. When data is sent from the source queue, the counter is decremented. When data is consumed by the IP module at the other side, credits are produced in a counter (*Credit*) to indicate that more empty space is available. These credits are sent to the producer of data to be added to its *Space* counter. In the *Æther*-real prototype, we piggyback credits in the header of the packets for the data in the other direction to improve network efficiency. Note that at most *Space* data items can be transmitted. We call *sendable data*, the minimum between the data items in the queue and the value in the counter *Space*.

From the source queues, data is packetized (*Pck*) and sent to the network via a single link. A packet header consists of the routing information (NI address for destination routing, and path for source routing), remote queue id (i.e., the queue of the remote network interface in which the data will be stored), and piggybacked credits (see Figure 3).

There are multiple channels which may require data transmission, we implement a scheduler to arbitrate between them. A queue becomes eligible for scheduling when either there is sendable data (i.e., there is data to be sent, and is space in the channel's destination buffer), or when there are credits to send. In this way, when there is no sendable data, it is still possible to send credits in an empty packet.

The scheduler checks if the current slot is reserved for a GT channel. If the slot is reserved and the GT channel is eligible for

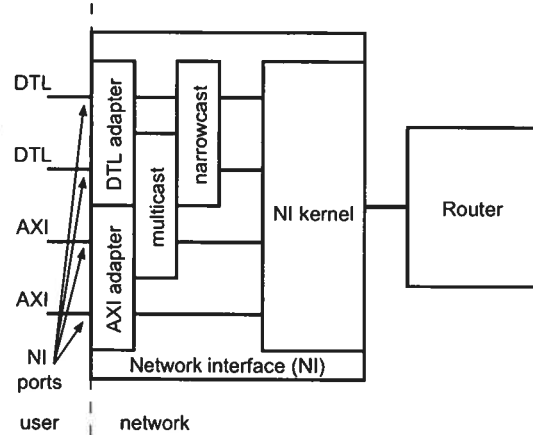


Figure 7. NI kernel and shells

scheduling, then the channel is granted data transmission. Otherwise, the scheduler selects an eligible BE channel using some arbitration scheme: e.g. round-robin, weighted round-robin, or based on the queue filling.

Once a queue is selected, a packet containing the largest possible amount of credits and data will be produced. The amount of credits is bound by implementation to the given number of bits in the packet header, and packets have a maximum length to avoid links being used exclusively by a packet/channel, leading to congestion.

On the outgoing path, packets are depacketized, credits are added to the counter *Space*, and data is stored in its corresponding queue, which is given by a queue id field in the header.

## 2.2.2 NI Shells: The interface to the IP

With the NI kernel described in the previous section, peer-to-peer connections (i.e., between one master and one slave) can be supported directly. These type of connections are useful in systems involving chains of modules communicating peer-to-peer with one another (e.g., video pixel processing [6]).

For more complex type of connections, such as narrowcast or multicast, and to provide conversions to other protocols, we add shells around the NI kernel. As an example, in Figure 7, we show a network interface with two DTL and two AXI ports. All ports provide peer-to-peer connections. In addition to this, the two DTL ports provide narrowcast connections, and one DTL and one AXI port provide multicast connections. Note that these shells add specific functionality, and can be plugged in or left out at design time according to the requirements. Network instantiation is simple, as we use an XML description to automatically generate the VHDL code for the NIs as well as for the network topology.

In Figures 8 and 9, we show a master and slave shells that implement a simplified version of a protocol such as AXI. The basic functionality of such a shell is to sequentialize commands and their flags, addresses, and write data in request messages, and to desequenzialize messages into read data, and write responses. Examples of the message structures (i.e., after sequentialization) passing from NI shells and NI kernel are shown in Figure 10.

In full-fledged master and slave shells, more blocks would be added to implement e.g., the unbuffered writes at the master side, and read linked, write conditional at the slave side.

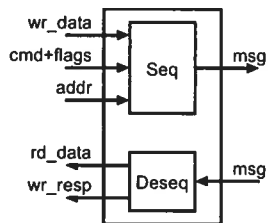


**Figure 8. Master shell**

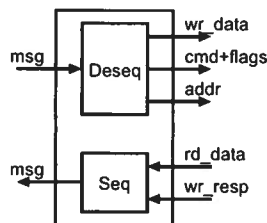


**Figure 9. Slave shell**

### 2.2.3 Implementation

We have synthesized an instance of a NI kernel with a slot table of 16 slots, and 4 ports having 1, 1, 2, and 4 channels, respectively, with all queues being 32-bit wide and 8-word deep. The queues are arca-efficient custom-made hardware fifos. We use these fifos instead of RAMs, because we need simultaneous access at all NI ports (possibly running at different speeds) as well as simultaneous read and write access for incoming and outgoing packets, which cannot be offered with a single RAM. Moreover, for the small queues needed in the NI, multiple RAMs have a too large area overhead. Furthermore the hardware fifos implement the clock domain boundary allowing each NI port to run at a different frequency. The rest of the NI kernel runs at a frequency of 500 MHz, and delivers a bandwidth towards the router of 16 Gbit/s in each direction. The synthesized area for this NI-kernel instance is  $0.13 \text{ mm}^2$  in a  $0.13\mu\text{m}$  technology.

Next to the kernel there are also a number of shells to implement one configuration port, two master ports, and one slave port. These shells add to the area another  $0.04 \text{ mm}^2$ , resulting in a total NI area of  $0.172 \text{ mm}^2$ .

## 3 Network Configuration

As mentioned in Section 2.1, in our prototype *Ethernet* network, we opt for centralized programming. This means that there is a single configuration module that configures the whole network, and that slot tables can be removed from the routers.

| cmd          | length | flags | seq no. | trans id |
|--------------|--------|-------|---------|----------|
| address      |        |       |         |          |
| write data 1 |        |       |         |          |
| ...          |        |       |         |          |
| write data N |        |       |         |          |

Request message format

| error       | seq no. | trans id |
|-------------|---------|----------|
| read data 1 |         |          |
| ...         |         |          |
| read data N |         |          |

Response message format

**Figure 10. Message format examples**

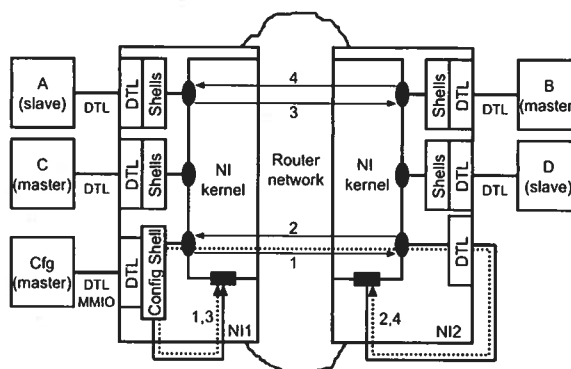


**Figure 11. NI configuration**

Consequently, only the NIs need to be programmed when opening/closing connections.

NIs are programmed via a configuration port (the DTL MMIO port on which the *Cfg* modules is connected). This port offers a memory-mapped view on all control registers in the NIs. This means that the registers in any NI are readable and writable using normal read and write transactions.

Configuration is performed using the network itself (i.e., there is no separate control interconnect needed for network programming). This is done by directly connecting the NI configuration ports to the network like any other slave (see NI2's configuration port in Figure 11).

At the configuration module *Cfg*'s NI, we introduce a configuration shell (*Config Shell*), which, based on the address configures the local NI (NI1), or sends configuration messages via the network to other NIs. The configuration shell optimizes away the need for an extra data port at NI1 to be connected to the NI1's configuration port.

In Figure 12, we show the necessary steps in setting up a connection between two modules (master B and slave A) from a configuration module (*Cfg*). Like for any other memory-mapped register, before sending configuration messages for programming the B to A connection, a connection to the remote NI must be set up. This involves two channels, one for the requests and one for the responses between NI1 and the configuration port of NI2. This connection is opened in two steps. First, the channel to the remote NI configuration port is set up by writing the necessary registers in NI1 (Step 1 in Figures 11 and 12). Second, we use this channel to set up (via the network) the channel from configuration port of NI2 to the configuration port of NI1 (Step 2). The three shown messages are delivered and executed in order at NI2. The last of them also requests an acknowledgment message to confirm that the channel has been successfully set up.

After these two configuration channels have been set up, the remote NI2 can be safely programmed. We can, therefore, proceed to setting up a connection from B to A. For programming NI2 (B's NI), the previously set up configuration connection is used. For programming NI1, the NI1's configuration port is accessed directly via *Config Shell*. First, the channel from the slave module A to the master module B is configured by programming NI1 (Step 3). Second, the channel from the master module B to the slave module A is configured (Step 4) through messages to NI2.

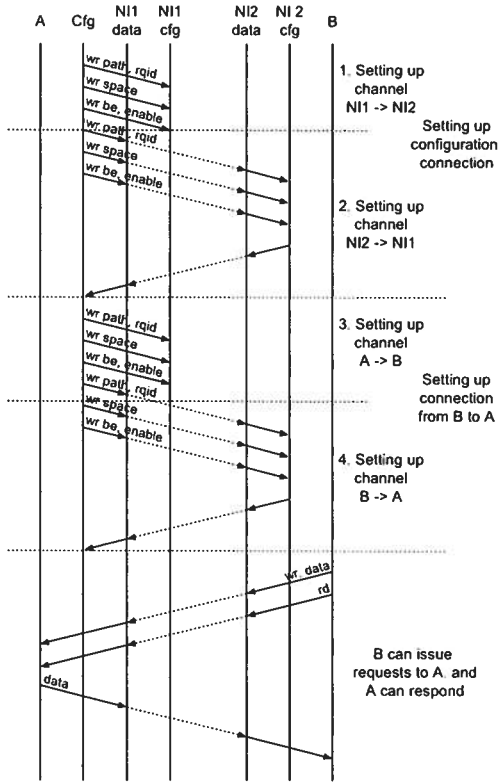


Figure 12. Connection configuration example

## 4 Conclusions

In this paper, we present the Æthereal network on chip, developed at the Philips Research Laboratories. This network offers, via connections, high-level services, such as transaction ordering, throughput and latency guarantees, and end-to-end flow control. The throughput/latency guarantees are implemented using pipelined time-division-multiplexed circuit-switching.

The network consists of routers and network interfaces. The routers use input queuing, wormhole routing, link-level flow control and source routing. It has two traffic classes for the GT and BE data. For GT, time slots are reserved such that no contention occurs, while for BE, we use a round-robin arbitration to solve contention. We show an instance of a router with 6 bidirectional ports, and BE input queues of 32-bit wide and 24-word deep each implemented using custom-made fifos. This router has an area of  $0.175\text{mm}^2$  after layout in  $0.13\mu\text{m}$  technology, and runs at 500 MHz. This has been achieved by omitting the slot tables, and making low area cost decisions at all levels.

The network interfaces have a modular design, composed of kernel and shells. The NI kernel provides the basic functionality, including arbitration between connections, ordering, end-to-end flow control, packetization, and a link protocol with the router. Shells implement (a) additional functionality, such as multicast and narrowcast connections, and (b) adaptors to existing protocols, such as AXI or DTL. All these shells can be plugged in or left out at instantiation time according to the needs to optimize area cost.

We show an instance of our network interface with a slot table

of 16 slots, and 4 ports having 1, 1, 2, and 4 channels, respectively. All queues are 32-bit wide and 8-word deep, and are implemented using custom-made fifos. These fifos also implement the clock domain boundary allowing NI ports to run at a different frequency than the network. The NI kernel runs at a frequency of 500 MHz. The synthesized area for the complete network interface is  $0.172\text{mm}^2$  in a  $0.13\mu\text{m}$  technology.

The network connections are configurable at runtime via a memory-mapped configuration port. We use the network to configure itself as opposed to using a separate control interconnect for network configuration.

In conclusion, we provide efficient network offering high-level services (including guarantees), which allows runtime network programming using the network itself.

## References

- [1] ARM. *AMBA AXI Protocol Specification*, June 2003.
- [2] L. Benini et al. Powering networks on chips. In *ISSS*, 2001.
- [3] L. Benini et al. Networks on chips: A new SoC paradigm. *IEEE Computer*, 35(1):70–80, 2002.
- [4] E. Bolotin et al. QNoC: QoS architecture and design process for network on chip. *Journal of Systems Architecture*, 49, Dec. 2003. Special issue on Networks on Chip.
- [5] W. J. Dally et al. Route packets, not wires: On-chip interconnection networks. In *DAC*, 2001.
- [6] O. P. Gangwal et al. Understanding video pixel processing applications for flexible implementations. In *Euromicro DSD*, 2003.
- [7] K. Goossens et al. Networks on silicon: Combining best-effort and guaranteed services. In *DATE*, 2002.
- [8] K. Goossens et al. Guaranteeing the quality of services in networks on chip. In J. Nurmi, H. Tenhunen, J. Isoaho, and A. Jantsch, editors, *Networks on Chip*, pages 61–82. Kluwer, 2003.
- [9] P. Guerrier et al. A generic architecture for on-chip packet-switched interconnections. In *DATE*, 2000.
- [10] F. Karim et al. An interconnect architecture for networking systems on chip. *IEEE Micro*, 22(5), 2002.
- [11] S. Kumar et al. A network on chip architecture and design methodology. In *ISVLSI*, 2002.
- [12] OCP International Partnership. *Open Core Protocol Specification. 2.0 Release Candidate*, 2003.
- [13] Philips Semiconductors. *Device Transaction Level (DTL) Protocol Specification. Version 2.2*, July 2002.
- [14] E. Rijpkema et al. Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip. In *DATE*, 2003.
- [15] A. Rădulescu et al. Communication services for networks on chip. In S. Bhattacharyya, E. Deprettere, and J. Teich, editors, *Domain-Specific Embedded Multiprocessors*. Marcel Dekker, 2003. to appear.
- [16] M. Sgroi et al. Addressing the system-on-a-chip interconnect woes through communication-based design. In *DAC*, 2001.
- [17] A. S. Tanenbaum. *Computer Networks*. Prentice Hall, 1996.

# Exhibit M

# Cost-Performance Trade-offs in Networks on Chip: A Simulation-Based Approach

Santiago Gonzalez Pestana, Edwin Rijpkema, Andrei Rădulescu, Kees Goossens and Om Prakash Gangwal  
Philips Research Laboratories, Eindhoven, The Netherlands  
E-mail: santiago.gonzalez.pestana@philips.com

## Abstract

*A challenge facing designers of systems on chip (SoC) containing networks on chip (NoC) is to find NoC instances that balance the cost (e.g. area) and performance (e.g. latency and throughput). In this paper we present a simulation-based approach to address this problem. We use XML to instantiate network components (routers, network interfaces) and their composition. NoCs are evaluated in terms of cost and performance by sweeping over different parameters (e.g. network topology, network interface queue depth). We then show, how we can obtain trade-off plots by using the results obtained with our simulation environment. Finally, by means of two examples we illustrate how trade-off plots can help the NoC designers in selecting the right network based on a set of different constraints.*

## 1 Introduction

The increasing complexity of systems-on-chip (SoC) together with the increasing wiring problems of newer IC technology generations make networks on chip (NoC) a promising replacement for busses and dedicated interconnect [3, 4, 9]. A NoC is an on-chip communication infrastructure that provides communication services to the IP blocks that connect to it. Examples of these services are connections with in-order data delivery and connections with end-to-end flow control.

NoCs are composed of *network interfaces* and *routers*. Network interfaces (NI) are the service access points of the network. These services are made available at one or more ports of the NIs. The routers provide the connectivity in the network; their job is to transport the data between the NIs.

The design of a NoC involves the design of NIs and routers together with the topology that specifies the way in which they are interconnected. The design space of designing NoCs is huge, spanned by the many parameters that describe the NIs, routers, and topology [13, 14]. Examples are the number of connections per NI, the depth of the queues in the routers, and the arity of a tree.

Chip design is typically characterized by tight cost constraints and high performance demands. Our goal is to help the designer in finding the right NoC. The right NoC is one that fulfills all cost and performance constraints and optimizes some other cost or performance metric. To fulfill the designer's need of making cost-performance trade-offs, we propose a method to obtain both cost and performance numbers of NoCs instances in the design space and a way to present these numbers.

We advocate an approach that relies on both VLSI study and simulation. The VLSI study is used to derive a model for the phys-

ical characteristics of a NoC, such as the silicon area and clock frequency of a NI as function of its parameters e.g. number of ports. Simulation then is used to generate performance numbers of the NoC as a whole. We use simulation for this because designing hardware for each NoC in the design space is unrealistic and, analytic methods make too many assumptions about the network and traffic to get accurate values for a real system.

To provide the performance numbers we have developed a simulator that can rapidly evaluate the performance of many NoC instances that are built from parameterized components. We automatically generate NoC instances for many points in the design space. These NoCs are specified in the XML, and their performance is evaluated without the need to recompile the simulator.

To prove our approach, the paper describes a case study in which synthetic workloads are offered to the NoC. Synthetic workloads are used because it makes the system parameterizable. In the case study we evaluate the cost and performance numbers of a number of network instances and show how to use these numbers to find the best NoC that meets the user requirement.

The paper is organized as follows. Section 2 describes that we span the NoC design space in terms of parameterized network components and topology. Section 3 describes our approach to obtain the cost and performance numbers that allow the user to make design trade-offs. Section 4 deals with the metrics that are of interest to the NoC designer and describes a simple NoC cost model. Sections 5 describes in detail the setup of a case study in which for a synthetic workload the best NoC must be found. Section 6 demonstrates our proposed approach: (1) cost and performance numbers are derived for a number of NoC instances, (2) the numbers are presented in a convenient way, and (3) the best network is selected for two example sets of requirements.

## 2 Network-on-chip design space

A network on chip (NoC) is composed of two types of components: routers (also called switches) and network interfaces (NI). Their composition is described by the topology in which these components are interconnected.

The NoC design space is spanned by choosing the router and NI architectures and the topology, and their corresponding parameters.

To illustrate the design space for a given router and NI architecture we use the network in Figure 1 as an example. The figure shows six IP blocks connected via four NIs to a four-router network having a mesh topology. In subsequent sections we describe the basic components and the topology to illustrate their parameterization.

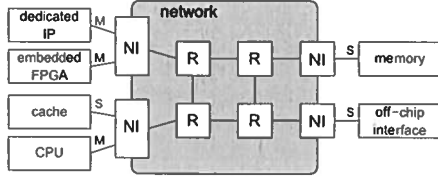


Figure 1. Example of SoC using a NoC.

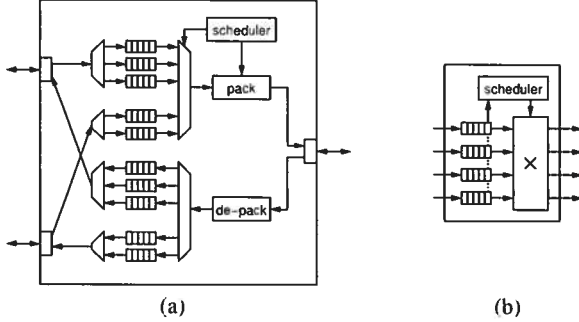


Figure 2. Example of NI architecture (a) and router architecture (b).

## 2.1 Network interface

A NI is described by two sets of parameters, i.e., parameters that describe the architecture of the NI and parameters that describe the instance of a given NI architecture. Two examples of the former set of parameters are whether services are connection based or not, and whether end-to-end flow-control is present or not. Clearly, what the set of the latter kind of parameters is, depends on what the actual architecture is. In this paper we focus on the latter set of parameters and fix the NI architecture to the .Ethereal NI described in [13], of which Figure 2(a) shows an example.

This NI architecture provides connection-based services made accessible at one or more ports. At the ports, shown at the left-hand side of Figure 2(a), a transaction-level protocol is used. That is, IPs connected to these ports issue *read* or *write*-like transactions. A write transaction consists of the write command together with the data. The NI considers the (write-command,data) pair as a message to be sent over the network. For a read, the NI considers the read command and the corresponding response of the data as two separate messages. The size of a message is expressed in bytes. The relation between connections and messages is that related messages, e.g. an audio stream of (write-command,data) messages from an audio decoder to an off-chip interface, are written in the same connection queue. For this queue, properties like in-order lossless delivery are given, but also guaranteed throughput (not dealt with in this paper, see [12]) can be provided. The messages in the queues are scheduled and are subsequently packetized to be transferred over the network of routers. The NI has two queues per connection, one for outgoing and one for incoming traffic. The NoC services are accessible via one or more ports of the NI and there are one or more connections per port.

The parameters describing the instance of the NI architecture are: number of ports, number of connections for each of these ports individually, and depths of the queues for each of the connections individually. Moreover, the depths of the incoming and outgoing queues is parameterized individually as well. The NI instance in Figure 2(a) has two ports, three connections for the first

port and two connections for the second port. The depths of all the queues is set to five words, where a word is four bytes.

## 2.2 Router

A router has two sets of parameters similar to the NI. The set of parameters that describe the architecture include routing algorithm, routing mode, buffering architecture, and schedule algorithm. In this paper we fix the architecture to the .Ethereal router architecture described in [11] of which an instance is shown in Figure 2(b). We only are interested in the Best Effort (BE) part of that router architecture.

The router is packet-switched and uses source routing, i.e. the routers use the route information that the NIs have put in the packets. It uses wormhole routing and input queuing.

The arity of the router is the number of input ports, which we take equal to the number of output ports. At every input port there is a queue that has a parameterized depth. So, parameters that describe the instance of the router architecture are arity and depth. The router instance in Figure 2(b) has arity four, and has queues of five words deep.

## 2.3 Network on chip

A NoC is composed of NI and router instances. To complete the NoC description we also must know the topology in which components are interconnected. In this paper we focus on regular topologies i.e., 2D-meshes and  $k$ -ary trees and  $k$ -ary fat-trees. Each of these topologies is parameterized by its own set of parameters. A 2D-mesh is parameterized in its width and height, and a tree and fat-tree by their arity and height [16].

In the network topologies envisaged in this paper we make the following assumptions. In all topologies the links are bidirectional. In meshes there is a NI at every router, and in  $k$ -ary (fat-)trees there are  $k$  NIs at every leaf router.

Note that the NoC design space as a whole is spanned by the selection of the topology and the NI and router architectures, and their respective parameters.

## 3 Networks on Chip design approach

In this section we propose a network-on-chip design-flow approach that guides the designer through the SoC design process allowing him to concentrate on the performance results. Performance results are obtained by using simulation results. Later in this section we address in detail the interface with the simulator in order to follow the proposed design approach. We use SystemC to [1] to build our simulator.

SystemC provides us different abstraction levels together with the flexibility to model, develop and simulate all parts composing a network on chip (e.g. routers, network interfaces) and external IP blocks connected to it. A number of simulators and design approach tuned to different applications (e.g. real time, shared-memory) can be found in the literature [7, 9, 15, 17]. A new simulator environment has been developed to provide a flexible compositional platform to experiment with a large number of components and their parameters. The environment natively is not oriented to any specific application or topology. Different applications using different services can be mapped without changing the system core. NoC parameters (defined by the user) define the NoC behavior. We use a transaction-level model (TLM), which allows us to describe hardware system components at a high level of abstraction. Although the results are less accurate than using register-transfer-level (RTL) simulations [10], we obtain higher simulation speed.

The simulator itself models the NoC at the level of flits, giving a good trade-off between simulation accuracy and speed.

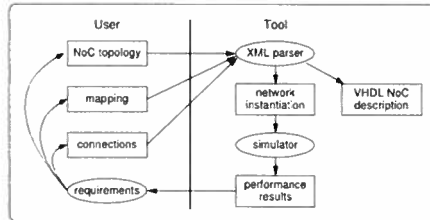


Figure 3. Design flow used in our environment.

To provide maximal flexibility to the designer in the parameter description, we propose the design flow shown in Figure 3. The designer concentrates on specifying three files (topology, mapping and connection) that are used by the XML parser to instantiate the SoC to be simulated. They describe the network and IP blocks along with their parameters. As an output we obtain a set of files containing performance results and statistics from the network. The designer can evaluate each NoC independently or automate the design flow by using our simulator to evaluate a large number of topologies, analyze them and select the most appropriate for a given set of constraints.

If the constraints are not (satisfactory) met (e.g. cost-performance requirements) the NoC designer can steer design decisions (e.g. mapping). In the next section we explain how we instantiate a general SoC design with the three files, which contain the parameters of all components and how they are interconnected in the NoC.

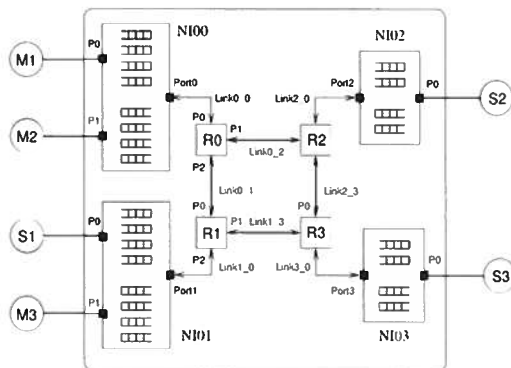


Figure 4. Example of SoC.

### 3.1 Interface to the simulator

The interface to the simulator environment is based on three files. The files are described using the Extensible Markup Language (XML). XML has among other features the advantages of being a standard language, it is suitable to organize and structure data, it is human readable, and parsers are available to automate the instantiation process. The XML system-on-chip description provided by the user is also used to generate VHDL code to have consistent views of the simulator and hardware implementation. The main motivation for using an XML-based description of NoC is twofold (a) flexibility to describe NoC components and (b) avoiding long re-compilation times, because we instantiate components at run time.

The use of three files to describe every component and their interrelations allow us to come with a flexible and well-defined

way to decouple network infrastructure components (e.g. routers, network interfaces) from user-defined modules (e.g. traffic generators, IP blocks). A set of basic components to experiment with (e.g. routers, network interfaces and traffic generators) are provided. Additionally, the user can supply a library of IP blocks or network elements that can be configured by means of attributes in the XML file and be instantiated at run time. The description and content of the following files use Figure 4 as an example.

**Topology file:** It contains the NoC topology description. We specify the routers and network interfaces, and the way they are interconnected (eg. mesh, ring, fat tree). The following parameters are given in the description. For the router, the type of router (e.g. input queuing, virtual output queuing), queue depth, number of input/output ports. For the network interface we specify the number of network interface ports, the number of connections and the router to which the network interface is connected.

In Figure 5 we can see part of the topology definition for the system shown in Figure 4. We have defined router “R0”, which is a 3-arity input-queued router, with a queue depth of 12 words. From the description see that “Link0\_0” connects network interface “NI00” to “R0” (links are bidirectional). A network interface can connect more than one IP block. In our example “NI00” contains one master<sup>1</sup> IP block as well as one slave IP block.

```
<RouterIQ id="R0" iq="12">
  <R_Port id="P0" link="Link0_0"/>
  <R_Port id="P1" link="Link0_2"/>
  <R_Port id="P2" link="Link0_1"/>
</RouterIQ>
<NI id="NI00">
  <NI_Port id="Port0" link="Link0_0"/>
  <MasterNIP id="P0" connections="2"/>
  <SlaveNIP id="P1" connections="2"/>
</NI>
<NI id="NI02">
  <NI_Port id="Port2" link="Link2_0"/>
  <SlaveNIP id="P0" connections="2"/>
</NI>
```

Figure 5. Example of a topology description.

**Mapping file:** Once the topology has been selected, we have to describe how IP blocks are going to be connected to the NoC. The mapping file provides a description of which blocks are connected to which network interface. In the same way as in the topology file, XML attributes are used to describe parameters that can be modified easily without the need of changing the simulator code.

Part of the XML mapping file is shown in Figure 6. We have declared a master module of type “TG” (Traffic Generator) with some extra parameters such as the type of traffic distribution (normal distribution), speed, or message size. This master module is connected to the NI “NI00” via the port “P0”.

```
<Master id="M1" type="TG" DIST="Nor" SPEED="8" SZ="128">
  <MasterNIP ni="NI00" id="P0"/>
</Master>
<Slave id="S3" type="Memory">
  <SlaveNIP ni="NI03" id="P0"/>
</Slave>
```

Figure 6. Mapping description example.

**Connection File:** To characterize the system completely, a set of connections is specified. Connections are established between one master network interface port and one or more slave network

<sup>1</sup>A master starts a transaction which is executed by the slave.

interface ports. Currently static connections are used, but in the future, our simulation environment will allow dynamic handling of connections. We use connections to connect one master IP block to one slave IP block; although other combinations are possible [12] we only use the most simple of them. Connections are opened with a set of properties (e.g. message integrity, transaction completion, transaction ordering) and parameters (e.g. routing path). The simulator provides a shortest path routing algorithm in case no path is specified in the file.

Figure 7 shows several connection configurations. Masters and slaves have their own connections identifiers for a given connection. As an example we show a connection between M2 and S3, the master identifies that connection with id 3 and the slave with id 2. The channel properties are Guaranteed Throughput (GT) for outgoing traffic and Best Effort (BE) for incoming traffic. For GT traffic we have to specify the slots we reserve to obtain the required bandwidth/latency guarantees (e.g slots="2 5"). Although we have flexibility to use GT traffic, in our experiments we only consider BE traffic.

```
<Connection master="M2" cidm="3" slave="S3" cids="2">
  <Request type="GT" slots="2 5" path="1 0"/>
  <Response type="BE" path="1 0"/>
</Connection>
<Connection master="M3" cidm="2" slave="S2" cids="3">
</Connection>
```

Figure 7. Part of the description of a connection file.

## 4 Cost and performance metrics

Due to the large design space of NoCs, obtaining a good balance of parameters such as cost and performance is a major challenge [2, 5, 8]. To find this balance the system designer needs to trade off cost and performance numbers to find, for example, the lowest-cost NoC that meets all performance requirements or the NoC that provides the highest throughput but still meets latency and area constraints.

What metrics should be optimized normally depend on the application that makes use of the NoC. For example, in portable devices the main concern typically is area and/or power, whereas for high-end video processing throughput and latency are more important.

In previous section we described how to obtain performance numbers, but we also must answer the question what exactly has to be traded off. This means that we need to clearly define the cost and performance metrics. The cost and performance numbers depend on the selected network components and the topology, and the setting of their corresponding parameters. Section 4.1 deals with cost metrics and introduces a simple cost model. Section 4.2 deals with performance metrics.

### 4.1 Cost metrics and cost model

NoC cost metrics relate to the physical aspects of the NoC. Examples are silicon area, power consumption, and bit-error rate due to cross-talk and external influences. As cost modeling is work in progress, we use for this paper a simple cost model for the silicon area metric. Silicon area is measured in  $mm^2$ .

To model the area of the NoC we use a zero wire cost model, that is, the area of the NoC is taken to be equal to total area of its components. The wires area caused by interconnecting the components are thus ignored in this paper.

In Section 2 we have seen that the network components, i.e. the routers and NIs, are parameterized. To allow a rapid evaluation of

the NoC area, we propose to derive the area of the router and NI as function of their parameters. For a particular NoC instance all parameters are fixed and these values then can be used in the area functions of the individual components.

The area functions are obtained by synthesis of RTL models of the router and NI architecture. We have done this for the architectures introduced in Section 2 as follows. We have a basic assumption on the form of the cost function as function of the parameters of interest. For example consider the router area as function of its arity. There are parts of the router whose area is linear in the arity (e.g. queuing) and parts that are quadratic in the arity (e.g. the switch inside the router), e.g. there is a part of the router that is linear in the arity (the control and queuing path) and a part that is quadratic in the arity (the switch). So, we expect the area to be of the form  $\alpha_1 \cdot a^2 + \alpha_2 \cdot a + \alpha_3$ . By synthesizing for several values of the arity we compute the best values for  $\alpha_1$ ,  $\alpha_2$  and  $\alpha_3$ .

A router having 48 words input queue depth has been synthesized in a  $0.13\mu m$  CMOS process for a clock frequency of 500 MHz and data path width of 32 bits (for each link/port). The resulting cost model given in terms of arity  $a$  is:

$$A_R(a) = 0.808a^2 + 23a \quad (10^{-3}mm^2) \quad (1)$$

Similarly, let  $p$  be the number of ports of the NI,  $c$  be the number of connections per port, and  $q$  be the depths of the queues in the NI. By synthesizing for several values of number of ports, connections per port, and queue depths  $q$  (with a frequency of 500 MHz and 32 bits wide data path) we derived the following area function.

$$A_{NI}(p, c, q) = 19.6pc + 0.72pcq + 4.8 \quad (10^{-3}mm^2) \quad (2)$$

### 4.2 Performance Metrics

There are several performance metrics that may be of importance. For the system designer that must select a NoC these are the metrics that reflect the total system behavior at the level of the offered services. Those are, throughput offered at the NI ports and end-to-end message latency.

**Throughput** is measured at the ports of the network. It is the average amount of user data that is accepted by the network on that port in a certain amount of time. Aggregate throughput is the sum of the throughputs at all network ports. Throughput is measured in bytes per second.

**Latency** is defined as the difference between the time at which the first word of a message has been offered to the network and the time the last word has been delivered to the destination. That is the so-called *total latency* ( $L_T$ ). Total latency is composed of the **waiting latency** ( $L_w$ ) and **network latency** ( $L_{net}$ ). If the NI has no space to accept data coming from the IP, the data is queued outside the network. The **offered load** (data produced by the IP) may be higher than the **accepted load** (i.e. accepted by the NI), and  $L_w \geq 0$ .

## 5 Case study: set-up

In this and the next section we use a case study to demonstrate our approach. The goal is to find the best network that meets the user requirements for a given application. We first describe the application model that we use. Then we describe the portion of the design space that we want to explore. Finally the cost and performance numbers are presented and their usefulness in selecting the right network is demonstrated.

## 5.1 Application model

In this case study the focus is on data streaming. To allow a simple scaling of the application, we have chosen a synthetic workload. This workload is generated by *traffic generators*. A traffic generator is the combination of a *traffic master* and a *traffic slave*.

In this paper we fix the write transactions data payload to 8 bytes. When these transaction are issued and to whom is described by the *temporal* and *spatial distribution*, respectively. In order to let the network not influence these distributions a master has a (logically) infinite buffer for each slave it communicates with. We also assume that slaves are infinite fast to avoid IP-NoC interactions (e.g. shortage of flow-control credits).

The temporal distribution describes the time at which new write transactions are issued. We use a normal distribution  $N(\mu, \sigma)$ , where  $\mu$  is the mean inter-arrival rate and  $\sigma$  is the variance. The mean  $\mu$  is directly derived from the required throughput, e.g., to let the traffic master produce data at a rate of 200 Mbyte/s, the value of  $\mu$  is  $200/8 = 25$  mega transaction per second. We further fix the variance  $\sigma$  to 10% to create some jitter in the data production rate.

The spatial distribution of a traffic master describes how the transactions are distributed toward the slaves. We use a uniform distribution.

To connect a single traffic generator to a network interface we require two network interface ports. Each traffic master issues write transactions to the traffic slave of every other traffic generator. The masters communicate to the slaves via connections. There is thus a single connection for each master-slave pair.

## 5.2 Experiments to realize

In this paper there are 16 traffic generators that communicate as just described. Note that we thus require  $16 \cdot 15 = 240$  connections which is more than what is found for typical applications of this size. Since each traffic generator requires two network interface ports, we thus must find the best network that has 32 network interface ports.

To find this network we will vary three free NoC parameters. i.e., network topology, number of ports per NI, and depth of the buffers in the NIs. Thus, a NoC is completely characterized by the triple  $\langle \text{topology}, \# \text{NIPs}, \text{queue depth} \rangle$  and we refer to them using this notation. E.g., the mesh network instantiated with 4 ports per NI and queue depth of 8 is referred to as  $\langle \text{mesh}, 4, 8 \rangle$ .

The topologies we evaluate are meshes, binary trees, and fat-trees. Because the number of total ports is constant (32), the size of these topologies depends on the number of ports per network interface. The number of ports per network interface does not significantly change the total NI cost but it does influence the size of the network. We evaluate cost and performance for networks with 2 and 4 ports per network interface.

The buffer depth in the NIs plays a paramount role both in cost and performance. We evaluate cost and performance with NI queues with a depth of 4 and 8 words.

For a mesh with four ports per NI we take a  $3 \times 3$ -mesh and do not connect an NI to one of the routers at the corner of the mesh, this gives us a total of 8 NIs and thus 32 NI ports in total (assuming one NI per router). With two ports per NI we take a  $4 \times 4$ -mesh, having one NI per router, and 2 ports per NI thus in total we have 32 NI ports in total.

For the binary (fat)-tree with four ports per NI we need three levels of routers. With two ports per NI, we need four levels. For

both topologies we assume there are two NIs per router at the bottom of the tree.

## 6 Case study: results

In this section we illustrate with an example the design approach and methodology that we proposed in this paper. We use our cost model and simulation environment to rapidly generate cost and performance numbers for the cases showed in previous section. To allow the selection of the right network the total set of cost and performance numbers are presented in a single scatter plot. In order to understand the trade-offs that are captured in the scatter plot we first present performance numbers of a single topology for different number of NI ports and different NI queue depths. Moreover, these performance numbers also explain how the scatter plot is obtained.

We present NoC performance in the so-called Chaos Normal Form (CNF) [6]. In CNF performance is presented by pair of plots: one being the *total accepted load* versus *total offered load* plot and the other being the *latency* versus *total offered load*. Total offered load is the sum of the loads offered at every network interface port, and total accepted load is the part of the offered load that the NIs actually accept.

Figures 8 and 9 show the performance numbers of the  $\langle \text{mesh}, p, d \rangle$  for  $p = 2, 4$  and  $d = 4, 8$ . In Figure 8, all curves start linearly with accepted load equal to offered load. The point at which the curves start to bend is the saturation point of the NoCs. We observe that both fewer ports per NI and deeper queues in the NI results in a higher saturation point and hence total accepted load.

In Figure 9 we can observe a steep increase in latency close to the saturation point. Typically networks are used below the saturation point to obtain acceptable latency. The pair of CNF plots can be used to obtain the maximum throughput for a given latency constraint. Figure 9 is used to find the offered load given the latency. The accepted load for this latency is equal to the offered load when we are below the saturation point.

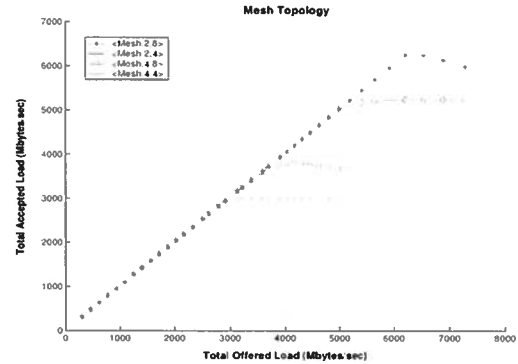


Figure 8. Total accepted load versus Total offered load for different mesh instances.

By using Figures 8 and 9 for our three topologies of interest we have derived a scatter plot as it is shown in Figure 10. For various latency bounds we use the NoC performance in CNF to obtain the accepted load. For each network we use the cost model to compute the NoC area. As each different NoC has its own area, the horizontal axis plots both area ( $mm^2$ ) and NoC instance. We

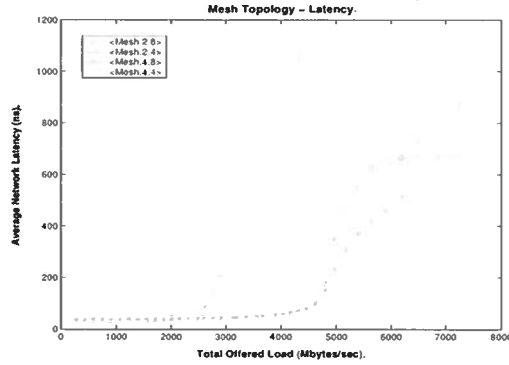


Figure 9. Average Network Latency versus Total offered load for different mesh instances.

have joined all points corresponding to the same latency constraint by a line for readability.

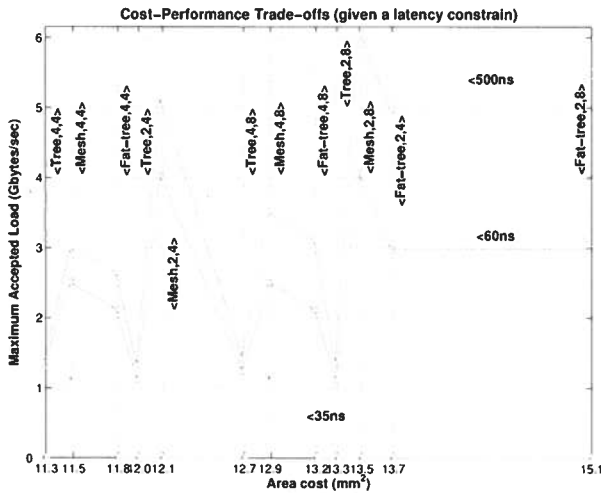


Figure 10. Cost performance trade-off.

Due to the all-to-all communication pattern of the case study, the total number of buffers is quadratical in the total number of NI ports, hence the cost figures plotted in Figure 10 are unrealistically high but they can be used to illustrate the method. In typical applications we expect an almost linear dependence, and therefore lower cost.

The user can achieve cost-performance trade-offs in several ways. Typically, a system designer must meet a set of constraints and wants to optimize for some other parameters. Below we give two examples and use the scatter plot in Figure 10 to find the best network.

**Example 1:** Find the lowest area network with the constraints that it provides a total accepted load of at least 2Gbyte/s with an average latency of at most 500ns. From the 500ns curve we select the left most point above the load=2Gbyte/s line, i.e. <mesh,4,4>. Note that because our simulation environment collects data as function of offered load, we can extract more latency bounds than the ones represented in Figure 10, without the need of re-simulation.

**Example 2:** Find the NoC that provides the highest accepted load, constrained by an area of  $12.5\text{mm}^2$  and an average latency

of 60ns. From the 60ns curve we select the highest accepted load at the left of the area= $12.5\text{mm}^2$  line, i.e. <mesh,2,4> giving an accepted load of 4Gbyte/s.

## 7 Conclusions

In this paper, we propose a method to help the designer in finding the right NoC given a set of constraints. To do so, we first introduce cost and performance metrics that allow a quantitative evaluation of NoCs. We focus on the metrics that are visible to the NoC user, such as silicon area, latency, and offered/accepted load.

To obtain both cost and performance numbers we use VLSI study to provide NI and router areas as functions of arity and buffer sizes, and number of ports and connections and buffer sizes, respectively. NoC cost is obtained by summing (assuming zero-cost wires) the cost of the NIs and routers. Performance numbers (latency and throughput) are obtained using simulations. The simulator is optimized for fast simulation of many NoC instances; it uses run-time network instantiation, and both transaction-level modeling and flit-level simulation.

The performance numbers obtained from the simulation and cost numbers from the cost model are used to demonstrate the trade-offs of performance (throughput and latency) versus area. The cost-performance numbers allow the trade-offs for various requirements. We demonstrate this with a case study in which the best network must be selected for a synthetic workload.

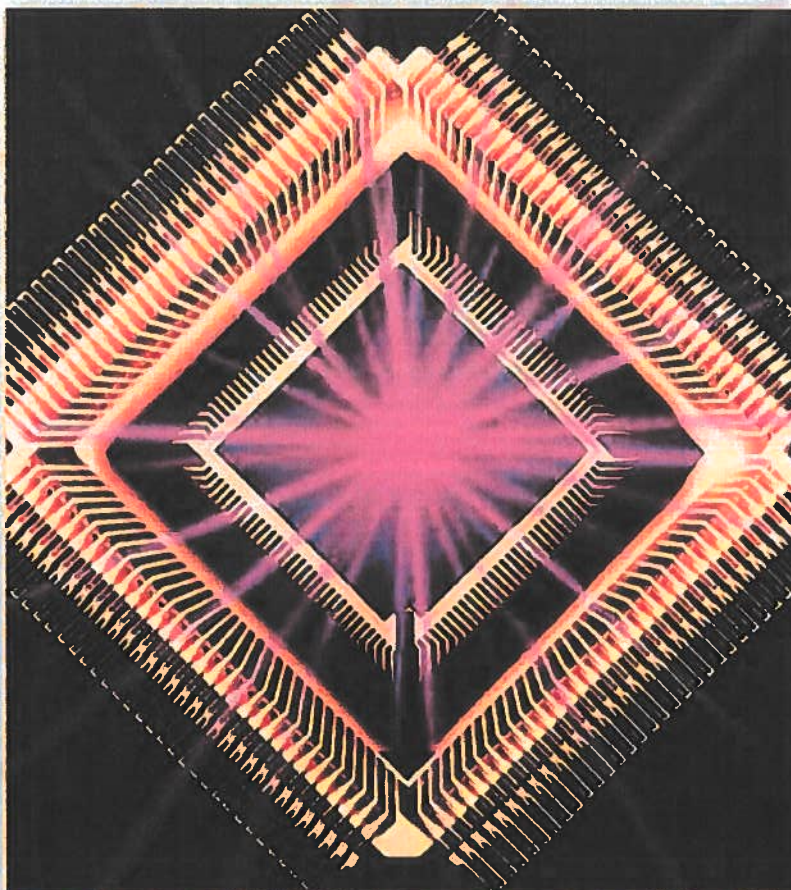
## References

- [1] *Functional Specification for SystemC 2.0*. www.systemc.org, 2001.
- [2] G. Apostolopoulos et al. Quality of service based routing: a performance perspective. In *SIGCOMM '98*, pages 17–28, 1998.
- [3] L. Benini et al. Powering networks on chips. *ISSS*, pages 33–38, 2001.
- [4] L. Benini et al. Networks on chips: A new SoC paradigm. *IEEE Computer*, 35(1):70–80, 2002.
- [5] E. Bolotin et al. QNoC: QoS architecture and design process for network on chip. *The Journal of Systems Architecture*, Dec. 2003.
- [6] J. Duato et al. *Interconnection Networks, An Engineering Approach*. Morgan Kaufmann, 2003.
- [7] M. Forsell. Advanced simulation environment for shared memory network-on-chips. In *20th IEEE Norchip Conference*, 2002.
- [8] J. Hu et al. Energy-aware mapping for tile-based NoC architectures under performance constraints. In *ASP-DAC 2003*, 2003.
- [9] S. Kumar et al. A network on chip architecture and design methodology. In *ISVLSI*, 2002.
- [10] S. Pasricha. Transaction level modelling of SoC with SystemC 2.0. 2003.
- [11] E. Rijkema et al. Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip. *DATE*, 2003.
- [12] A. Rădulescu et al. Communication services for networks on chip. In S. Bhattacharyya, E. Deprettere, and J. Teich, editors, *Domain-Specific Embedded Multiprocessors*. Marcel Dekker, 2004.
- [13] A. Rădulescu et al. An efficient on-chip network interface offering guaranteed services, shared-memory abstraction, and flexible network programming. In *DATE*, 2004.
- [14] I. Saastamoinen et al. Buffer implementation for Proteo network-on-chip. In *ISCA'S*, pages 113–116, vol.2, 2003.
- [15] G. Varatkar et al. Traffic analysis for on-chip networks design of multimedia applications. In *DAC*, 2002.
- [16] A. Varma and C. Raghavendra. *Interconnection Networks for Multiprocessors and Multicomputers: Theory and Practice*. 1994.
- [17] D. Wiklund et al. SoCBUS: Switched network on chip for hard real time embedded systems. In *IPDPS*, 2003.

# Exhibit N

# *Xpipes:* A Network-on-Chip Architecture for Gigascale Systems-on-Chip

Davide Bertozzi and Luca Benini



© DIGITAL STOCK

## Abstract

The growing complexity of embedded multi-processor architectures for digital media processing will soon require highly scalable communication infrastructures. Packet switched Networks-on-Chip (NoC) have been proposed to support the trend for Systems-on-Chip integration. In this paper, an advanced NoC architecture, called Xpipes, targeting high performance and reliable communication for on-chip multi-processors is introduced. It consists of a library of soft macros (switches, network interfaces and links) that are design-time composable and tunable so that domain-specific heterogeneous architectures can be instantiated and synthesized. Links can be pipelined with a flexible number of stages to decouple link throughput from its length and to get arbitrary topologies. Moreover, a tool called XpipesCompiler, which automatically instantiates a customized NoC from the library of soft network components, is used in this paper to test the Xpipes-based synthesis flow for domain-specific communication architectures.

**Keywords:** Systems-on-Chip, Networks-on-chip, pipelining, reliability, soft macros, network instantiation

## Introduction

The integration of an entire system onto the same silicon die (*System-on-Chip, SoC*) has become technically feasible as an effect of the increasing integration densities made available by deep sub-micron technologies and of the computational requirements of the most aggressive applications in the multimedia, automotive and ambient intelligence domain.

SoCs represent high-complexity, high-value semiconductor products that incorporate building blocks from multiple sources (either in-house made or externally supplied): in particular, general-purpose fully programmable processors, co-processors, DSPs, dedicated hardware accelerators, memory blocks, I/O blocks, etc. Even though commercial products currently exhibit only a few integrated cores (e.g., a RISC general purpose MIPS core and a VLIW Trimedia processor for the Philips Nexpria platform for multimedia applications), in the next few years technology will allow the integration of thousands of cores, making a large computational power available.

In contrast to past projections, which assumed that technology advances only needed to be linear and that all semiconductor products would deploy them, today the introduction of new technology solutions is increasingly application driven [1]. As an example, let us consider ambient intelligence, which is considered the new paradigm for consumer electronics. Systems designed for ambient intelligence will be based on high-speed digital signal processing, with computational loads ranging from 10 MOPS for lightweight audio processing, 3 GOPS for video processing, 20 GOPS for multilingual conversation interfaces and up to 1 TOPS for synthetic video generation. This computational challenge will have to be addressed at manageable power levels and affordable costs [2], and a single processor will not suffice, thus driving the development of more and more complex *multi-processor SoCs (MPSoCs)*.

In this context, the performance of gigascale SoCs will be limited by the ability to efficiently interconnect pre-designed and pre-verified functional blocks and to accommodate their communication requirements, i.e. it will be communication—rather than computation—dominated. Only an interconnect-centric system architecture will be able to cope with these new challenges.

Current on-chip interconnects consist of low-cost shared communication re-

sources, where an arbitration logic is needed for the serialization of bus access requests: only one master at a time can drive the bus. In spite of its low complexity, the main drawback of this solution is its lack of scalability, which will result in unacceptable performance degradation (e.g., contention-related delays for bus accesses) when the level of SoC integration will exceed a dozen of cores. Moreover, the connection of new blocks to a shared bus increases its associated load capacitance, resulting in more energy consuming bus transactions.

State-of-the-art communication architectures make use of evolutionary approaches, such as full or partial crossbars, allowing a higher degree of parallelism in accessing communication resources, but in the long term more aggressive solutions are required.

A scalable communication infrastructure that better supports the trend of SoC integration consists of an on-chip packet-switched micro-network of interconnects, generally known as *Network-on-Chip (NoC)* architecture [3, 4, 5]. The basic idea is borrowed from traditional large-scale multi-processors and the wide-area networks domain, and envisions on-chip router (or switch)-based networks on which packetized communication takes place, as depicted in Fig. 1. Cores access the network by means of proper interfaces, and have their packets forwarded to destination through a multi-hop routing path.

The scalable and modular nature of NoCs and their support for efficient on-chip communication potentially leads to NoC-based multi-processor systems characterized by high structural complexity and functional diversity.

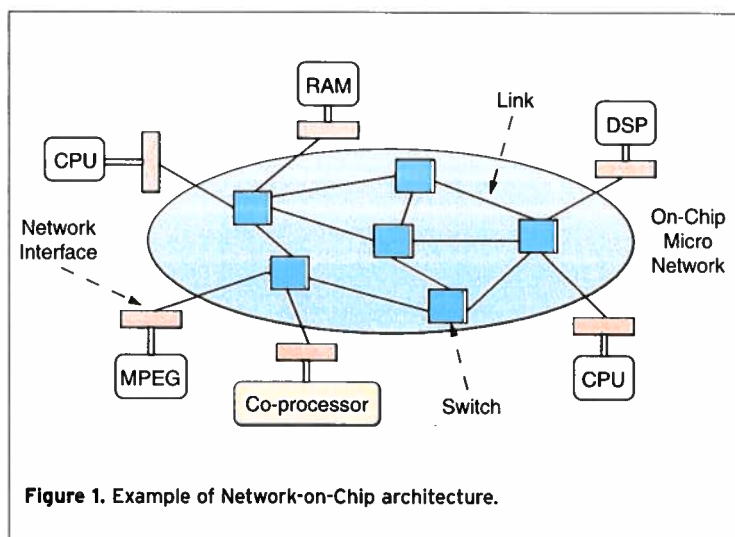


Figure 1. Example of Network-on-Chip architecture.

Davide Bertozzi and Luca Benini are with the Dipartimento Elettronica Informatica Sistemistica, University of Bologna, Via Risorgimento, 2, 40136 Bologna, Italy. E-mail: {dbertozzi, lbenini}@deis.unibo.it.

An important design decision for NoCs concerns topology selection. Several researchers [5, 6, 7, 8] envision NoCs as regular tile-based topologies (such as mesh networks and fat trees), which are suitable for interconnecting homogeneous cores in a chip multi-processor. Of course, the network topology depends on system functional requirements, hence on the size and the placement of the integrated modules. In general, SoC integration potentially encompasses heterogeneous technologies and specialized components (used by designers to optimise performance at low power consumption and competitive cost), leading to the integration of heterogeneous cores having varied functionality, size and communication requirements [9]. This might motivate the adoption of custom, domain-specific irregular topologies [10].

As an example, consider the implementation of an MPEG4 decoder [11], depicted in Fig. 2(b), where blocks are drawn roughly to scale and links represent inter-block communication. First, the embedded memory (SDRAM) is much larger than all other cores and it is a critical communication bottleneck. Block sizes are highly non-uniform and the floorplan does not match the regular, tile-based floorplan showed in Fig. 2(a). Second, the total communication bandwidth to/from the embedded SDRAM is much larger than that required for communication among the other cores. Third, many neighbouring blocks do not need to communicate. Even though it may be possible to implement MPEG4 onto a homogeneous fabric, there is a significant risk of either under-utilizing many tiles and links, or, at the opposite extreme, of achieving poor performance because of localized congestion. These factors motivate the use of an application-specific on-chip network [12].

This paper illustrates an embodiment of the concept of NoC represented by the *Xpipes* NoC architecture devel-

oped at University of Bologna [13]. *Xpipes* aims at providing high performance and reliable communication for gigascale SoCs. While describing the basic building blocks of this solution, the main NoC design principles will be briefly explained, detailing the specific contribution of *Xpipes* to the fast-evolving research activity in the domain of NoC-based communication architectures.

*Xpipes* has been designed as a library of highly parameterised soft macros (network interface, switch and switch-to-switch link), which are design-time tunable and composable. Therefore, arbitrary topologies can be instantiated and the challenging issue of heterogeneous NoC design can be addressed. In fact, an illustrative example will show how *Xpipes* can be employed in a design space exploration framework, wherein the ability of several customized NoC topologies to efficiently accommodate the communication requirements of an application is investigated and compared.

### Design Challenges for On-Chip Communication Architectures

Designing communication architectures for highly integrated deep sub-micron SoCs is a non-trivial task that needs to take into account the main challenges posed by technology scaling and by exponentially increasing system complexity. A few relevant SoC design issues are hereafter discussed:

- **Technology issues.** While gate delays scale down with technology, global wire delays typically increase or remain constant as repeaters are inserted. It is estimated that in 50 nm technology, at a clock frequency of 10 GHz, a global wire delay can be up to 6–10 clock cycles [3]. Therefore, limiting the on-chip distance travelled by critical signals will be key to guarantee the performance of the overall system, and will be a common design

guideline for all kinds of system interconnects. Synchronization of cores on future SoCs will be unfeasible due to deep sub-micron effects (clock skew, power associated with clock distribution trees, etc.), and an alternative scenario consists of self-synchronous cores that communicate with one another through a network-centric architecture [14]. Finally, signal integrity issues (cross-talk, power supply noise, soft errors, etc.) will lead to more transient and permanent failures of signals, logic values, devices and

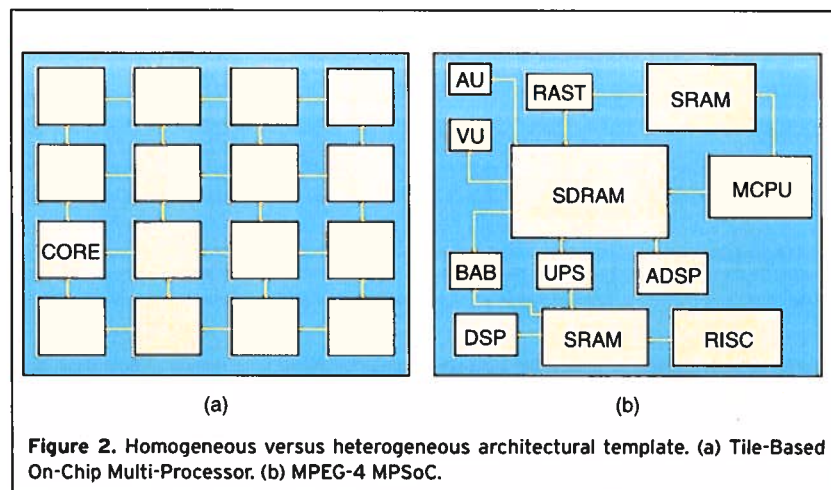


**Figure 2.** Homogeneous versus heterogeneous architectural template. (a) Tile-Based On-Chip Multi-Processor. (b) MPEG-4 MPSoC.

interconnects, thus raising the reliability concern for on-chip communication [15]. In many cases, on-chip networks can be designed as regular structures, allowing electrical parameters of wires to be optimised and well controlled. This leads to lower communication failure probabilities, thus enabling the use of low swing signalling techniques [16], and to the capability of exploiting performance optimisation techniques such as wavefront pipelining [17].

- **Performance issues.** In traditional busses, all communication actors share the same bandwidth. As a consequence, performance does not scale with the level of system integration, but degrades significantly. Though, once the bus is granted to a master, access occurs with no additional delay. On the contrary, NoCs can provide much better performance scalability. No delays are experienced for accessing the communication infrastructure, since multiple outstanding transactions originated by multiple cores can be handled at the same time, resulting in a more efficient network resource utilization. However, given a certain network dimension (e.g., number of instantiated switches), large latency fluctuations for packet delivery could be experienced as a consequence of network congestion. This is unacceptable when hard real time constraints of an application have to be met, and two solutions are viable: network overdimensioning (for NoCs designed to support best-effort traffic only) or implementation of dedicated mechanisms to provide guarantees for timing constrained traffic (e.g., loss-less data transport, minimal bandwidth, bounded latency, minimal throughput, etc.) [18].

- **Design productivity issues.** It is well known that synthesis and compiler technology development does not keep up with IC manufacturing technology development [19]. Moreover, time-to-market needs to be kept as low as possible. The reuse of complex pre-verified design blocks is efficient means to increase productivity, and regards both computation resources and the communication infrastructure [20]. It would be highly desirable to have processing elements that could be employed in different platforms by means of a plug-and-play design style. To this purpose, a scalable and modular on-chip network represents a more efficient communication infrastructure compared with shared bus-based architectures. However, the reuse of processing elements is facilitated by the definition of standard network interfaces, which also make the modularity property of the NoC

effective. The *Virtual Socket Interface Alliance* (VSIA) has attempted to set the characteristics of this interface industry-wide [21]. OCP [21] is another example of standard interface sockets for cores. It is worth remarking that such network interfaces also decouple the development of new cores from the evolution of new communication architectures. Finally, let us observe that NoC components (e.g., switches or interfaces) can be instantiated multiple times in the same design (as opposed to the arbiter of traditional shared busses, which is instance-specific) and reused in a large number of products targeting a specific application domain.

The above-mentioned issues are extensively discussed in [3, 5]. Some commercial SoC interconnects, such as the *Sonics MicroNetwork* [23] and the *STBUS* interconnect from STMicroelectronics are examples of evolutionary architectures which provide designers with a degree of freedom to instantiate different bus-based topologies. More radical solutions have been explored. Dally and Lacy sketch the architecture of a VLSI multi-computer using 2009 technology [24], where communication is based on packet switching. This seminal work draws upon the past experiences in designing parallel computers and reconfigurable architectures (FPGAs and their evolutions) [12, 25–27]. Most of the proposed NoC platforms are packet switched and exhibit regular structure. Examples are mesh interconnections [7, 28] and fat-trees [29].

The need to map communication requirements of heterogeneous cores may lead to the adoption of irregular topologies. The *Aethereal* NoC design framework presented in [30] aims at providing a complete infrastructure for developing heterogeneous NoCs with end-to-end quality of service guarantees. The network supports *guaranteed throughput* (GT) for real time applications and *best effort* (BE) traffic for timing unconstrained applications. Support for heterogeneous architectures requires highly configurable network building blocks, customisable at instantiation time for a specific application domain. For instance, the *Proteo* NoC [31] consists of a small library of predefined, parameterised components that allow the implementation of a large range of different topologies, protocols and configurations.

The *Xpipes* interconnect [13] and its synthesizer *XpipesCompiler* [32] push this approach to the limit, by instantiating an application specific NoC from a library of composable components providing reliable and latency insensitive operation.

### NoC Architecture

In general, the most relevant tasks of a *network interface*

(NI) are: (i) hiding the details about network communication protocol to the cores, so that they can be developed independently of the communication infrastructure, (ii) communication protocol conversion (from end-to-end to network protocol), (iii) data packetization (packet assembly, delivery and disassembly).

The former objective can be achieved by means of standard interfaces, such as VCI [21] and OCP [22], whose main characteristics are a high degree of configurability to adapt to the core's functionality and the independence of the request and response phases, thus supporting multiple outstanding requests and pipelining of transfers.

Data packetization is a critical task for the network interface, and has an impact on the communication latency, in addition to the latency of the communication channel. Messages that have to be transmitted across the network are usually partitioned into fixed-length packets. Packets in turn are often broken into message flow control units called *flits*. In the presence of channel width constraints, multiple physical channel cycles can be used to transfer a single flit. A *phit* is the unit of information that can be transferred across a physical channel in a single step. Flits represent logical units of information, as opposed to phits that correspond to physical quantities. In many implementations, a flit is set to be equal to a phit.

The packet preparation process consists of building the packet header, payload and packet tail. The header contains the necessary routing and network control information (e.g., the source and destination address). When source routing is used, the destination address is ignored and replaced with a route field that specifies the route to destination. This overhead in terms of packet header is counterbalanced by a simpler routing logic at the network switches: they simply have to look at the routing fields in the packets and route them over the specified switch output port. The packet tail indicates the end of a packet and usually contains parity bits for error-detecting or error-correcting codes. Packetizing data deals effectively with communication errors. In fact, packet boundaries contain the effect of errors and allow error recovery on a packet-by-packet basis.

The task of the switch is to carry packets injected into the network to their final destination, following a statically defined or dynamically determined routing path. The switch transfers packets from one of its input ports to one or more of its output ports.

The switch design is usually characterized by a power-performance trade-off: power-hungry switch memory resources can be required by the need to support high performance on-chip communication. A specific design of a switch may include both input and output buffers or only one type of buffers. Input queuing uses fewer

buffers, but suffers from head-of-line blocking. Virtual output queuing has a higher performance, but at the cost of more buffers. Network flow control (or routing mode) specifically addresses the limited amount of buffering resources in switches. Three policies are feasible in this context [33].

In *store-and-forward routing*, an entire packet is received and entirely stored before being forwarded to the next switch. This is the most demanding approach in terms of memory requirements and switch latency. Also *virtual cut-through routing* requires buffer space for an entire packet, but allows lower latency communication, in that a packet is forwarded as soon as the next switch guarantees that the complete packet will be accepted. If this is not the case, the current router must be able to store the whole packet.

Finally, a *wormhole routing* scheme can be employed to reduce switch memory requirements and to permit low latency communication. The first flit of a packet contains routing information, and header flit decoding enables the switches to establish the path and subsequent flits simply follow this path in a pipelined fashion by means of switch output port reservation. A flit is passed to the next switch as soon as enough space is available to store it, even though there is not enough space to store the whole packet. If a certain flit faces a busy channel, subsequent flits have to wait at their current locations and are therefore spread over multiple switches, thus blocking the intermediate links. This scheme avoids buffering the full packet at one switch and keeps end-to-end latency low, although it is more sensitive to deadlock and may result in low link utilization.

Guaranteeing quality of service in switch operation is another important design issue, which needs to be addressed when time-constrained (hard or soft real time) traffic is to be supported. Throughput guarantees or latency bounds are examples of time-related guarantees.

Contention related delays are responsible for large fluctuations of performance metrics, and a fully predictable system can be obtained only by means of contention-free routing schemes. With *circuit switching*, a connection is set up over which all subsequent data are transported. Therefore, contention resolution takes place at set-up at the granularity of connections, and time-related guarantees during data transport can be given. In *time division circuit switching* (see [23] for an example), bandwidth is shared by time division multiplexing connections over circuits.

In packet switching, contention is unavoidable since packet arrival cannot be predicted. Therefore arbitration mechanisms and buffering resources must be implemented at each switch, thus delaying data in an unpredictable manner and making it difficult to provide

guarantees. Best effort NoC architectures can mainly rely on network overdimensioning to bound fluctuations of performance metrics.

Finally, the design of communication (switch-to-switch and network interface-to-switch) links represents a critical issue with respect to the system performance. As geometries shrink, gate delay improves much faster than the delay in long wires. Therefore, the long wires increasingly determine the maximum clock rate, and hence performance, of the entire design. The problem becomes particularly serious for domain-specific heterogeneous SoCs, where the wire structure is highly irregular and may include both short and extremely long switch-to-switch links. Techniques have to be devised to decouple link throughput from its length and to allow functionally correct operation of the switches in presence of mis-aligned inputs due to the unequal length of the input links.

Next, implementation details about *Xpipes* network building blocks are provided, illustrating the specific design choices and how the above-mentioned issues have been addressed.

#### ***Xpipes* Architecture Overview**

*Xpipes* targets Multi-GHz heterogeneous packet-switched NoCs, thanks to the design of high performance network building blocks and to their instantiation time flexibility. The *Xpipes* design process was therefore partitioned into two relevant stages: (i) development of highly parameterizable network components, written in synthesizable SystemC; (ii) definition of a NoC instance by specifying parameters of the composable soft macros and an arbitrary network topology. A *XpipesCompiler* tool has been developed for this purpose.

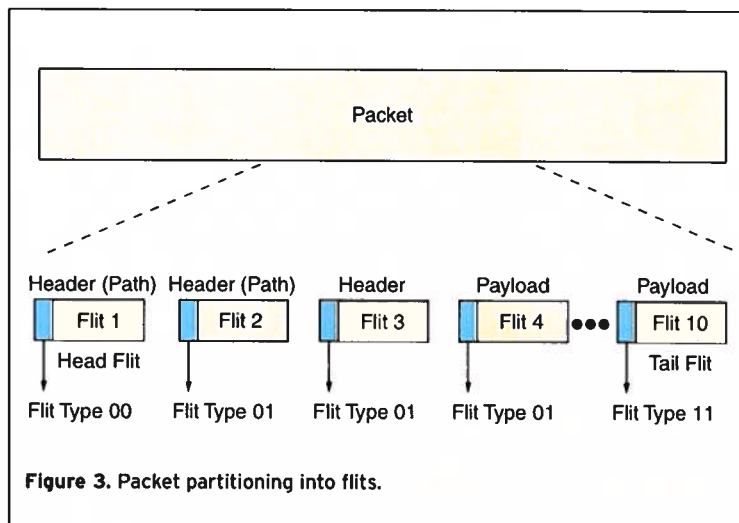
The high degree of parameterisation of *Xpipes* network building blocks regards both global network-specific parameters and block-specific parameters. The former ones include flit size, degree of redundancy of error control logic, address space of the cores, maximum number of hops between any two nodes, maximum number of bits for end-to-end flow control, etc. On the contrary, parameters specific to the network interface are: type of interface (master, slave or both), flit buffer size at the output port, content of routing tables for source-based routing, other interface parameters to the cores such as number of address/data lines, maximum burst length, etc. Parameterisation of the switches mainly regard the number of I/O ports, the number of virtual channels for each physical output link and the link buffer size. Finally, the length of each indi-

vidual link can be specified in terms of number of repeater stages, as will be discussed next.

*Xpipes* network interfaces use OCP as point-to-point communication protocol with the cores, and take care of protocol conversion to adapt to the network protocol. Data packetization results in the packet partitioning procedure illustrated in Fig. 3. A *flit type* field allows to identify the head and the tail flit, and to distinguish between header and payload flits.

The NoC backbone relies on a wormhole switching technique and makes use of a static routing algorithm called *street sign routing*. Routes are derived by the network interface by accessing a look-up table based on the destination address. Such information consists of direction bits read by each switch and indicating the output port of the switch that flits belonging to a certain packet have to be directed to. This routing algorithm allows a lightweight switch implementation as no dynamic decisions have to be taken at the switching nodes.

One of the main *Xpipes* design guidelines was the support for communication reliability. It was achieved by means of distributed error detection with link level retransmission as error recovery technique. Although a distributed approach to error detection causes a higher area overhead at the network nodes compared with an end-to-end solution, it is better able to contain the effects of error propagation, for instance preventing packets with corrupted header from being directed to the wrong path. In order to counterbalance this overhead, error detection with retransmission of incorrectly received data was preferred to error correction, since this latter requires complex energy-inefficient decoders. If average bit error rate is not high, performance penalty caused by retransmissions as perceived from the application can be neglected and



error detecting schemes have been showed to minimize average energy-per-bit [15]: as a consequence, they were the reference choice to provide communication reliability in *Xpipes*.

The retransmission policy implemented in *Xpipes* is *GO-BACK-N*. Flits are transmitted continuously and the transmitter does not wait for an ACK after sending a flit. Such an ACK is received after a round-trip delay. When a NACK is received, the transmitter backs up to the flit that is negatively acknowledged and re-sends it in addition to the  $N$  succeeding flits that were transmitted during the round-trip delay. At the receiver, the  $N-1$  received flits following the corrupted one are discarded regardless of whether they were received error-free or not. *GO-BACK-N* trades-off inefficiency in bus usage (retransmission of many error-free flits) with a moderate implementation cost, and was preferred to a selective-repeat scheme, wherein only those flits that are negatively acknowledged are retransmitted but more buffering resources are required at the destination switch.

In *Xpipes* switches, different error detecting decoders can be instantiated at design time, as a function of their latency and redundancy (additional parity lines) overhead and of their error detection capability, that has to be compared with the estimated (technology-dependent) bit error rate and the required *mean time between failures (MTBF)*. The considered error detecting codes

were several versions of the *Hamming code* and of the *Cyclic Redundancy Check (CRC) Code*, each one characterized by a different minimum distance and hence error detection capability. The ultimate objective was to select, for each supply voltage and flit size, coding schemes providing a MTBF of at least 1 year, and to successively select among them a scheme that minimizes decoding latency and/or redundancy.

Finally, it is worth noting that the support for high performance communication was provided in *Xpipes* by means of proper design techniques for the basic network components (such as link pipelining, deeply pipelined switches, latency insensitive component design). Implementation details about the network building blocks are provided below.

### Network Link

A solution to overcome the interconnect-delay problem consists of pipelining interconnects [34, 35]. Wires can be partitioned into segments (or relay stations, which have a function similar to the one of latches in a pipelined data path) whose length satisfies pre-defined timing requirements (e.g., the desired clock speed of the design). In this way, link delay is changed into latency, but the data introduction rate is not bound by the link delay any more. Now the latency of a channel connecting two modules may end up being more than one clock

cycle. This requires the system to be composed of modules whose behavior does not depend on the latency of input communication channels [34].

*Xpipes* interconnect makes use of pipelined links and of latency-insensitive operation of its building blocks. Switch-to-switch links are subdivided into basic segments whose length can be tailored to the desired clock frequency that needs to be achieved in the design. In this way, the system operating frequency is not bound by the delay of long links. According to the link length, a certain number of clock cycles is needed by a flit to cross the interconnect. If network switches are designed in such a way that their functional correctness depends on the flit arriving order and not on their timing, the input links of the switches can be different and of

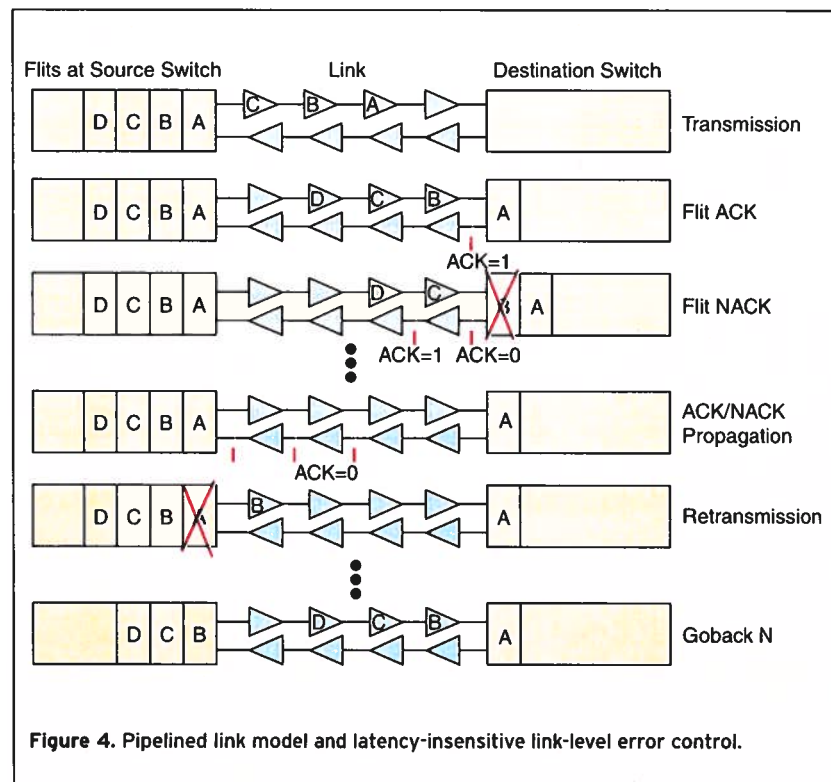


Figure 4. Pipelined link model and latency-insensitive link-level error control.

any length. These design choices are at the basis of latency insensitive operation of the NoC and allow the construction of an arbitrary network topology and hence support for heterogeneous architectures.

Fig. 4 illustrates the link model, which is equivalent to a pipelined shift register. Pipelining has been used both for data and control lines. The figure also illustrates how pipelined links affect the operation of latency-insensitive link-level error control: the retransmission of a corrupted flit between two successive switches is represented. Multiple outstanding flits propagate across the link during the same clock cycle. When flits are correctly received at the destination switch, an ACK is propagated back to the source, and after  $N$  clock cycles (where  $N$  is the link length expressed in terms of number of repeater stages) the flit will be discarded from the buffer of the source switch. On the contrary, a corrupted flit is NACKed and will be retransmitted in due time.

### Switch

A schematic representation of a switch for the *Xpipes* NoC is illustrated in Fig. 5. The example configuration

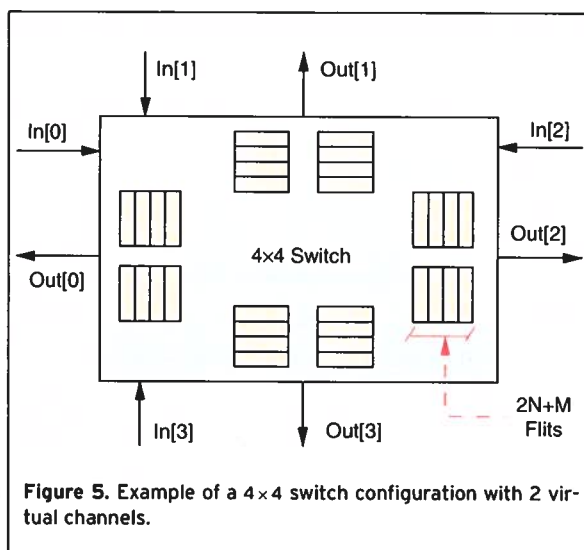


Figure 5. Example of a 4x4 switch configuration with 2 virtual channels.

exhibits 4 inputs, 4 outputs and 2 virtual channels multiplexed across the same physical output link. Switch operation is latency insensitive, in that correct operation is guaranteed for arbitrary link pipeline depth. For

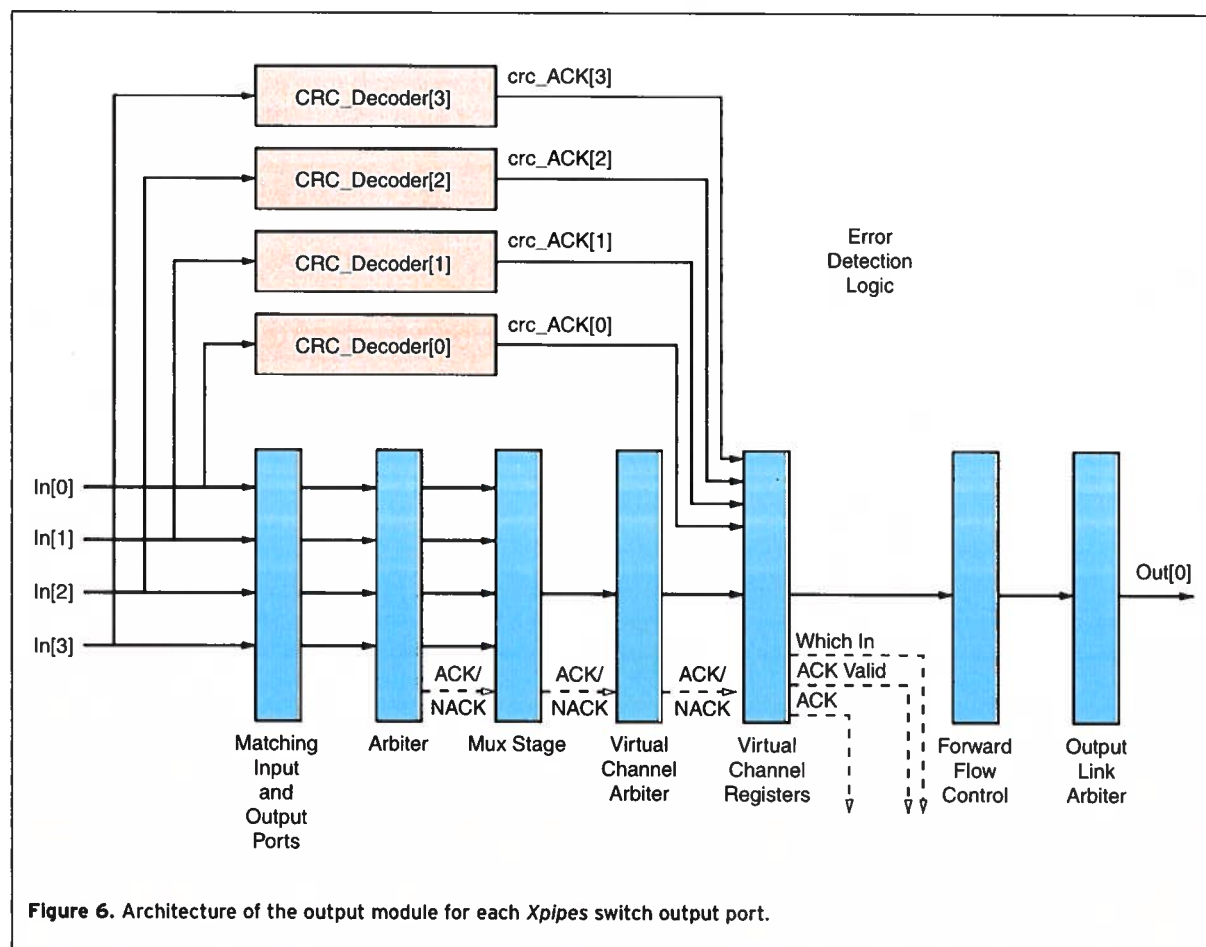


Figure 6. Architecture of the output module for each *Xpipes* switch output port.

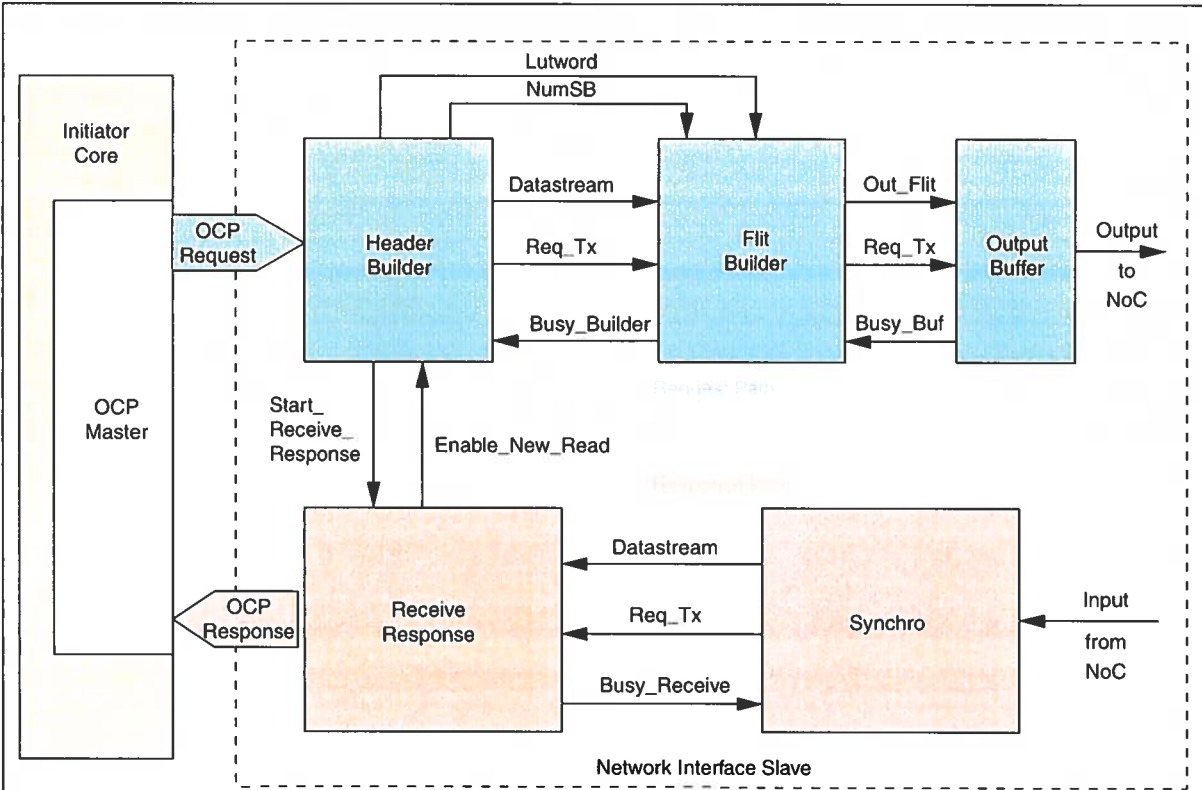


Figure 7. Architecture of Xpipes network interface slave.

latency insensitive operation, the switch has virtual channel registers to store  $2N + M$  flits, where  $N$  is the link length (expressed as number of basic repeater stages) and  $M$  is a switch architecture related contribution (12 cycles in this design). The reason is that each transmitted flit has to be acknowledged before being discarded from the buffer. Before an ACK is received, the flit has to propagate across the link ( $N$  cycles), an ACK/NACK decision has to be taken at the destination switch (a portion of  $M$  cycles), the ACK/NACK signal has to be propagated back ( $N$  cycles) and recognized by the source switch (remaining portion of  $M$  cycles). During this time, other  $2N + M$  flits are transmitted but not yet ACKed.

Output buffering was chosen for Xpipes switches, and the resulting architecture consists of multiple replications of the same output module reported in Fig. 6, one for each switch output port. All switch input ports are connected to the each module's inputs. Flow control signals generated by each module (such

as ACK and NACK for incoming flits) are collected by a centralized switch unit, which directs them back to the proper source switch.

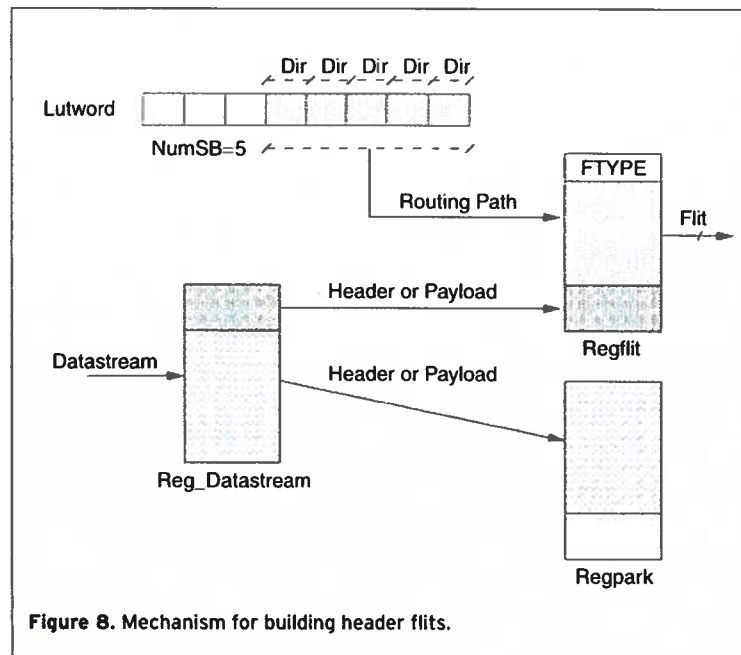


Figure 8. Mechanism for building header flits.

As can be observed from Fig. 6, each output module is deeply pipelined (7 pipeline stages) so to maximize the operating clock frequency of the switch. The CRC decoders for error detection work in parallel with switch operation, thereby hiding their latency.

The first pipeline stage checks the header of incoming packets on the different input ports to determine whether those packets have to be routed through the output port under consideration. Only matching packets are forwarded to the second stage, which resolves contention based on a round-robin policy. Arbitration is carried out when the tail flit of the preceding packet is received, so that all other flits of a packet can be propagated without contention related delay at this stage. A NACK for flits of non-selected packets is generated. The third stage is just a multiplexer, which selects the prioritised input port. The following arbitration stage keeps the status of virtual channel registers and determines whether flits can be stored into the registers or not. A header flit is sent to the register with the greatest number of free locations, and followed by successive flits of the same packet. The fifth stage is the actual buffering stage, and the ACK/NACK response at this stage indicates whether a flit has been successfully stored or not. The following stage takes care of forward flow control: a flit is transmitted to the next switch only when adequate free register locations are available at the output port of interest of the destination switch. Finally, a last arbitration stage multiplexes the virtual channels on the physical output link on a flit-by-flit basis, thereby improving network throughput.

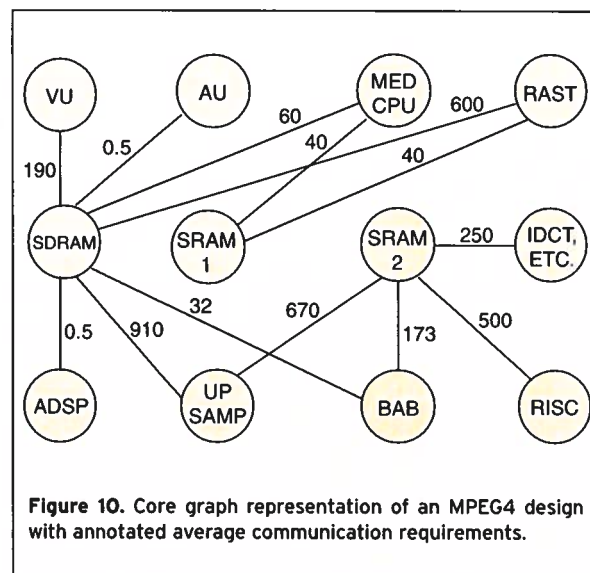
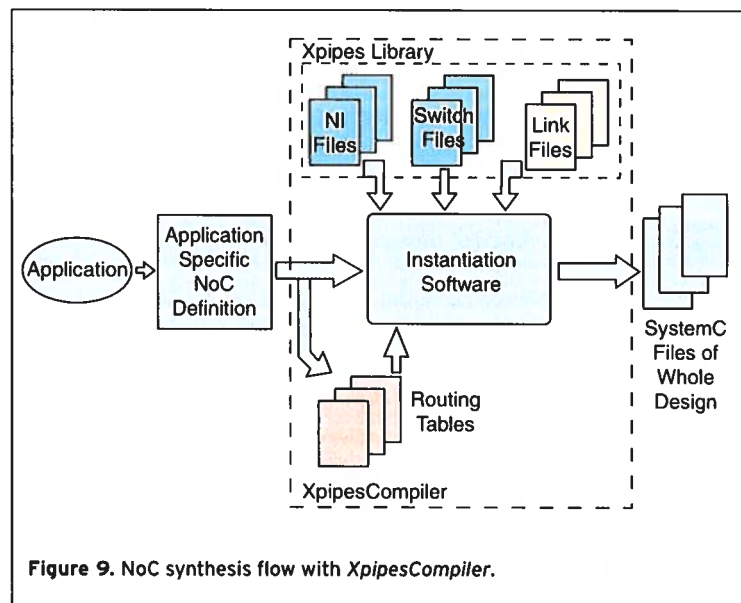
### Network Interface

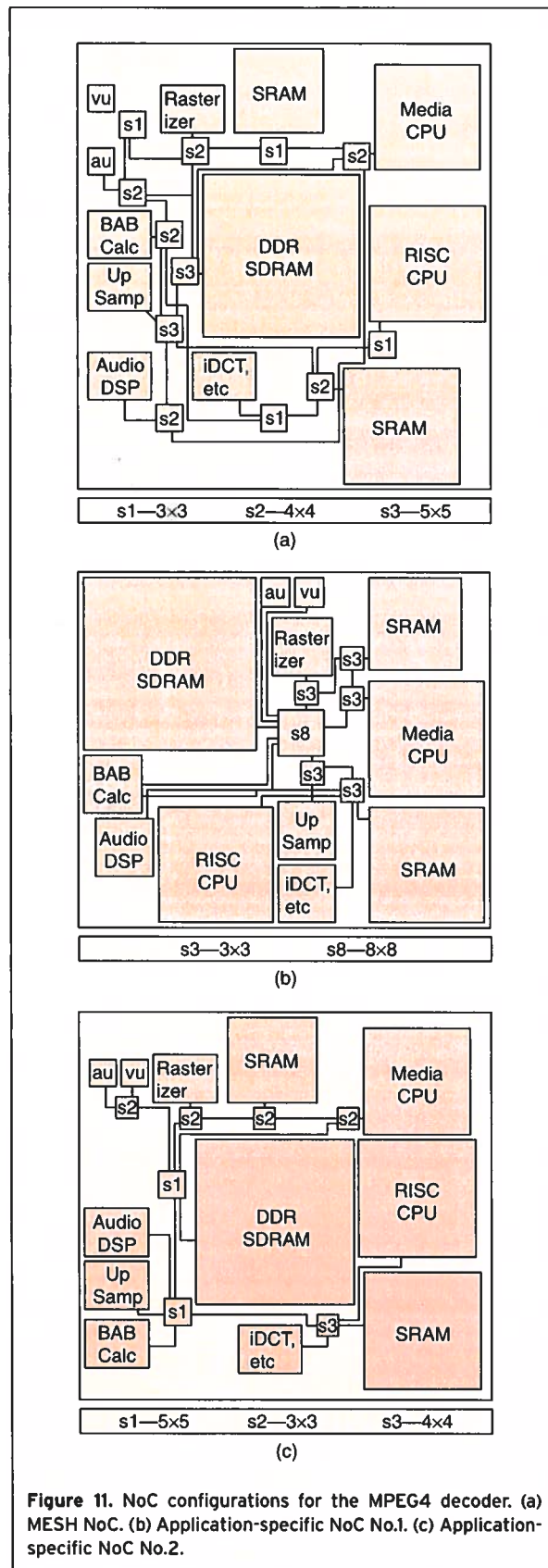
The *Xpipes* network interface provides a standardized OCP interface to networked cores. The NI for cores that initiate communication (*initiators*) needs to turn OCP-compliant transactions into packets to be transmitted across the network. It represents the slave side of an OCP end-to-end connection (the master side being the initiator core), and it is therefore referred to as *network interface slave (NIS)*. Its architecture is showed in Fig. 7.

The NIS has to build the packet header, which has to be spread over a variable number of flits depending on the length of the path to the destination node. The look-up table containing static routing information is accessed by the *HEADER\_BUILDER* block and the desti-

nation address is used as table entry. In this way, two routing fields are extracted: *numSB* (the number of hops to destination) and *lutword* (the actual direction bits read from the look-up table). Together with the core transaction related information (*datastream*), they are forwarded to the *FLIT\_BUILDER* block, provided the enable signal *busy\_builder* is not asserted.

Based on the input data, the module *FLIT\_BUILDER* has the task of building the flits to be transmitted via the output buffer *OUTPUT\_BUFFER*, according to the mechanism illustrated in Fig. 8. Let us assume that a packet requires *numSB* = 5 hops to get to the destination, and that the direction to be taken at each switch is expressed by *DIR*. The module *FLIT\_BUILDER* builds the





first flit by concatenating the flit type field with path information. If there is some space left in the flit, it is filled with header information derived from the input *datastream*. The unused part of the *datastream* is stored in a *regpark* register, so that a new *datastream* can be read from the *HEADER\_BUILDER* block. The following header and/or payload flits will be formed by combining data stored in *regpark* and *reg\_datastream*. No partially filled flits are transmitted to make transmission more efficient. Finally, the module *OUTPUT\_BUFFER* stores flits to be sent across the network, and allows the NIS to keep preparing successive flits also when the network is congested. Size of this buffer is a design parameter.

The response phase is carried out by means of two modules. *SYNCHRO* receives incoming flits and reads out only useful information (e.g., it discards routing fields). At the same time, it contains buffering resources to synchronize the network's requests to transmit the remaining flits of a packet with the core consuming rate. The *RECEIVE\_RESPONSE* module translates useful header and payload information into OCP-compliant response fields.

When a read transaction is initiated by the initiator core, the *HEADER\_BUILDER* block asserts a *start\_receive\_response* signal that triggers the waiting phase of the *RECEIVE\_RESPONSE* module for the requested data. As a consequence, the NIS supports only one outstanding read operation to keep interface complexity low. Although no read after read transactions can be initiated unless the previous one has been completed, an indefinite number of write transactions can be carried out after an outstanding read or write has been initiated.

The architecture of a *network interface master* is similar to the one just described, and is not reported here for lack of space.

#### Domain-Specific Network-on-Chip Synthesis Flow

The individual components of SoCs are inherently heterogeneous with widely varying functionality and communication requirements. The communication infrastructure should optimally match communication patterns among these components. Looking for the most efficient solution can be a (possibly automated) step of a more general design methodology of custom domain-specific NoCs, and *Xpipes* could be used within this framework as a library of components for the synthesis of the selected NoC configuration.

Therefore, for a *Xpipes*-based design methodology, a tool is needed to automatically instantiate network building blocks (switches, network interfaces and links) from the library of composable soft macros described in SystemC. *XpipesCompiler* [32] is the tool

designed to automatically instantiate a *Xpipes*-based application-specific NoC for heterogeneous on-chip multi-processors.

The complete *XpipesCompiler*-assisted NoC design flow is depicted in Fig. 9. From the specification of an application, the designer (or a high-level analysis and exploration tool) creates a high-level view of the SoC floorplan, including nodes (with their network interfaces), links and switches. Based on the clock speed target and link routing, the number of pipeline stages for each link is also determined. The information on the network architecture is then specified in an input file for the *XpipesCompiler*. Routing tables for the network interfaces are also specified. The tool takes the *Xpipes* library of soft network components as an additional input. The output is a SystemC hierarchical description, which includes all switches, links, network nodes and interfaces and specifies their connectivity. Then the final description can be compiled and simulated at the cycle-accurate and signal-accurate level. At this point, the description can be fed to back-end RTL synthesis tools for silicon implementation.

In a nutshell, the *XpipesCompiler* generates a set of network component instances that are custom-tailored to the specification contained in its input network description file. This tool allows a very instructive comparison of the effects (in terms of area, power and performance) of mapping applications on customized domain-specific NoCs and regular mesh NoCs.

As an example, let us focus on the MPEG4 decoder already introduced in this paper. Its core graph representation together with its average communication requirements are presented in Fig. 10. The edges are annotated with the average bandwidth requirements of the cores in MB/s. Customized application-specific NoCs that closely match the application's communication characteristics have been manually developed and compared with a regular mesh topology. The different NoC configurations are presented in Fig. 11. In the MPEG4 design considered, many of the cores communicate with each other through the shared SDRAM. So a large switch is used to connect the SDRAM (Fig. 11(b)), while smaller switches are instantiated for the other cores. An alternative custom NoC is also considered (Fig. 11(c)): it is an optimised mesh network, with superfluous switches and switch I/Os removed. We refer to this solution as a "distributed" custom NoC, as opposed to the "centralized" one.

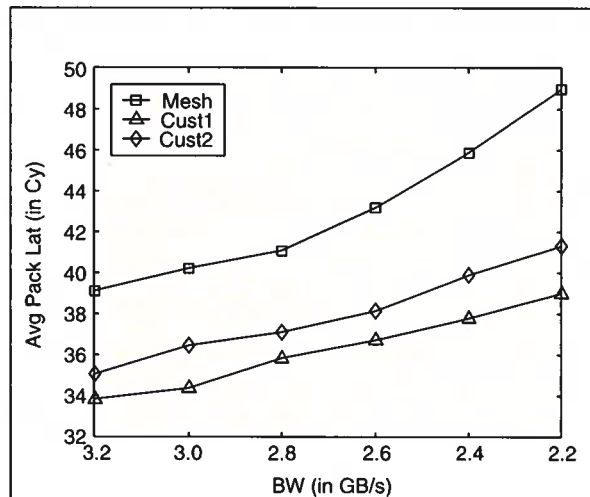


Figure 12. Average packet latency as a function of the link bandwidth.

Area (in 0.1  $\mu$ m technology) and power estimates for different NoC configurations are reported in Table 1. As it can be observed, area and power savings are achieved with the custom configurations. However, the distributed solution turns out to be more effective, mainly because the centralized approach makes use of a very large switch. In fact, since power dissipation on a switch increases non-linearly with increase in switch size, there is more power dissipation on the switches of the custom NoC1 (that has an  $8 \times 8$  switch) than in the mesh NoC. However, most of the traffic traverses short links in this custom NoC, thereby providing marginal power savings for the whole design. Power and area savings achieved by the custom NoC2 are more significant thanks to the use of smaller switches.

Fig. 12 reports the variation of average packet latency (for 64 B packets, 32 bit flits) with link bandwidth. Custom NoCs, as synthesized by *XpipesCompiler*, have lower packet latencies, as the average number of switches and link traversals is lower. At the minimum plotted bandwidth value, almost 10% latency saving is achieved. Moreover, the latency increases more rapid-

Table 1. Area and power estimates for the MPEG4-related NoC configurations.

| NoC Configuration | Area (mm <sup>2</sup> ) | Ratio Mesh/Cust | Power (Mw) | Ratio Mesh/Cust |
|-------------------|-------------------------|-----------------|------------|-----------------|
| Mesh              | 1.31                    |                 | 114.36     |                 |
| Custom 1          | 0.86                    | 1.52            | 110.66     | 1.03            |
| Custom 2          | 0.71                    | 1.85            | 93.66      | 1.22            |

ly with the mesh NoC as the link bandwidth decreases. Also, custom NoCs have better link utilization: around 1.5 times the link utilization of a mesh topology.

Overall, it should be observed that area, power and performance optimisations by means of custom NoCs turn out to be more difficult for MPEG4 than for other applications such as Video Object Plane Decoders (VOPD) and Multi-Window Displayers [32]. In fact, the MPEG4 core graph shows that almost all cores communicate with many other cores (thus requiring many switches) and most of the traffic traverses the larger switches connected to the memories (thus generating high power dissipation). On the contrary, a custom NoC can be generated for VOPD by observing that a half of the cores communicate with more than a single core. This motivates a configuration of the custom NoC, having less than half the number of switches than the mesh NoC. In this way, the custom NoC consumes about 5.7 times less area and 2.7 times less power than the corresponding mesh NoC.

These examples show that customized domain-specific NoCs exhibit large potentials for power minimization of gigascale on-chip multi-processors, and push the development of design exploration frameworks helping designers to assess the effectiveness of alternative custom NoC architectures. *Xpipes* and its synthesizer have been designed as key elements of such a NoC exploration and synthesis flow.

### Conclusions

Current application and technology trends motivate a paradigm shift in on-chip interconnect architectures from bus-based solutions to packet-switched Networks-on-Chip. This paper analyses the key drivers for the development of packet-switched on-chip micro-networks and introduces a NoC architecture targeting high performance and reliable communication for domain-specific SoCs. *Xpipes* is a library of soft macros (switches, network interfaces and links) that are design-time composable and tunable, well suited for irregular topologies. An example NoC synthesis flow has also been illustrated, wherein a compiler tool (*XpipesCompiler*) is used to automatically instantiate customized network building blocks from the *Xpipes* library.

*Xpipes* and related NoC design efforts are mostly focused on revisiting the architectural foundations of the on-chip communication. This is, however, only a facet of a much more complex challenge, namely, the development of a comprehensive architectural, software and operating system infrastructure required to efficiently support the next generation large-scale multi-processor Systems-on-Chip.

### Bibliography

- [1] A. Allan, D. Edenfeld, W.H. Joyner, Jr, A.B. Kahng, M. Rodgers, and Y. Zorian, "2001 Technology Roadmap for Semiconductors," *IEEE Computer*, pp. 42–53, Jan. 2002.
- [2] F. Boekhorst, "Ambient intelligence, the next paradigm for consumer electronics: How will it affect silicon?," *ISSCC 2002*, vol. 1, pp. 28–31, Feb. 2002.
- [3] L. Benini and G. De Micheli, "Networks on chips: A new SoC paradigm," *IEEE Computer*, vol. 35, no. 1, pp. 70–78, Jan. 2002.
- [4] P. Wielage and K. Goossens, "Networks on silicon: Blessing or nightmare?," in *Proc. Of the Euromicro Symp. on Digital System Design DSD02*, Sept. 2002, pp.196–200.
- [5] W.J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," *Design and Automation Conf. DAC01*, Jun. 2001, pp. 684–689.
- [6] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet switched interconnections," *Design, Automation and Testing in Europe DATE00*, Mar. 2000, pp. 250–256.
- [7] S. Kumar, A. Jantsch, J.P. Soininen, M. Forsell, M. Millberg, J. Oeberg, K. Tiensyrja, and A. Hemani, "A network on chip architecture and design methodology," *IEEE Symp. on VLSI ISVLSI02*, Apr. 2002, pp. 105–112.
- [8] S.J. Lee et al., "An 800 MHz Star-Connected On-Chip Network for Application to Systems on a Chip," *ISSCC03*, Feb. 2003.
- [9] S. Ishiwata et al., "A single chip MPEG-2 codec based on customizable media embedded processor," *IEEE JSSC*, vol. 38, no. 3, pp. 530–540, Mar. 2003.
- [10] H. Yamauchi et al., "A 0.8 W HDTV video processor with simultaneous decoding of two MPEG2 MP@HL streams and capable of 30 frames/s reverse playback," *ISSCC02*, vol. 1, Feb. 2002, pp. 473–474.
- [11] E. B. Van der Tol and E.G.T. Jaspers, "Mapping of MPEG4 decoding on a flexible architecture platform," *SPIE 2002*, Jan. 2002, pp. 1–13.
- [12] H. Zhang et al., "A 1 V heterogeneous reconfigurable DSP IC for wireless baseband digital signal processing," *IEEE Journal of SSC*, vol. 35, no.11, pp. 1697–1704, Nov. 2000.
- [13] M. Dall'Osso, G. Biccari, L. Giovannini, D. Bertozzi, and L. Benini, "Xpipes: A latency insensitive parameterized network-on-chip architecture for multi-processor SoCs," *ICCD03*, Oct. 2003, pp. 536–539.
- [14] ITRS 2001. Available: <http://public.itrs.net/Files/2001ITRS/Home.htm>
- [15] D. Bertozzi, L. Benini, G. De Micheli, "Energy-reliability trade-off for NoCs," in *Networks on Chip*, A. Jantsch and Hannu Tenhunen, eds., Dordrecht: Kluwer, 2003, pp. 107–129.
- [16] H. Zhang, V. George, and J.M. Rabaey, "Low-swing on-chip signaling techniques: Effectiveness and robustness," *IEEE Trans. on VLSI Systems*, vol. 8, no. 3, pp. 264–272, Jun. 2000.
- [17] J. Xu, and W. Wolf, "Wave pipelining for application-specific networks-on-chips," *CASES02*, Oct. 2002, pp. 198–201.
- [18] K. Goossens, J. Dielissen, J. van Meerbergen, P. Poplavko, A. Radulescu, E. Rijpkema, E. Waterlander, and P. Wielage, "Guaranteeing the quality of services in networks on chip," in *Networks on Chip*, A. Jantsch and Hannu Tenhunen, eds. Dordrecht: Kluwer, 2003, pp. 61–82.
- [19] ITRS 1999. Available: [http://public.itrs.net/files/1999\\_SIA\\_Roadmap/](http://public.itrs.net/files/1999_SIA_Roadmap/)
- [20] A. Jantsch and H. Tenhunen, "Will networks on chip close the productivity gap?," in *Networks on Chip*, A. Jantsch and Hannu Tenhunen, eds. Dordrecht: Kluwer, 2003, pp. 3–18.
- [21] VSI Alliance. Virtual Component Interface Standard, 2000.
- [22] OCP International Partnership. Open Core Protocol Specification, 2001.
- [23] D. Wingard, "MicroNetwork-based integration for SoCs," *Design Automation Conf. DAC01*, Jun. 2001, pp. 673–677.
- [24] W.J. Dally and S. Lacy, "VLSI architecture: Past, present and future,"

*Conf. Adv. Research in VLSI*, 1999, pp. 232–241.

[25] D. Culler, J.P. Singh, and A. Gupta, *Parallel Computer Architecture, a Hardware/Software Approach*, San Moteo, MA: Morgan Kaufmann, 1999.

[26] K. Compton and S. Hauck, "Reconfigurable computing: A survey of system and software," *ACM Computing Surveys*, vol. 34, no. 2, pp. 171–210, Jun. 2002.

[27] R. Tessier and W. Burleson, "Reconfigurable computing and digital signal processing: A survey," *Journal of VLSI Signal Processing*, vol. 28, no. 3, pp. 7–27, May–Jun. 2001.

[28] Dale Liu et al., "SoCBUS: The solution of high communication bandwidth on chip and short TTM," invited paper in *Real Time and Embedded Computing Conf.*, Sept. 2002.

[29] J. Walrand and P. Varaja, *High Performance Communication Networks*, San Francisco: Morgan Kaufmann, 2000.

[30] E. Rijpkema, K. Goossens, A. Radulescu, J. van Meerbergen, P. Wielage, and E. Waterlander, "Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip," *Design Automation and Test in Europe DATE03*, Mar. 2003, pp. 350–355.

[31] I. Saastamoinen, D. Siguenza-Tortosa, and J. Nurmi, "Interconnect IP node for future systems-on-chip designs," *IEEE Work on Electronic Design, Test and Applications*, Jan. 2002, pp. 116–120.

[32] "XpipesCompiler: A Tool for Instantiating Application Specific Networks-on-Chip," submitted to DATE 2004.

[33] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection networks: An Engineering Approach*, IEEE Computer Society Press, 1997.

[34] L.P. Carloni, K.L. McMillan, and A.L. Sangiovanni-Vincentelli, "Theory of latency-insensitive design," *IEEE Trans. On CAD of ICs and Systems*, vol. 20, no. 9, pp. 1059–1076, Sept. 2001.

[35] L. Scheffer, "Methodologies and tools for pipelined on-chip interconnects," *Int. Conf. On Computer Design*, 2002, pp. 152–157.



**Luca Benini** received the B.S. degree (summa cum laude) in electrical engineering from the University of Bologna, Italy, in 1991, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University in 1994 and 1997, respectively. He is an associate professor

in the Department of Electronics and Computer Science in the University of Bologna. He also holds visiting researcher positions at Stanford University and the Hewlett-Packard Laboratories, Palo Alto, CA.

Dr. Benini's research interests are in all aspects of computer-aided design of digital circuits, with special emphasis on low-power applications, and in the design of portable systems. On these topics, he has published more than 200 papers in international conference proceedings and journals. He is a co-author of two books: *Dynamic Power Management, Design Techniques and CAD tools*, Kluwer 1998, and *Memory Design Techniques for Low Energy Embedded Systems*, Kluwer 2002.

Dr. Benini is a member of the technical program committee for several technical conferences, including the Design Automation Conference, the International Sympo-

sium on Low Power Design and the International Symposium on Hardware-Software Codesign; he is a vice programme chair for the 2004 Design, Automation and Test in Europe.



**Davide Bertozzi** received the PhD degree in Electrical Engineering and Computer Science from University of Bologna (Italy) in 2003.

He currently holds a Post-Doc position at the same University. He has been a visiting researcher at Stanford University, NEC Research America and Philips Research Labs. His research interests include Multi-Processor System-on-Chip Architectures, with particular emphasis on the communication infrastructure and on its high-performance and energy-efficient implementation.

## CALL FOR PAPERS

# PATMOS 2004

Fourteenth International Workshop on  
Power and Timing Modeling, Optimization and Simulation

September 15–17, 2004 Isle of Santorini, Greece

<http://www.vlsi.ee.upatras.gr/patmos04>

The PATMOS objective is to provide a forum to discuss and investigate the emerging problems in methodologies and tools for the design of next generations of integrated circuits and systems. The technical program will focus on speed and low-power aspects, with particular emphasis on modeling, characterization, design and architectures.

Contributions are invited for regular presentations and discussion sessions. Electronic submission is required and should follow the style for the final publication. Check the workshop web-page (<http://www.vlsi.ee.upatras.gr/patmos04>) for complete author and submission instructions. Submitted papers will be reviewed formally and anonymously by several reviewers.

Proposals for panel sessions and special sessions are encouraged and must be received not later than *March 14, 2004*.

### Author's Schedule

- \* Deadline for paper submission: **March 14, 2004**
- \* Notification of acceptance: **May 16, 2004**
- \* Deadline for final paper submission: **June 13, 2004**

#### General Co-Chairs

**Vassilis Paliouras**  
[paliouras@ee.upatras.gr](mailto:paliouras@ee.upatras.gr)

**Odysseas Koufopavlou**  
[odysseas@ee.upatras.gr](mailto:odysseas@ee.upatras.gr)  
University of Patras  
Electrical and Computer  
Engineering Dept.,  
26500 Patras, Greece  
Phone: +30 2610 997 319  
Fax: +30 2610 994 798

#### Program Chair

**Prof. Enrico Macii**  
[enrico.macii@polito.it](mailto:enrico.macii@polito.it)  
Politecnico di Torino  
Dip. di Automatica e Informatica  
Corso Duca degli Abruzzi 24  
10129 Torino, Italy  
Phone: +39-011-564 7074  
Fax: +39-011-564 7099

# Exhibit O

# User-Level Network Interface Protocols



**User-level communication architectures remove the operating system from the critical communication path, but designing protocols for these architectures is not trivial. The authors survey six design issues using examples from 11 communication systems, all of which have been implemented on a Myrinet network.**

Raoul A.F. Bhoedjang  
Tim Rühl  
Henri E. Bal  
Vrije  
Universiteit  
Amsterdam

**M**odern high-speed local area networks offer great potential for communication-intensive applications, but their performance is limited by the use of traditional communication protocols, such as TCP/IP. In most cases, these protocols require that all network access be through the operating system, which adds significant overhead to both the transmission path (typically a system call and data copy) and the receive path (typically an interrupt, a system call, and a data copy). To address this performance problem, several *user-level communication architectures* have been developed that remove the operating system from the critical communication path.<sup>1,2</sup> In this article, we describe six important issues to consider in designing communication protocols for user-level architectures.

The issues we have selected focus on the performance and semantics of a communication system. The first, the *data transfer* mechanism a protocol uses, significantly affects latency and throughput. Direct memory access usually gives the best throughput in user-level communication, but to avoid unnecessary copying, sophisticated *address translation* mechanisms are needed, another important design issue. To make user-level protocols feasible in a multiprogramming environment, designers also need *protection* mechanisms that do not rely on operating system intervention. Network packets are received using some *control transfer* mechanism—either polling or interrupts, each of which has its advantages and drawbacks. The last issues we focus on are *reliability* and *multicast*, which in modern networks can be supported both on the host processor and the network interface.

To provide a basis for analyzing these issues, we present a simple network interface protocol for Myricom's Myrinet network.<sup>3</sup> As the sidebar "The Myrinet Network" describes, Myrinet has a programmable network interface. Researchers can thus explore many protocol design options, and several groups have designed communication systems for Myrinet. In this article, we refer to 11 such systems. Table 1 lists the

characteristics of each. As the table shows, the systems differ significantly in how they resolve the design issues we describe, but all aim for high performance and provide a lean, low-level, and more or less generic communication facility.

## A SIMPLE NETWORK INTERFACE PROTOCOL

Figure 1 shows the operation of our simple network interface protocol for Myrinet. Operation begins when a user process of the sending host—host A—invokes a simple *send* primitive to send a data packet. Because the protocol sends packets with a maximum payload of 256 bytes, users must fragment their data so that each fragment fits in a packet.

*Send* performs two actions. It first copies the user data to a packet buffer in a special staging area, the *DMA area*, in the host's memory (step 1). The host then writes a *send* request into a descriptor in network interface memory (step 2) and sets a flag to inform the network interface of this event. These descriptors, which are stored in a circular buffer, or *send ring*, contain the destination machine, the size of the packet's payload, and the packet's offset in the DMA area.

The network interface repeatedly polls the flag of the first empty descriptor in the send ring. As soon as the host sets this flag, the network interface reads the packet's offset in the descriptor and adds it to the address of the start of the DMA area, which yields the physical address of the packet (step 3). The network interface initiates a DMA transfer over the I/O bus to copy the packet's payload from host memory to its own memory (step 4). It then reads the destination machine in the descriptor and looks up the route for the packet in a routing table. Finally, it starts a second DMA to transmit the packet (step 5).

The data transfers in steps 1, 4, and 5 can be performed concurrently. For example, if the host sends a long, multipacket message, one packet's network DMA can be overlapped with the next packet's host-to-interface DMA. When the network interface detects that a given packet's network DMA has completed, it sets a flag to release the

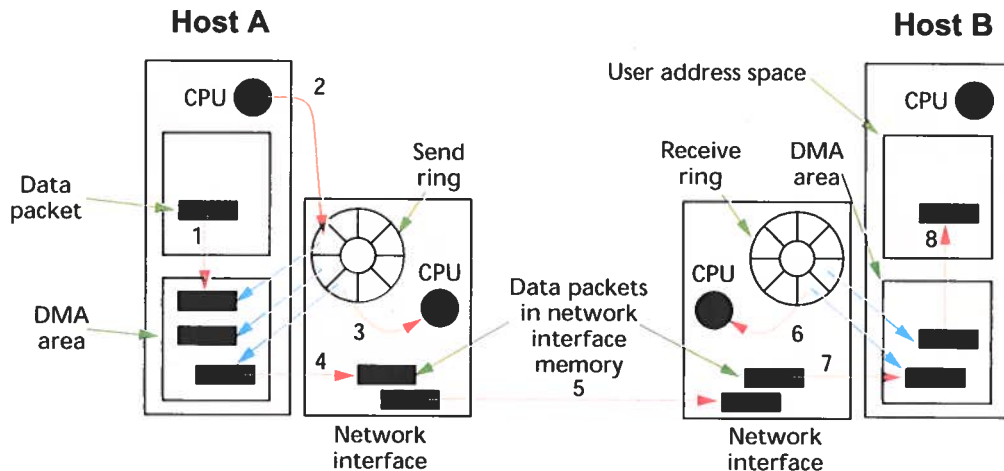


**Figure 1.** How the simple network interface protocol operates. Host A copies user data into the DMA area (step 1) and writes a packet descriptor to the send ring (step 2). The network interface reads the packet descriptor (step 3) and copies the packet through DMA to its memory (step 4). After the network transfer (step 5), the receiving network interface reads its receive ring to find an empty buffer in the DMA area of host B (step 6) and copies the packet using DMA to this area (step 7). Host B optionally copies data from the packet to a user buffer (step 8).

## The Myrinet Network

Myrinet, developed by Myricom, is a switched, gigabit-per-second technology that uses variable-length packets. The packets are wormhole-routed through a network of highly reliable links and crossbar switches. Techniques like wormhole routing are normally found only in supercomputers.

Figure A illustrates the architecture of a node in a Myrinet cluster. Each machine (host) has a network interface card that contains a processor and some memory, which is used to store the interface's control program and data. The network interface connects to the host's I/O bus—a typical organization for commodity hardware.

Our host system has a 200-MHz Pentium Pro processor, a host bus speed of 66 MHz × 64 bits, and an I/O bus speed (PCI) of 33 MHz × 32 bits. All hosts run the BSD/OS (Version 3.0) operating system.

Myrinet requires that all packets be staged through network interface memory, both at the sending and the receiving side. It uses fast but expensive SRAM, so the memory is relatively small. Both the host and the network interface can use DMA to access data in each other's memory, but as the main article describes, DMA transfers suffer from a start-up overhead. The host can also access the network interface's memory using programmed I/O, which has no start-up costs, but results in high access times relative to host memory.

Like other modern networks, Myrinet has a programmable network interface processor. Although this gives protocol designers much flexibility, the network interface processor is much slower than the host CPU, so it can do only simple tasks. Adding just a few instructions to the critical path of a Myrinet control program noticeably increases end-to-end latency.

Our network interfaces have a 33-MHz LANai4.1 processor

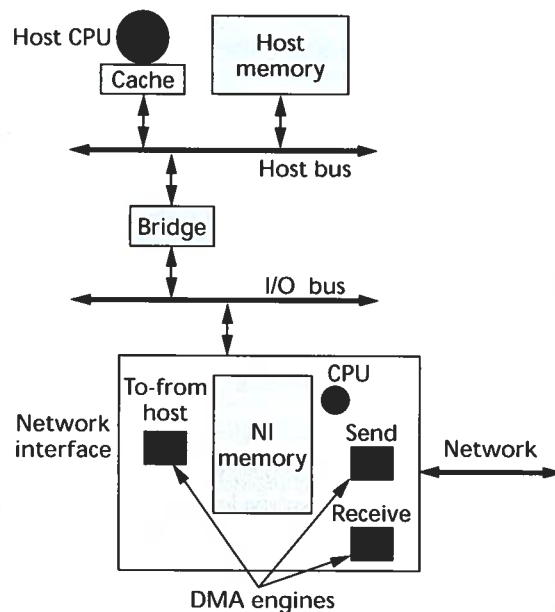


**Figure A.** The host and network interface architecture of Myricom's Myrinet.

(LANai is the processor's proprietary name), 1-Mbyte SRAM, a link speed of 2 × 1.28 Gbps, a to-from host DMA start-up cost of 5 LANai cycles, a send DMA start-up cost of 5 LANai cycles, and a receive DMA start-up cost of 4 LANai cycles.

**Table 1. Characteristics of 11 communication systems built for Myrinet.**

| System               | Data transfer (host-NI) | Translation                                  | Protection            | Control transfer                | Reliability                                                                                    | Multicast support          |
|----------------------|-------------------------|----------------------------------------------|-----------------------|---------------------------------|------------------------------------------------------------------------------------------------|----------------------------|
| AM-II <sup>1</sup>   | PIO & DMA*              | DMA areas                                    | Yes                   | Polling + interrupts            | Reliable, network interface: alternating bit, host: sliding window                             | No                         |
| FM <sup>2</sup>      | PIO                     | DMA area (recv)                              | No                    | Polling                         | Reliable, host-level credits                                                                   | No                         |
| FM/MC <sup>3</sup>   | PIO                     | DMA area (recv)                              | No                    | Polling + interrupts            | Reliable, unicast: host-level credits, multicast: network-interface-level credits              | Yes (on network interface) |
| PM <sup>4</sup>      | DMA                     | Software TLB* on network interface           | Yes (gang scheduling) | Polling                         | Reliable, ACK/NACK protocol on network interface                                               | Yes (multiple sends)       |
| VMMC <sup>5</sup>    | DMA                     | Software TLB on network interface            | Yes                   | Polling + interrupts            | Reliable, exploits hardware backpressure                                                       | No                         |
| VMMC-2 <sup>6</sup>  | DMA                     | UTLB* in kernel, cached on network interface | Yes                   | Polling + interrupts            | Reliable                                                                                       | No                         |
| LFC <sup>7</sup>     | PIO                     | User translates                              | No                    | Polling + interrupts + watchdog | Reliable, unicast: network-interface-level credits, multicast: network-interface-level credits | Yes (on network interface) |
| Hamlyn <sup>8</sup>  | PIO & DMA               | DMA areas                                    | Yes                   | Polling + interrupts            | Reliable, exploits hardware backpressure                                                       | No                         |
| Trapeze <sup>9</sup> | DMA                     | DMA to page frames                           | No                    | Polling + interrupts            | Unreliable                                                                                     | No                         |
| BIP <sup>10</sup>    | PIO & DMA               | User translates                              | No                    | Polling                         | Reliable, rendezvous and backpressure                                                          | No                         |
| U-Net <sup>11</sup>  | DMA                     | TLB on network interface (U-Net/MM)          | Yes                   | Polling + interrupts            | Unreliable                                                                                     | No                         |

\* PIO is programmed I/O, DMA is direct memory access, TLB is translation look-aside buffer, and UTLB is user-managed TLB.

1. B. Chun, A. Mainwaring, and D. Culler, "Virtual Network Transport Protocols for Myrinet," *IEEE Micro*, Jan. 1998, pp. 53-63.
2. S. Pakin, M. Lauria, and A. Chien, "High Performance Messaging on Workstations: Illinois Fast Messages (FM) for Myrinet," *Proc. Supercomputing '95*, ACM Press, New York, 1995.
3. H.E. Bal et al., "Performance of a High-Level Parallel Language on a High-Speed Network," *J. Parallel and Distributed Computing*, Feb. 1997, pp. 49-64.
4. H. Tezuka et al., "Pin-down Cache: A Virtual Memory Management Technique for Zero-copy Communication," *Proc. Int'l Parallel Processing Symp.*, IEEE CS Press, Los Alamitos, Calif., 1998, pp. 308-314.
5. C. Dubnicki et al., "Design and Implementation of Virtual Memory-Mapped Communication on Myrinet," *Proc. Int'l Parallel Processing Symp.*, IEEE CS Press, Los Alamitos, Calif., 1997, pp. 388-396.

6. C. Dubnicki et al., "Myrinet Communication," *IEEE Micro*, Jan. 1998, pp. 50-52.
7. R.A.F. Bhoedjang, T. Rühl, and H.E. Bal, "Efficient Multicast on Myrinet Using Link-Level Flow Control," *Proc. Int'l Conf. Parallel Processing*, IEEE CS Press, Los Alamitos, Calif., 1998, pp. 381-390.
8. G. Buzzard et al., "An Implementation of the Hamlyn Sender-Managed Interface Architecture," *Proc. Symp. Operating Systems Design and Implementation*, Usenix Assoc., Berkeley, Calif., 1996, pp. 245-259.
9. K. Yocum et al., "Cut-Through Delivery in Trapeze: An Exercise in Low-Latency Messaging," *Proc. Int'l Symp. High Performance Distributed Computing*, IEEE CS Press, Los Alamitos, Calif., 1997, pp. 243-252.
10. L. Prylli and B. Tourancheau, "Protocol Design for High Performance Networking: A Myrinet Experience," Tech. Report 97-22, LIP-ENS Lyons, France, 1997.
11. T. von Eicken et al., "U-Net: A User-Level Network Interface for Parallel and Distributed Computing," *Proc. Symp. Operating System Principles*, ACM Press, New York, Dec. 1995, pp. 303-316.

descriptor. The host may reuse a descriptor only after the network interface has released it.

The receiving host's—host B's—network interface contains a receive ring with descriptors that point to free buffers in its DMA area. Host B's network interface uses this information to determine where to store incoming packets (step 6). If no free host buffer is available, the network interface drops the packet; otherwise it starts a DMA to transfer the packet to host B's memory (step 7). Each host buffer contains a flag the network interface sets as part of the DMA transfer. The host checks if a packet is available by polling the flag of the next host's receive buffer. With each successful poll, a user function is invoked that handles

the packet. This function reads the packet and optionally copies its contents to a user buffer (step 8).

Although the protocol avoids all operating system overhead, keeps the network interface code very simple, and uses little network interface memory, it has several shortcomings:

- *All network transfers go through the DMA area.* As steps 1 and 7 in Figure 1 show, the data packet goes through the DMA areas of both hosts. Although this avoids the cost of kernel calls for each network access, it introduces extra memory copies for applications that send and receive from arbitrary locations.

The simple Myrinet protocol uses five data transfers to communicate a packet.

- *The protocol provides no protection.* As Figure 1 shows, network packets are staged through network interface memory. If the simple protocol lets multiple users access the network interface, the users can read and modify each other's data in the interface memory. They could even modify the interface's control program and use it to access any host memory location.
- *The receiving host uses polling to transfer control.* While simple, polling is not always the most efficient method to detect incoming packets (control transfer). If the host polls too frequently, it will have a high overhead; if it polls too late, it will not respond quickly enough to incoming packets. The alternative to polling, interrupts, is expensive on current operating systems.
- *The protocol is unreliable, even though the hardware is reliable.* If the senders send packets faster than the receiver can handle them, the receiving network interface will run out of buffer space and drop incoming packets.
- *The protocol supports only point-to-point messages.* Multicast is a fundamental component of collective communication operations, such as those supported by the MPI message-passing standard. Although you can implement multicast on top of unicast messages, it is usually inefficient.

The one-way latency of this simple protocol is 12  $\mu$ s and its throughput is 32 Mbytes/s. In contrast, on the same hardware, the highly optimized BIP system has a minimum latency of 4  $\mu$ s and a throughput of 126 Mbytes/s.

## DATA TRANSFERS

As Figure 1 shows, the simple Myrinet protocol uses five data transfers to communicate a packet. Optimized protocols involve at least three transfers: the packet must be moved from the sender's memory to its network interface, a *host-to-interface* transfer; from this network interface to the receiver's network interface, an *interface-to-interface* transfer; and then to the receiving process's address space, an *interface-to-host* transfer. Some network technologies do not stage data through the network interface memory and require only a host-to-network and a network-to-host transfer. Although we describe only the Myrinet case, most design issues (use of programmed I/O versus DMA, pinning, and maximum packet size) also apply to the two-step case.

## HOST TO INTERFACE

On Myrinet, this data transfer can use either DMA or *programmed I/O*.<sup>4</sup> With programmed I/O, the host processor reads the data from host memory and writes

it into network interface memory, typically one or two words at a time, which results in many bus transactions. DMA uses a special DMA engine to transfer the entire packet in large bursts. Also, DMA transfers proceed in parallel with host computations.

On the surface, you might expect DMA to always outperform programmed I/O. However, much depends on the type of host CPU, the DMA engine, and the packet size. The Pentium Pro, for example, supports *write-combining buffers*, which boost the throughput of programmed I/O by combining multiple write commands over the I/O bus into a single bus transaction. Figure 2 shows the host-to-interface throughput using various data transfer mechanisms when copying data from a Pentium Pro to a Myrinet network interface card. As the figure shows, for buffers up to 1,024 bytes, programmed I/O with write combining is faster than DMA because of the DMA start-up cost.

With Myrinet, both a user process and the network interface can start a DMA transfer without any operating system involvement. Because DMA transfers are performed asynchronously, the operating system may decide to swap out the page that happens to be the source or destination of a running DMA transfer. If this happens, part of the transfer's destination will be corrupted. One solution is to let applications *pin* a limited number of pages in their address space, which the operating system can then never swap out. Unfortunately, pinning a page requires a system call, and the amount of memory that can be pinned is limited by the available physical memory and particular operating system policy.

Network interface protocols that use DMA often choose to copy the data into a reserved (and pinned) DMA area, which costs an extra memory copy and thus decreases throughput. As Figure 2 shows, DMA with copying is consistently slower than programmed I/O with write combining. The Shrimp system,<sup>2</sup> in contrast, provides special hardware that lets user processes start DMA transfers without pinning, but this mechanism works only for host-initiated transfers, not for transfers the network interface initiates, so pinning is still required on the receiving side.

With programmed I/O, pinning is not necessary. Even if the operating system swaps out the page during transfer, the next memory reference by the host will generate a page fault, causing the operating system to swap the page back in. As Table 1 shows, many protocols use DMA. FM, FM/MC, and LFC use programmed I/O for host-to-interface transfers; AM-II, Hamlyn, and BIP do so only for small messages.

Another important design choice is the maximum packet size. Large packets typically yield better throughput, because per-packet overhead is incurred fewer times. The throughput of our simple protocol, for example, increases from 32 Mbytes/s to 43 Mbytes/s

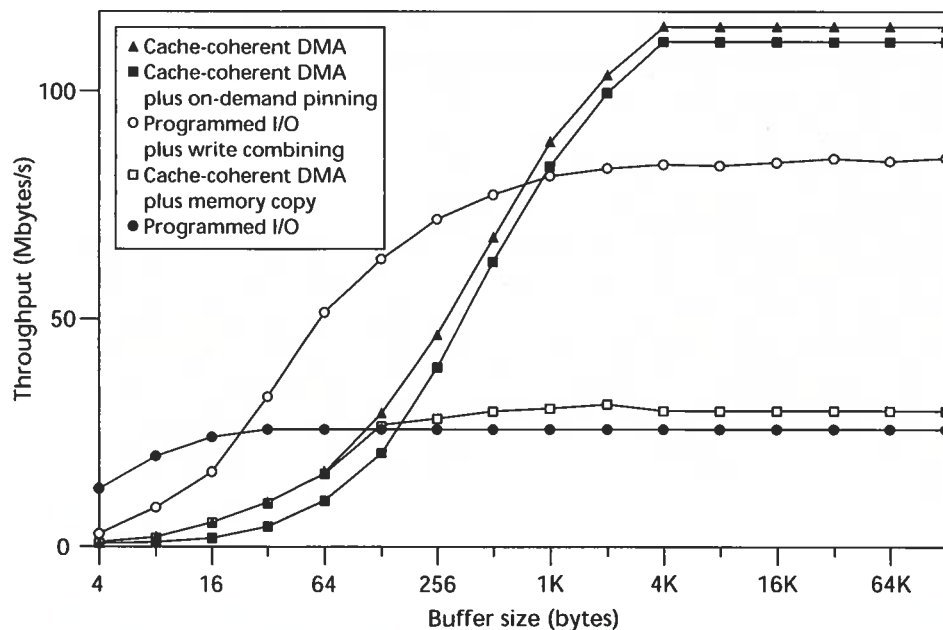


**Figure 2.** Throughput in a host-to-network-interface transfer using different transfer mechanisms. For buffers up to 1,024 bytes, programmed I/O with write combining outperforms DMA.

by using 1,024-byte instead of 256-byte packets. The choice of the maximum packet size is influenced by the system's page size, memory space considerations, and hardware restrictions.

## INTERFACE TO INTERFACE

The interface-to-interface transfer goes through the Myrinet links and switches. All Myrinet protocols use the network interface's DMA engine to send and receive network data. In theory, they could use programmed I/O, but because the network interface processor is slow, DMA transfers are almost always faster.

On Myrinet, the hardware stalls the sending network interface if the receiver does not extract data. To prevent deadlock, however, there is a time limit on the stall. If the receiver does not drain the network within this time limit, the network will reset the network interface or truncate a blocked packet. Many Myrinet control programs deal with this real-time constraint by copying data fast enough to prevent resets. Other protocols use a software flow control scheme to avoid the problem, as we describe later.

## INTERFACE TO HOST

The transfer from network interface to host on the receiving side can again use either DMA or programmed I/O. Because reads over the I/O bus are typically much slower than DMA transfers, most protocols use DMA. Some systems, such as AM-II, use programmed I/O on the host to receive small messages and use DMA for large messages.

## ADDRESS TRANSLATION

The use of DMA transfers between host and network interface memory introduces two problems. First, most systems require that every host memory page involved in a DMA transfer be pinned (with a

system call) to prevent the operating system from replacing that page. Second, typically the network interface's DMA engine must know the physical addresses of each page it transfers data to or from. Operating systems, however, do not export virtual-to-physical mappings to users, so users normally cannot pass physical addresses to the network interface. Even if they could, the interface would have to check the addresses to ensure that users pass only addresses of pages they have access to.

There are several ways to address these problems. One is to avoid all DMA transfers by using programmed I/O. However, the high cost of I/O bus reads means this solution is realistic only on the sending side.

Another approach—and one the simple Myrinet protocol uses—is to require users to copy their data into and out of special DMA areas. These areas need to be pinned only once, when the application opens the device, not during send and receive operations. Address translation is easy because, for each DMA area, the operating system allocates a contiguous chunk of physical memory and passes the area's physical address to the network interface. Users then specify send and receive buffers using an offset in their DMA area, as described earlier. The network interface maintains only the area's physical address and size. AM-II and Hamlyn use this approach, taking the extra copying costs for granted. As Figure 2 shows, however, this copying significantly affects throughput.

In the third approach, the application or a library dynamically pins and unpins the pages that contain send and receive buffers. The network interface then performs DMA transfers directly to those pages. VMCC-2, PM, and BIP use this approach, which follows the cache-coherent DMA curve in Figure 2. However, the network interface must know the current virtual-to-physical mappings of individual pages.

Users write to network interface memory to initialize send descriptors.

but cannot store information for every single virtual page.

BIP and LFC provide a simple kernel module that translates virtual addresses to physical addresses. Users are responsible for pinning their pages and obtaining the physical addresses of these pages from the kernel module. The drawback here is that the network interface cannot check if the physical addresses it receives are valid and if they refer to pinned pages.

An alternative is to let the kernel and network interface cooperate such that the interface can track valid address translations in either hardware or software. Systems like VMMC-2 and U-Net/MM<sup>5</sup> (an extension of U-Net) let the network interface *cache* a limited number of valid address translations that refer to pinned pages. This caching works well for applications that exhibit locality in the pages they use for sending and receiving data. When the network interface finds a translation of a user-specified address in the cache, it can access that address using a DMA transfer. If there is a cache miss, the network interface must interact with the kernel to pin the page and obtain its translation.

In U-Net/MM, the network interface generates an interrupt when it cannot translate an address. The kernel receives the interrupt, looks up the address in its page table, pins the page, and passes the translation to the network interface. In VMMC-2, a library manages address translations for user buffers. It invokes a kernel module to create *references* to address translations and then maps users' virtual addresses to these references. When the network interface receives an unknown reference, it finds the corresponding translation using a DMA transfer to the kernel module's data structure. To avoid such DMA transfers on the critical path, the network interface maintains its own cache of references. The "cache-coherent DMA + on-demand pinning" curve in Figure 2 approximates throughput with this approach.

## PROTECTION

As Figure 1 shows, users write to network interface memory to initialize send descriptors. If multiple users share the network interface, one could corrupt another's send descriptors. The simple protocol avoids this problem by providing user-level network access to, at most, one user at a time—a limitation that is undesirable in many cases.

A straightforward solution to the protection problem is to use the virtual-memory system to give each user access to a different part of the network interface memory.<sup>1</sup> The operating system maps this part into the user's address space when the user opens the device. The virtual-memory hardware then traps all user accesses outside the mapped area. Users write

their commands, and possibly their network data, to their own pages in network interface memory. The network interface must then check each user's page for new requests and process only legal requests.

Because network interface memory is typically small, only a limited number of processes can be given direct access to the network interface in this way. AM-II applies a technique similar to paging to solve this problem. It uses part of its network interface memory to cache active communication end points, and stores inactive end points in host memory. When its network interface receives a message for an inactive end point or when a process tries to send a message via an inactive end point, the interface and operating system cooperate to activate the end point by moving its state to network interface memory, possibly replacing another end point.

Likewise, to maintain protection in the DMA areas, users need their own area. VMMC-2 and U-Net/MM eliminate the need for a fixed DMA area by dynamically pinning and unpinning user pages and by caching translations in network interface memory. BIP also eliminates the DMA area, but does not maintain protection.

## CONTROL TRANSFER

Interrupts are generally expensive (delivery time of 10  $\mu$ s on our hardware), so all user-level architectures support some form of polling. The goal of polling is to give the host a fast mechanism to check if a message has arrived. This check must be inexpensive, because it may be executed very often. A simple approach is to let the network interface set a flag in its memory and to let the host check this flag. This approach is inefficient, however, because every poll results in an I/O bus transfer. In addition, polling traffic will slow down other I/O traffic, including network packet transfers between network interface and host memory.

In the simple Myrinet protocol, the network interface writes a flag in cached host memory (using DMA) when a message is available. The host polls by reading its local memory; because polls are executed frequently, the flag usually resides in the data cache, so failed polls do not generate memory or I/O traffic. When the network interface writes the flag, the host incurs a cache miss and reads the flag from its memory. On a 200-MHz Pentium Pro, this scheme costs 5 ns for a failed poll (a cache hit) and 125 ns for a successful poll (a cache miss). In contrast, each poll over the I/O bus costs 500 ns.

Even if the polling mechanism is efficient, inserting polls manually is tedious and error-prone. Several systems use a compiler or a binary rewriting tool to insert polls in loops and functions. Finding the right polling frequency remains arduous, however. Multiprocessor systems can solve this problem by dedicating a second processor to polling and message handling.

AM-II, FM/MC, LFC, Hamlyn, Trapeze, U-Net,

Existing Myrinet protocols differ widely in the way they address reliability. The most important choice is whether or not to assume that the network is reliable.

VMMC, and VMMC-2 support both interrupts and polling. The receiver usually enables or disables interrupts; sometimes the sender also sets a flag in each packet that determines if an interrupt is to be generated when the packet arrives. To reduce interrupts, LFC implements a polling watchdog<sup>6</sup> on the network interface. This mechanism starts a timer when a message arrives and generates an interrupt only if the host does not issue a poll before the timer expires.

### RELIABILITY

Existing Myrinet protocols differ widely in the way they address reliability. The most important choice is whether or not to assume that the network is reliable. Myrinet has a very low error rate: The risk of a packet getting lost or corrupted is small enough to consider it a fatal event. If necessary, such events can be handled using higher level software (checkpointing, for example).

Many Myrinet protocols assume the hardware is reliable. The advantage of this approach is efficiency, because no retransmission protocol or time-out mechanism is needed. Even if the network is fully reliable, however, the software protocol may still drop packets because of insufficient buffer space—a common cause of packet loss. Each protocol needs communication buffers on both the host and network interface and both are a scarce resource. This problem, which occurs in the simple protocol, can be solved by recovering from buffer overflow or preventing it.

Several other protocols do not assume the network is reliable and so either present an unreliable programming interface, as in U-Net and Trapeze, or implement a retransmission protocol on either the host or network interface. The cost of setting timers and processing acknowledgments is typically no more than a few microseconds.

### RECOVERING FROM OVERFLOW

In PM, which assumes the hardware is reliable, the receiver simply discards incoming data packets if it has no room. It returns an acknowledgment, ACK, to the sender if it accepts the packet and a negative acknowledgment, NACK, if it does not. When an acknowledgment arrives, the receiver processes it completely before it accepts a new packet from the network. A NACK indicates a packet was discarded and needs to be retransmitted. Retransmission continues until an ACK is received. Because PM never drops acknowledgments and assumes that the network hardware is reliable, it need not use time-outs. This protocol is fairly simple; the main disadvantages are the extra acknowledgment messages and the increased network load when retransmissions occur.

### PREVENTING OVERFLOW

The other approach when assuming the hardware

is reliable is to avoid overflow. Some systems use a flow control scheme that blocks the sender if the receiver is running out of buffer space. For large messages, BIP requires the application to use rendezvous-style communication to deal with flow control. The receiver posts a receive request and provides a buffer before a message is sent. Thus, the send of a large message never completes before a receive has been posted. FM and FM/MC implement flow control using a host-level credit scheme. Before a host can send a packet, it needs to have a credit for the receiver, which represents a packet buffer in the receiver's memory. Credits can be handed out in advance (by preallocating buffers for specific senders), but if a sender runs out of credits it must block until it gets new credits.

A host-level credit scheme does not prevent the overflow of network interface buffers, however. With some protocols, the network interface temporarily stops receiving messages if the network interface buffers overflow. Such protocols rely on Myrinet's hardware, link-level, flow-control mechanism (back-pressure) to stall the sender. LFC takes another approach and applies a flow-control scheme to network interface buffers that is similar to the host-level credit scheme of FM and FM/MC. To avoid host buffer overflow, LFC implements additional flow control between a receiving network interface and its host to ensure that the network interface does not release a packet buffer before it has copied the packet's contents to a free host buffer. Flow control at the network interface level also makes it easier to implement multicasting on the network interface. LFC, for example, uses a single flow-control scheme for reliable point-to-point and multicast communication.

### MULTICAST

Hardware support for multicast in switched, worm-hole-routed network technologies like Myrinet is an active research area. The simplest way to implement a multicast in software is to let the sender send a point-to-point message to each multicast destination. However, the point-to-point start-up cost, which is incurred for every multicast destination, includes the data copy to network interface memory, possibly preceded by a copy to a DMA area. PM avoids this repeated copying by passing all multicast destinations to the network interface, which then repeatedly transmits the same packet to each destination.

Although more efficient than a repeated send on the host, this multisend still means the network interface is a serial bottleneck. Most proposals for network-interface-supported multicasting are therefore based on *spanning tree protocols*, which allow multicast packets to be transmitted in parallel (with logarithmic rather

than linear complexity). The system can implement tree-based protocols efficiently by forwarding multicast packets on the network interface instead of on the host. This is particularly efficient when combined with interrupt-driven packet delivery because it removes the interrupt handling costs from the forwarding path. LFC and FM/MC implement the forwarding of multicast packets at the network interface level.<sup>7</sup>

**M**uch research has been done on all these design issues for both Myrinet and other network technologies. In addition to the 11 systems in Table 1, which originate mostly from academic environments, industry has recently developed the Virtual Interface Architecture (<http://www.viarch.org>), a draft standard for user-level communication in cluster environments.

Programmable network interfaces have much flexibility, often compensating for the lack of hardware support in the more advanced interfaces used by massively parallel processors. The polling watchdog device, for example, is easily implemented in the network interface control program, as in LFC. Address translation can be implemented in software on the network interface and the operating system, yielding functionality comparable to the address translation hardware of machines like Meiko's CS2.

Eventually, hardware implementations may be more efficient, but Myrinet's programmable network interface has let us and other researchers quickly and easily experiment with different protocols for commodity network interfaces. As a result, the performance of these protocols has substantially increased. In combination with the economic advantages of commodity networks, this makes such networks a key technology for parallel cluster computing. ♦

#### Acknowledgments

We thank Criel Jacobs, Rutger Hofman, Aske Plaat, Kees Verstoep, and the anonymous *Computer* reviewers for their valuable feedback. We thank Andrew Chien and Scott Pakin for making the FM software available and Mario Lauria and Matt Buchanan for their suggestion to use write combining on the Pentium Pro.

This research is supported in part by a Pionier grant from the Netherlands Organization for Scientific Research.

#### References

1. P. Druschel, L.L. Peterson, and B.S. Davie, "Experiences with a High-Speed Network Adaptor: A Software Perspective," *Proc. Conf. Communications Architectures,*

*Protocols, and Applications*, ACM Press, New York, 1994, pp. 2-13.

2. M.A. Blumrich et al., "Design Choices in the SHRIMP System: An Empirical Study," *Proc. Int'l Symp. Computer Architecture*, IEEE CS Press, Los Alamitos, Calif., 1998, pp. 330-341.
3. N.J. Boden et al., "Myrinet: A Gigabit-per-Second Local Area Network," *IEEE Micro*, Jan./Feb. 1995, pp. 29-36.
4. P.A. Steenkiste, "A Systematic Approach to Host Interface Design for High-Speed Networks," *Computer*, Mar. 1994, pp. 47-57.
5. M. Welsh et al., "Memory Management for User-Level Network Interfaces," *IEEE Micro*, Mar./Apr. 1998, pp. 77-82.
6. O. Maquelin et al., "Polling Watchdog: Combining Polling and Interrupts for Efficient Message Handling," *Proc. Int'l Symp. Computer Architecture*, IEEE CS Press, Los Alamitos, Calif., 1996, pp. 179-188.
7. K. Verstoep, K.G. Langendoen, and H.E. Bal, "Efficient Reliable Multicast on Myrinet," *Proc. Int'l Conf. Parallel Processing*, Vol. III, IEEE CS Press, Los Alamitos, Calif., 1996, pp. 156-165.

*Raoul A.F. Bhoedjang is a researcher in computer science at the Vrije Universiteit, where his research interests are in parallel programming and the structure of communication architectures. He received an MSc in computer science from the Vrije Universiteit. He is a member of the ACM.*

*Tim Rühl is a researcher in computer science at the Vrije Universiteit and a software engineer at Data Distilleries B.V. His research interests are advanced parallel programming models and their efficient implementation. He received an MSc in computer science from the Vrije Universiteit. He is a member of the ACM and the IEEE.*

*Henri E. Bal is a professor in the Faculty of Sciences at the Vrije Universiteit, where he heads research groups on parallel programming and physics-applied computer science. His research interests include parallel and distributed programming, network computing, applications, and programming languages. He received an MSc in mathematics from Delft University of Technology and a PhD in computer science from the Vrije Universiteit.*

*Contact the authors at the Dept. of Mathematics and Computer Science, Faculty of Sciences, Vrije Universiteit, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands, {raoul, tim, bal}@cs.vu.nl.*