

Exhibit C-18 to Defendants' Invalidity Contentions:
Comparison of U.S. Patent No. 7,594,052 ("’052 patent") and A. Radulescu, et al., *An Efficient On-Chip Network Interface Offering Guaranteed Services, Shared Memory Abstraction, and Flexible Network Configuration* ("2004 Radulescu")

As described in the following claim chart, claim 6 of U.S. Patent No. 7,594,052 ("’052 patent") is invalid under 35 U.S.C. §§ 102 and/or 103 in view of A. Radulescu, et al., *An Efficient On-Chip Network Interface Offering Guaranteed Services, Shared Memory Abstraction, and Flexible Network Configuration* ("2004 Radulescu"), which was published in the Proceedings of the Design, Automation and Test in Europe Conference on February 16, 2004. The citations below are to 2004 Radulescu and compares 2004 Radulescu to the ’052 patent claims.¹

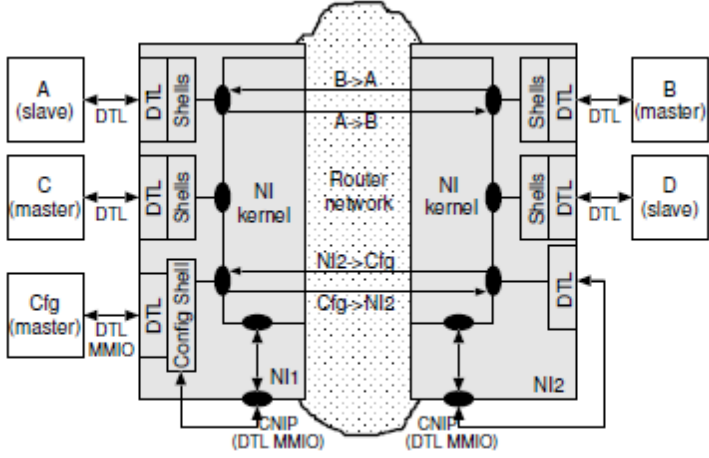
To the extent Plaintiff alleges that 2004 Radulescu does not disclose any particular limitation of claim 6 (the "Asserted Claim") of the ’052 patent, either expressly or inherently, it would have been obvious to a person of ordinary skill in the art as of the priority date of the ’052 patent to modify 2004 Radulescu and/or to combine the teachings of 2004 Radulescu with other prior art references, including but not limited to the prior art references presently found in Defendants' Invalidity Contentions and any supplements thereto and the relevant section of charts for other prior art for the ’052 patent in a manner that would have rendered the Asserted Claim invalid as obvious. The inclusion of the particular limitation into the methods and systems described in 2004 Radulescu would have been obvious for at least the following reasons: it would have been obvious to modify the methods and/or systems disclosed in 2004 Radulescu, alone or in combination with the references identified in Defendants' Invalidity Contentions, to meet this limitation based on: (a) the teachings in 2004 Radulescu itself; (b) a person of ordinary skill in the art implementing a predictable variation of the methods and/or systems disclosed in 2004 Radulescu or in the other identified references; (c) the logic, judgment, intelligence, creativity, innovation and common sense a person of ordinary skill in the art would employ; (d) the inferences and creative and routine steps that a person of ordinary skill in the art would employ; (e) market demand for methods and/or systems to include this limitation;

¹ Nothing in these claim charts should be construed as an admission regarding infringement, either literally or under the doctrine of equivalents, or as an admission regarding Defendants' understanding of the proper scope of the Asserted Claim. Given Plaintiff's vague and ambiguous infringement contentions, the exemplary citations herein necessarily account for a variety of possible infringement arguments and claim constructions. Defendants reserve the right to rely on additional citations or sources of evidence that also may be applicable, or that may become applicable in light of claim construction, changes in Plaintiff's infringement contentions, and/or information obtained during discovery as the case progresses. Various positions put forth in this document are predicated on Plaintiff's overly broad interpretation of its claims as evidenced by the infringement allegations received to date. Those positions are not intended to and do not necessarily reflect Defendants' interpretation of the true and proper scope of Plaintiff's claims, and Defendants reserve the right to adopt claim construction positions that differ from or even conflict with various positions put forth in this document.

and/or (f) design needs or market pressures to solve a problem with a finite number of solutions that could have produced the desired outcome.

A. Claim 6

'052 Patent Claim Limitation	2004 Radulescu
[6pre-a] Method of communication service mapping in an integrated circuit, having a plurality of processing modules (M, S),	To the extent the preamble is limiting, 2004 Radulescu discloses and/or renders obvious this part of the preamble based on at least the following citations. See, e.g.: • 2004 Radulescu at 1 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. • 2004 Radulescu at 2 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

'052 Patent Claim Limitation	2004 Radulescu
	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. * * * * 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. 2004 Radulescu at Fig. 8  The diagram illustrates the NI configuration for a router network. It shows two NI kernels, NI1 and NI2, connected via a central Router network. NI1 is connected to a Config Shell, which in turn connects to a DTL MMIO block. NI2 is connected to a DTL MMIO block. Both NI1 and NI2 have DTL shells. NI1 is connected to slave modules A and C, and master module B. NI2 is connected to master module B and slave module D. The diagram also shows data flow between the NI kernels and the router network, including labels like B>A, A>B, NI2>Cfg, and Cfg>NI2. Figure 8. NI configuration

'052 Patent Claim Limitation	2004 Radulescu
	See also, e.g., 2004 Radulescu at 3–5, Figs. 1–9. Furthermore, to the extent this part of the preamble is not disclosed in 2004 Radulescu, it (and the claim as a whole) would have been obvious based on the combinations identified in Defendants’ Invalidity Contentions.
[6pre-b] wherein at least one first of said processing modules (M) requests at least one communication service to at least one second processing module (S) based on specific communication properties and at least one communication service identification,	To the extent the preamble is limiting, 2004 Radulescu discloses and/or renders obvious this part of the preamble based on at least the following citations. See, e.g.: • 2004 Radulescu at 1 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 <i>decoupling between computation and communication</i> [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 <i>guaranteed services</i> 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 <i>Æthereal</i>, offers a <i>shared-memory abstraction</i> to the IP modules. Communication is performed using a transaction-based protocol, where master IP modules issue <i>request messages</i> (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 <i>response message</i> (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.

'052 Patent Claim Limitation	2004 Radulescu
	• 2004 Radulescu at 2 2 NoC Services As mentioned in the previous section, the communication services of the <i>Æ</i>thereal 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 <i>end to end</i> 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 <i>transactions</i> [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).

- 2004 Radulescu at Figs. 3-6

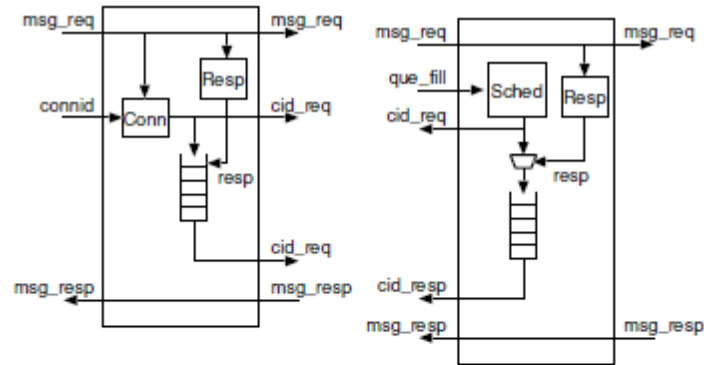


Figure 3. Narrowcast shell **Figure 4. Multi-connection shell**

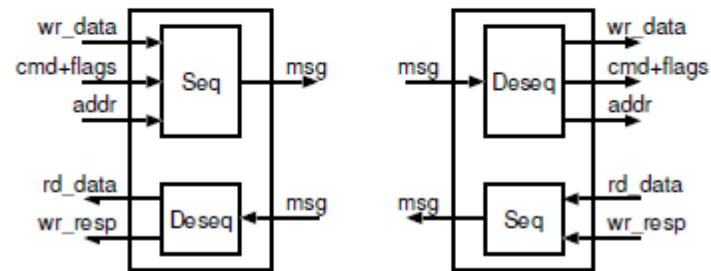


Figure 5. Master shell **Figure 6. Slave shell**

- 2004 Radulescu at 4

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.

'052 Patent Claim Limitation	2004 Radulescu
	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 (<i>Conn</i> 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 multiconnection 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. 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.

- 2004 Radulescu at Fig. 9

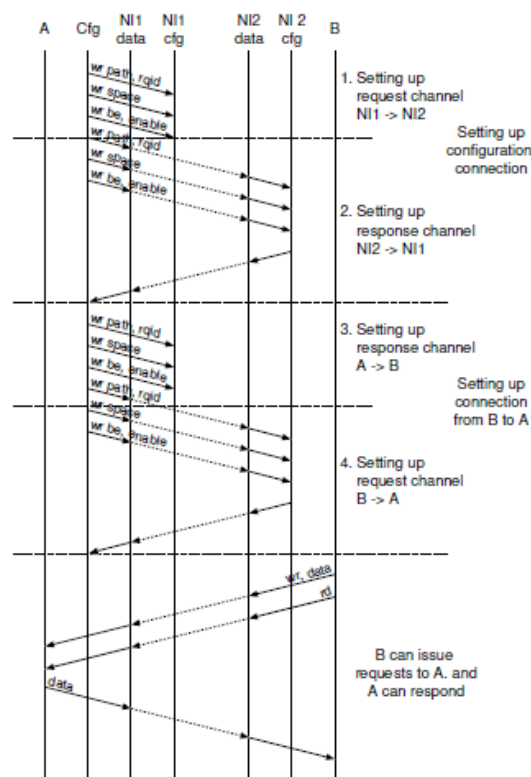
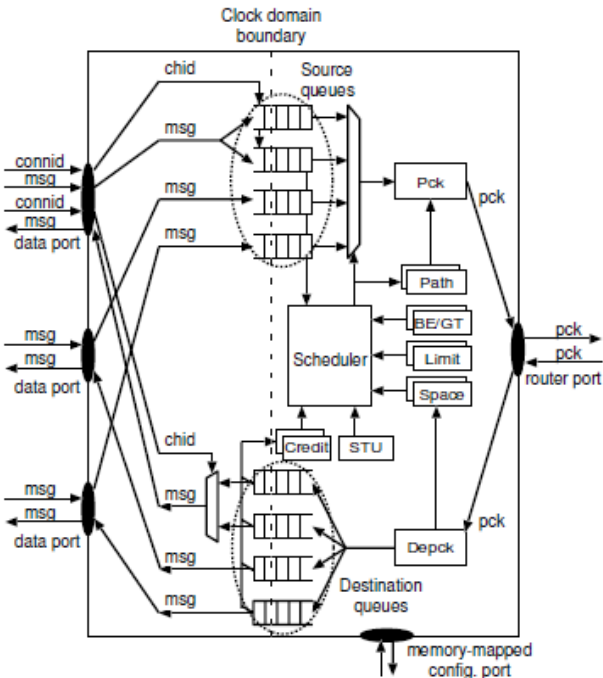


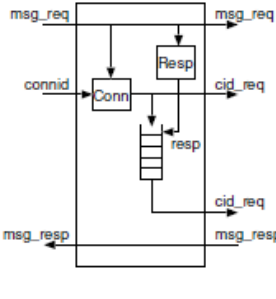
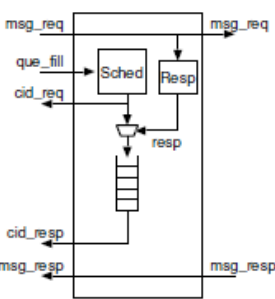
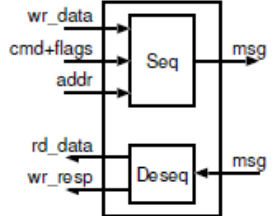
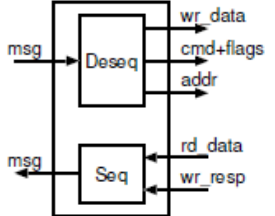
Figure 9. Connection configuration example

- 2004 Radulescu at 5

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

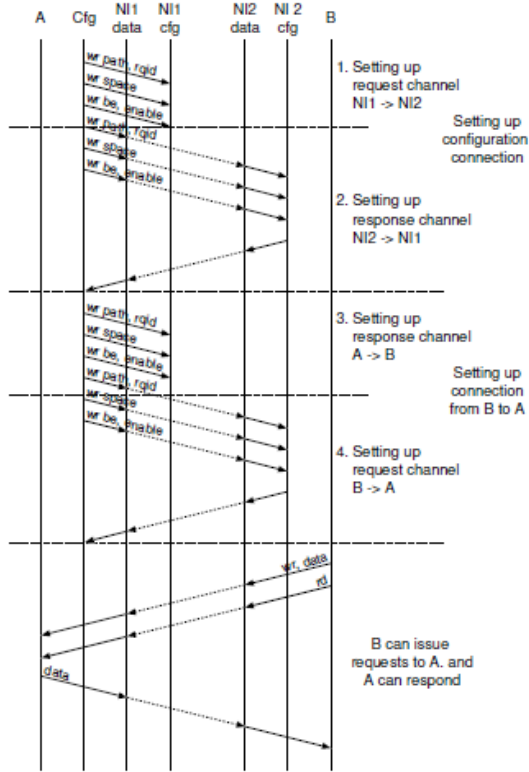
'052 Patent Claim Limitation	2004 Radulescu
	necessary registers in NI1 (Step 1 in Figure 9). Second, we use this channel to set up (via the NoC) the response channel from NI2's CNIP to Cfg (Step 2). The three shown messages are delivered and executed in order at NI2. The last of them also requests an acknowledgment message to confirm that the channel has been successfully set up. 2004 Radulescu at Fig. 2  Figure 2. Network interface kernel 2004 Radulescu at 3 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

'052 Patent Claim Limitation	2004 Radulescu
	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 <i>ports</i>. 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 <i>connid</i> signals to select on which connection a message is supplied or consumed. • 2004 Radulescu at 6 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. Furthermore, to the extent this part of the preamble is not disclosed in 2004 Radulescu, it (and the claim as a whole) would have been obvious based on the combinations identified in Defendants' Invalidity Contentions.
[6pre-c] wherein said at least one communication service identification comprises at least one communication thread or at least one address range, said address range for identifying one or more second processing modules (S) or a	To the extent the preamble is limiting, 2004 Radulescu discloses and/or renders obvious this part of the preamble based on at least the following citations. See, e.g.: • 2004 Radulescu at 2 Backward compatibility with existing protocols, such as AXI or DTL, is achieved by using a model based on <i>transactions</i> [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

'052 Patent Claim Limitation	2004 Radulescu
memory region within said one or more second processing modules (S), comprising the steps 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). 2004 Radulescu at Figs. 3-6   Figure 3. Narrowcast shell Figure 4. Multi-connection shell   Figure 5. Master shell Figure 6. Slave shell 2004 Radulescu at 4 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,

'052 Patent Claim Limitation	2004 Radulescu
	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 multiconnection 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. 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. See also, e.g., 2004 Radulescu at 1–3, 5–6, Figs. 1–9. Furthermore, to the extent this part of the preamble is not disclosed in 2004 Radulescu, it (and the claim as a whole) would have been obvious based on the combinations identified in Defendants' Invalidity Contentions.

'052 Patent Claim Limitation	2004 Radulescu
[6b] enabling a connection based communication having a set of connection properties,	2004 Radulescu discloses and/or renders obvious this limitation based on at least the following citations. See, e.g.: • 2004 Radulescu at 1 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 <i>decoupling between computation and communication</i> [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). • 2004 Radulescu at 2 The <i>Æthereal</i> NoC offers its services on <i>connections</i>, 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 <i>channels</i> (between a single master and a single slave). To each channel, <i>properties</i> are attached, such as guaranteed message delivery or not, in order or unordered 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 <i>guarantees</i>. In <i>Æthereal</i>, 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

'052 Patent Claim Limitation	2004 Radulescu
	cost compared to alternatives, such as rate-based packet switching [29], or deadline-based packet switching [20]. 2004 Radulescu at Fig. 9  Figure 9. Connection configuration example

'052 Patent Claim Limitation	2004 Radulescu
	• 2004 Radulescu at 5 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. See also, e.g., 2004 Radulescu at 3–4, Figs. 1–6. Furthermore, to the extent this limitation is not disclosed in 2004 Radulescu, this limitation (and the claim as a whole) would have been obvious based on the combinations identified in Defendants' Invalidity Contentions.
[6c] controlling the communication between said at least one first of said plurality of processing modules (M) and said interconnect means (N) by at least one network interface (NI) associated to said at least one first of said processing modules,	2004 Radulescu discloses and/or renders obvious this limitation based on at least the following citations. See, e.g.: • 2004 Radulescu at Fig. 1

'052 Patent Claim Limitation	2004 Radulescu
	The diagram illustrates the architecture of a Network Interface (NI) divided into user and network spaces. In the user space, there are DTL and AXI adapters. In the network space, there are NI kernel ports, multicast and narrowcast shells, and the NI kernel. Data flows from user applications through the adapters into the NI kernel, which then connects to a router via NI kernel ports. Figure 1. NI kernel and shells 2004 Radulescu at 3 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 <i>kernel</i>, 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 <i>shells</i>, 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.

- 2004 Radulescu at Fig. 8

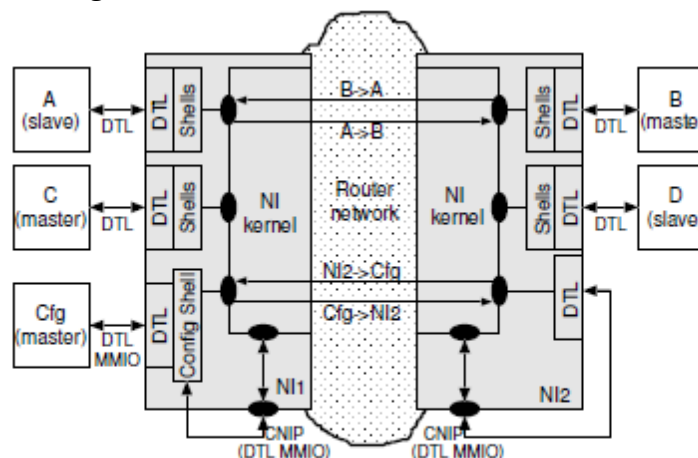


Figure 8. NI configuration

- 2004 Radulescu at 5

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.

- 2004 Radulescu at 6

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.

'052 Patent Claim Limitation	2004 Radulescu
	See also, e.g., 2004 Radulescu at 1–5, Figs. 2—7. Furthermore, to the extent this limitation is not disclosed in 2004 Radulescu, this limitation (and the claim as a whole) would have been obvious based on the combinations identified in Defendants’ Invalidity Contentions.
[6d] mapping the requested at least one communication service based on said specific communication properties to a connection based on a set of connection properties according to said at least one communication service identification.	2004 Radulescu discloses and/or renders obvious this limitation based on at least the following citations. See, e.g.: • 2004 Radulescu at 1-2 Our NoC, called <i>Æthereal</i>, offers a <i>shared-memory abstraction</i> to the IP modules. Communication is performed using a transaction-based protocol, where master IP modules issue <i>request messages</i> (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 <i>response message</i> (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).

- 2004 Radulescu at 2

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).

* * * *

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 unordered 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

'052 Patent Claim Limitation	2004 Radulescu
	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 <i>guarantees</i>. In <i>Æthereal</i>, 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].

- 2004 Radulescu at Figs. 3-7

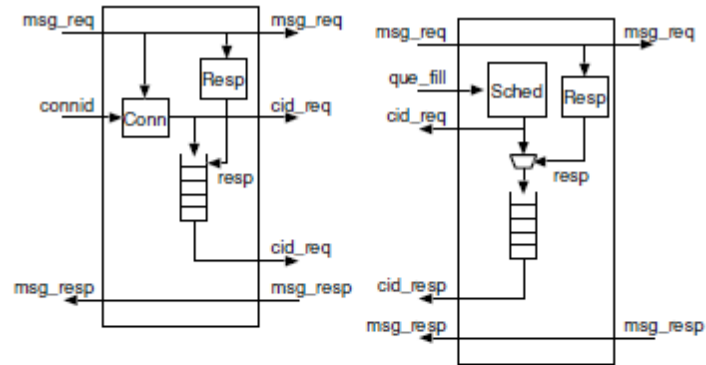


Figure 3. Narrowcast shell Figure 4. Multi-connection shell

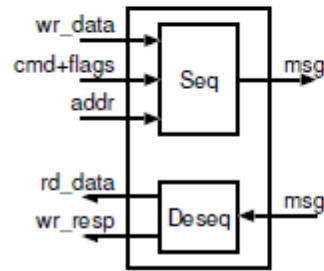


Figure 5. Master shell

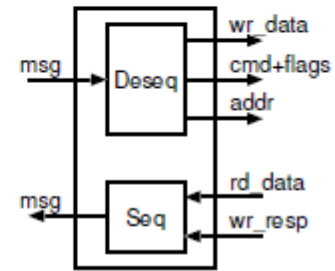


Figure 6. Slave shell

'052 Patent Claim Limitation	2004 Radulescu																																														
	<table><tr><td>cmd</td><td>length</td><td>flags</td><td></td><td>seq no.</td><td>trans id</td></tr><tr><td colspan="6">address</td></tr><tr><td colspan="6">write data 1</td></tr><tr><td colspan="6">. . .</td></tr><tr><td colspan="6">write data N</td></tr></table> Request message format <table><tr><td>error</td><td></td><td>seq no.</td><td>trans id</td></tr><tr><td colspan="4">read data 1</td></tr><tr><td colspan="4">. . .</td></tr><tr><td colspan="4">read data N</td></tr></table> Response message format Figure 7. Message format examples 2004 Radulescu at 4 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 (<i>Conn</i> 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	cmd	length	flags		seq no.	trans id	address						write data 1						. . .						write data N						error		seq no.	trans id	read data 1				. . .				read data N			
cmd	length	flags		seq no.	trans id																																										
address																																															
write data 1																																															
. . .																																															
write data N																																															
error		seq no.	trans id																																												
read data 1																																															
. . .																																															
read data N																																															

'052 Patent Claim Limitation	2004 Radulescu
	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 multiconnection 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. 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.

- 2004 Radulescu at Fig. 9

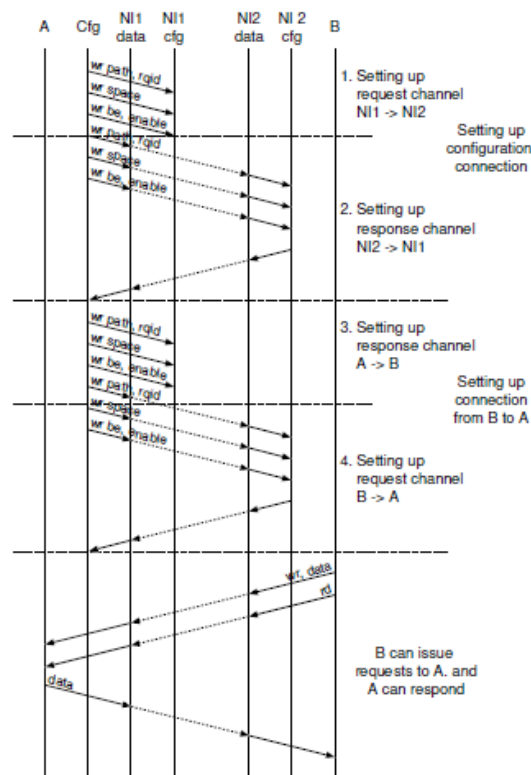
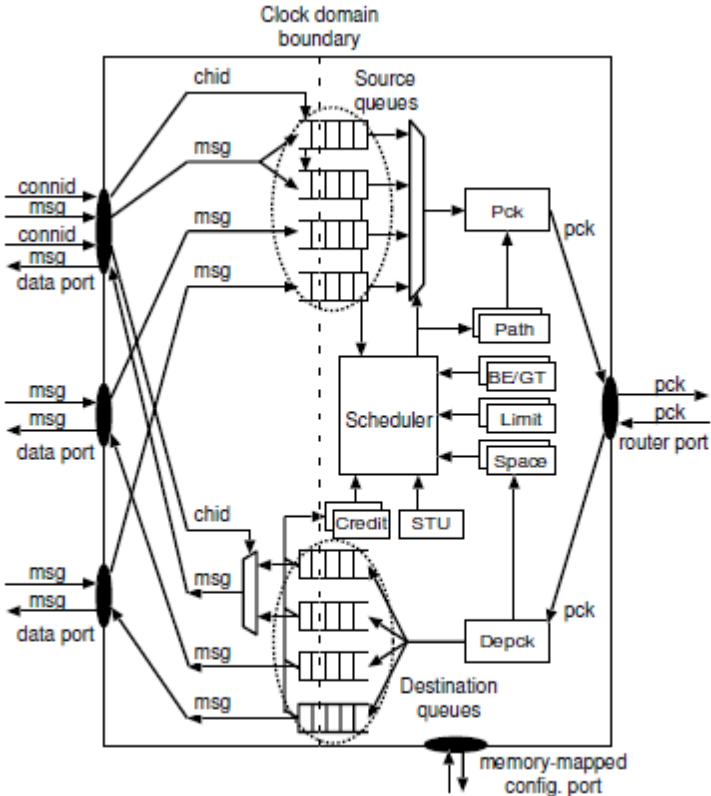


Figure 9. Connection configuration example

- 2004 Radulescu at 5

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)

'052 Patent Claim Limitation	2004 Radulescu
	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. 2004 Radulescu at Fig. 2  Figure 2. Network Interface kernel

'052 Patent Claim Limitation	2004 Radulescu
	• 2004 Radulescu at 3 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 <i>ports</i>. 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 <i>connid</i> signals to select on which connection a message is supplied or consumed. • 2004 Radulescu at 6 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. See also, e.g., 2004 Radulescu at 1–6, Figs. 1–9. Furthermore, to the extent this limitation is not disclosed in 2004 Radulescu, this limitation (and the claim as a whole) would have been obvious based on the combinations identified in Defendants’ Invalidity Contentions.