

IPR2024-00566

Declaration of Henry Houh, Ph.D. Under 37 C.F.R. § 1.68 in Support of
Petition for *Inter Partes* Review of U.S. Patent Nos. 9,367,657

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

ARM LIMITED

Petitioner,

v.

ICPILLAR LLC

Patent Owner.

U.S. Patent No. 9,367,657

Inter Partes Review No.: IPR2024-00566

DECLARATION OF HENRY HOUH, Ph.D., UNDER 37 C.F.R. § 1.68 IN
SUPPORT OF PETITION FOR *INTER PARTES* REVIEW OF U.S.
PATENT NO. 9,367,657

Mail Stop: Patent Board
Patent Trial and Appeal Board
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

TABLE OF CONTENTS

I, HENRY HOUH, PH.D. HEREBY DECLARE AS FOLLOWS:	1
I. INTRODUCTION.....	1
II. BACKGROUND AND QUALIFICATIONS	3
III. MATERIALS CONSIDERED FOR THIS DECLARATION.....	13
IV. SUMMARY OF OPINIONS	13
V. OVERVIEW OF THE ‘657 PATENT, ITS PATENT FAMILY, AND ITS PROSECUTION HISTORY.....	14
A. Overview of the ‘657 Patent.....	14
B. Overview of the Prosecution History of the ‘657 Patent and Related Applications	24
i. Prosecution History of the application leading to the 657 Patent (application 14/584,897)	24
VI. LEVEL OF ORDINARY SKILL IN THE PERTINENT ART	27
VII. CLAIM CONSTRUCTION	28
VIII. UNDERSTANDING OF THE LAW	29
IX. SCOPE AND CONTENT OF THE PRIOR ART	31
A. Rompaey	31

B. Banerjee 36

C. Background Prior Art 40

1. Sugiimoto.....40

2. Kobayashi 45

3. Fletcher 48

X. DETAILED UNPATENTABILITY ANALYSIS FOR THE ‘657 49

**A. Ground 2: The Challenged Claims Are Obvious In View of
Rompaey alone or in combination with CoWare I.....50**

1. Rompaey Discloses, And/ Or Renders Obvious, The Challenged
Claims 51

a) Claim 1 51

Preamble: “A method for designing an integrated circuit which
controls a device when the integrated circuit is integrated into the
device comprising:” 51

i.	1[a] “selecting a value for a control type parameter associated with an electronic device from a hierarchal classification database;.....53 selecting a value for a display type parameter associated with an electronic device from a hierarchal classification database.....53 selecting a value for an input type parameter for the electronic device from the hierarchal classification database;.....53 selecting a value for an output type parameter for the electronic device from the hierarchal classification database;.....53 receiving the selected values for the control type, display type, input type and output type parameters into a memory of a processor programmed by software commands to design a hardware integrated circuit and associated operating software for controlling the hardware integrated circuit to control the electronic device” ..54 (a) Rompaey’s Selection Actions From A Hierarchal Classification Database 54 (b) Rompaey’s “Parameter” Types.....108 (c) Rompaey Renders This Element Obvious Alone Or In Combination With CoWare I. 133
----	--

- ii. 1[b]: “analyzing by the processor the selected values for the control type, display type, input type and output type parameters of the electronic device;” 142
 - iii. 1[c] “outputting by the processor a hardware integrated circuit design based on the selection of input and output characteristics of the electronic device, the hardware integrated circuit including a communication bus for communicating with the electronic device; and” 152
 - iv. 1[d] “outputting by the processor software programming commands to be .executed by the hardware integrated circuit for controlling the electronic device in accordance with the selection of categorized input and output characteristics of the electronic device through the communication bus of the hardware integrated circuit using a communication protocol compatible with the electronic device and the hardware integrated circuit.” 159
- b) Claim 5: “The method of claim 1, wherein the hardware integrated circuit design is based on at least one predetermined hardware integrated circuit component which is able to control one

of the selected values for the control type, display type, input type and output type parameters.”	164
c) Claim 6: “The method of claim 5, wherein the hardware integrated circuit design is provided as a software schematic, layout, or both.”	167
d) Claim 8: “The method of claim 1, wherein the hierarchal classification database is a component of a fabless development kit.” 168	
e) Claim 9: Preamble “A system comprising:”	170
i. 9[a] “one or more processors; a design user interface provided by programming commands executed by the one or more processors, the design user interface allowing a user to characterize an electronic device by selecting a value for a control type parameter, a value for a display type parameter, a value for an input type parameter and a value for an output type parameter, all of which are associated with the operation and control of the electronic device;”	170

- ii. 9[b] “logic encoded in one or more non-transitory computer-readable media that, when executed by the one or more processors, is operable to.” 171
- iii. 9[c]: “receive the selected control type, display type, input type and output type parameter values selected by the user” 173
- iv. 9[d] “into a memory associated with the one or more processors, the logic executed by the one or more processors operable to design a hardware integrated circuit design based on the selection of input and output characteristics of the device, the hardware integrated circuit design including a communication bus, and” ..173
- v. 9[e] “output software programming commands to be executed by the hardware integrated circuit for controlling the device in accordance with the selected control type, display type, input type and output type parameter values selected by the user using a communication protocol compatible with the device and the hardware integrated circuit through the communication bus of the hardware integrated circuit.” 174
- f) Claim 15: “The system of claim 9, wherein the one or more non-transitory computer-readable media further comprises logic that,

when executed by the one or more processors, is further operable to
simulate the hardware integrated circuit's functionality based on the
hardware integrated circuit design” 175

g) Claim 16: “The system of claim 9, wherein the design user
interface is in communication with a hierarchal classification
database, and the user selects the selected control type, display type,
input type and output type parameter values from the hierarchal
classification database” 177

B. Ground 1: The Challenged Claims Are Obvious Over Banerjee..179

1. Banerjee Discloses, And/ Or Renders Obvious, The Challenged
Claims. 181

a) Claim 1: Preamble: “A method for designing an integrated
circuit which controls a device when the integrated circuit is
integrated into the device comprising” 181

i. Element 1[a] “selecting a value for a control type parameter
associated with an electronic device from a hierarchal
classification database;.....182

selecting a value for a display type parameter associated with an electronic device from a hierarchal classification database.....	182
selecting a value for an input type parameter for the electronic device from the hierarchal classification database;.....	182
selecting a value for an output type parameter for the electronic device from the hierarchal classification database;.....	182
receiving the selected values for the control type, display type, input type and output type parameters into a memory of a processor programmed by software commands to design a hardware integrated circuit and associated operating software for controlling the hardware integrated circuit to control the electronic device”	182
ii. Element 1[b] “analyzing by the processor the selected values for the control type, display type, input type and output type parameters of the electronic device;”	226
iii. Element 1[c] “outputting by the processor a hardware integrated circuit design based on the selection of input and output characteristics of the electronic device, the hardware integrated circuit including a communication bus for communicating with the electronic device; and”	234

- iv. Element 1[d] “outputting by the processor software programming commands to be .executed by the hardware integrated circuit for controlling the electronic device in accordance with the selection of categorized input and output characteristics of the electronic device through the communication bus of the hardware integrated circuit using a communication protocol compatible with the electronic device and the hardware integrated circuit.” 236
- b) Claim 5: “The method of claim 1, wherein the hardware integrated circuit design is based on at least one predetermined hardware integrated circuit component which is able to control one of the selected values for the control type, display type, input type and output type parameters.” 240
- c) Claim 6: “The method of claim 5, wherein the hardware integrated circuit design is provided as a software schematic, layout, or both.” 241
- d) Claim 8: “The method of claim 1, wherein the hierarchal classification database is a component of a fabless development kit.”

- e) Claim 9: Preamble “A system comprising:” 245
- i. Element 9[a] “one or more processors; a design user interface
provided by programming commands executed by the one or more
processors, the design user interface allowing a user to characterize
an electronic device by selecting a value for a control type
parameter, a value for a display type parameter, a value for an
input type parameter and a value for an output type parameter, all
of which are associated with the operation and control of the
electronic device;” 246
- ii. Element 9[b] “logic encoded in one or more non-transitory
computer-readable media that, when executed by the one or more
processors, is operable to.” 248
- iii. Element 9[c] “receive the selected control type, display type, input
type and output type parameter values selected by the user” 249
- iv. Element 9[d] “into a memory associated with the one or more
processors, the logic executed by the one or more processors
operable to design a hardware integrated circuit design based on
the selection of input and output characteristics of the device, the

- hardware integrated circuit design including a communication bus,
and” 249
- v. Element 9[e] “output software programming commands to be
executed by the hardware integrated circuit for controlling the
device in accordance with the selected control type, display type,
input type and output type parameter values selected by the user
using a communication protocol compatible with the device and
the hardware integrated circuit through the communication bus of
the hardware integrated circuit.” 250
- f) Claim 15: “The system of claim 9, wherein the one or more
non-transitory computer-readable media further comprises logic that,
when executed by the one or more processors, is further operable to
simulate the hardware integrated circuit's functionality based on the
hardware integrated circuit design” 250
- g) Claim 16: “The system of claim 9, wherein the design user
interface is in communication with a hierarchal classification
database, and the user selects the selected control type, display type,

input type and output type parameter values from the hierarchal
classification database” 251

**XI. SECONDARY CONSIDERATIONS DO NOT REBUT THE PRIMA
FACIE CASE OF OBVIOUSNESS 252**

XII. CONCLUSION.....254

HENRY H. HOUH.....256

Education.....256

Experience 256

Honors.....260

Patents and Patent Publications.....260

Trials, Testimony and Depositions.....261

Publications 266

Leadership, Activities and Interests 269

I, Henry Houh, Ph.D. hereby declare as follows:

I. INTRODUCTION

1. I have been retained as an expert witness on behalf of Arm Limited (Petitioner) for the above-captioned Petition(s) for *Inter Partes* Review (“IPR”) of claims 1, 5, 6, 8, 9, 15, and 16 of U.S. Patent No. 9,367,657 (“the ‘657 patent”) (the Challenged Claims). I am being compensated for my time in connection with this IPR at my standard consulting rate of \$820 per hour. My compensation is in no way dependent on the outcome of this matter.
2. I have been asked to provide my opinions regarding whether the Challenged Claims are unpatentable as obvious. In preparing this declaration, I have reviewed the ‘657 patent, the file history of the ‘657 patent, and numerous prior art references from the time of the alleged invention. I have also reviewed the patent, file history, and other materials for the parent patent of the ‘657 patent, U.S. Pat. No. 8,924,899 (the “‘899 patent”). I submitted a declaration in IPR2024-00467 for the parent ‘899 patent.
3. In forming the opinions expressed in this declaration, I relied upon my education and experience in the relevant field of the art, and have

considered the viewpoint of a person having ordinary skill in the relevant art, as of May 2013.

4. I understand that ‘657 claims priority through the parent ‘899 patent to an application filed on May 23, 2013. My analysis as set forth below uses that date as the priority date of the ‘657 patent and the Challenged Claims.
5. My opinions directed to the unpatentability of the challenged claims are based, at least in part, on the following prior art publications:

Reference	Date of Public Availability
Rompaey, U.S. Pat. No. 5,870,588	Rompaey was filed on October 23, 1996, and issued on February 9, 1999. Rompaey is prior art under at least the pre-AIA version of 35 U.S.C. § 102(a), (b), (e), and (g) and the post-AIA version of 35 U.S.C. § 102(a)(1), (a)(2) and (d). Rompaey is attached as Exhibit 1004 to the IPR.
Banerjee, U.S. Pat. Appl. No. 2008/0222581	Banerjee was filed on March 9, 2007, and published on September 11, 2008. Banerjee

	is prior art under at least pre-AIA version of 35 U.S.C. § 102(a), (b) (e), and (g) and the post-AIA version of 35 U.S.C. § 102(a)(1), (a)(2) and (d). Banerjee is attached as Exhibit 1005 to the IPR.
CoWare I	<i>CoWare—A design environment for heterogeneous hardware/software systems, Verkest, D., Van Rompaey, K., Bolsens, I. et al. Des Autom Embed Syst 1, 357–386 (1996).</i> CoWare I is prior art under at least pre-AIA version of 35 U.S.C. § 102(a), (b) and the post-AIA version of 35 U.S.C. § 102(a)(1), (a)(2) and (d). CoWare I is attached as Exhibit 1007 to the IPR.

II. BACKGROUND AND QUALIFICATIONS

6. My background and expertise that qualify me as an expert in the technical issues in this case are as follows:

7. As indicated on my *Curriculum Vitae* attached as **Exhibit A**, I received a Ph.D. in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology (“MIT”) in 1998. I also received a Master of Science degree in Electrical Engineering and Computer Science in 1991, a Bachelor of Science Degree in Electrical Engineering and Computer Science in 1989, and a Bachelor of Science Degree in Physics in 1990, all from MIT.
8. During my college studies, I focused on communications and data networking. I took specialized courses including graduate courses in telecommunications networks, optical communications, and data networking. I, along with other graduate students in a networking research group, maintained both the computer workstations and the networking devices in the research group.
9. I also served as a teaching assistant for the Computation Structures core course a number of times. At the time, the course involved teaching computer architecture at a low level, and the laboratory component involved building a CPU from discrete components at the gate level and higher. When I was enrolled in this course as an undergraduate student, I redesigned the hardware and software, including microcode and nanocode

of this CPU, to achieve an average speed up of roughly thirty times. Based in part on this, I became a laboratory assistant, head laboratory assistant, and then a teaching assistant in my second term senior year, which is unusual in that teaching assistantships at MIT are normally reserved for graduate students. I further became head teaching assistant and also taught some recitation sections.

10. As part of my doctoral research at MIT from 1991-1998, I worked as a research assistant in the Telemedia Network Systems (“TNS”) group at the Laboratory for Computer Science. The TNS group built a high-speed gigabit ATM network and applications which ran over the network, such as remote video capture (including audio), processing and display on computer terminals. In addition to helping design the core network components (such as the ATM switch), designing and building the high-speed ATM links, and designing and writing the device drivers for the interface cards, I also set up the group’s web server, which at the time was one of the first several hundred web servers in existence. I used programmable logic devices, such as programmable array logic and field programmable gate arrays, and their software development environments, in the design of these components.

Our high-speed data network carried multimedia data including video and audio data within ATM cells.

11. The TNS group was the first group to initiate a remote video display over the World Wide Web. Vice President Al Gore visited our group in 1996 and received a demonstration of—and remotely drove—a radio-controlled toy car with a wireless video camera mounted on it; the video was encoded by TNS-designed hardware, streamed over the TNS-designed network and displayed using TNS-designed software. During this time, I also modified a web server to determine the location of client, and present different web pages depending on location.

12. I authored or co-authored twelve papers and conference presentations on our group's research. I also co-edited the final report of the gigabit networking research effort with Professor David Tennenhouse and Senior Research Scientist David Clark. David Clark is generally considered to be one of the fathers of the Internet Protocol and served as Chief Protocol Architect for the Internet. With its focus on networking, the group, including myself, set up and maintained the network and computer systems. These systems included the networking on the workstations and desktops, the distributed file system, desktops and workstations, setting up and

maintaining the distributed file system (Network File System) and the authentication system (Network Information Service, formerly known as Yellow Pages). Our system allowed users to log into any of the group's workstations using their username/password, which allowed that all of the user's files would be virtually mounted on that workstation as a networked home directory.

13.I defended and submitted my Ph.D. thesis, titled "Designing Networks for Tomorrow's Traffic," in January 1998. As part of my thesis research, I analyzed local-area and wide-area flows to show a more efficient method for routing packets in a network, based on traffic patterns at the time. My thesis also addressed real-time streamed audio and video. The network traffic that I analyzed was IP protocol traffic, including UDP and TCP.

14.I have been involved in Voice over Internet Protocol ("VoIP") technologies since 1997 and have specific experience in designing and testing VoIP networks.

15.From 1997 to 1999, I worked at NBX Corporation, which was acquired by 3Com Corporation in 1999. During this time, I was a Senior Scientist and Engineer working in IP Telephony. NBX delivered the world's first fully featured business telephone system to run over a data network, the

NBX100. NBX was one of the first business phone systems to be configurable via a web interface. Users and administrators had access to varying levels of configuration for the phone system.

16. As part of my work at NBX, I designed the core audio reconstruction algorithms for the telephones which depacketized the voice data and reconstructed the audio. In addition, I designed the voice data packet transmission algorithms. I created a system to capture and analyze network packets sent by devices in the NBX system for aid in testing and debugging. I also designed and validated the core packet transport protocol used by the phone system. In addition, I designed and oversaw the development of the underlying transport protocol used by the NBX100 phone system for reliable packet transport. That transport protocol is still used by the NBX100 system and its successors and is estimated to be used hundreds of millions of times daily. I wrote NBX's first demonstration IP software stack, which added the capability for utilizing the NBX100 phone system on an IP network. NBX first demonstrated a phone in the NBX100 system working over the Internet in 1998 at a trade show in Las Vegas. I was later the lead architect in designing NBX's next-generation highly scalable system, and, after NBX was acquired by 3Com, I did some work with

3Com's cable equipment division, including demonstrating a working NBX IP phone system over 3Com's cable equipment infrastructure using an early version of DOCSIS at a trade show in 1999. The NBX100 was the market's leading business phone system to run on a data network for several years following its introduction. During that time, I became more familiar with the various standards relevant to Internet telephony as well as the problems which designers of commercial telephony operations were faced with in implementing VoIP.

17.I, along with two of NBX's founders, was awarded U.S. Patent No. 6,697,963 titled "Telecommunication method for ensuring on-time delivery of packets containing time-sensitive data," for some of the work we did while at NBX.

18.After NBX, I worked at Teradyne, a test tool company primarily focused on semiconductors. Teradyne had recently acquired Hammer, a company that specialized in load and functional testing for telecommunications systems. The Hammer product is well known as a telecom test tool. Teradyne spun out Hammer and several other internal divisions into an independent company called Empirix. I became Chief Technologist of the Hammer

division of Empirix. Empirix was a leader in VoIP network testing and monitoring.

19. At Empirix, I laid out a new multi-year product vision for data network testing, secured internal funding for the effort, and led a team to deliver a new technology platform to the market in February 2001. This new product, PacketSphere, initially emulated network behavior so that wide-area VoIP connections could be tested in a lab. A later release allowed PacketSphere to generate high volumes of VoIP calls, including media streams, and to monitor the quality of VoIP voice streams. Later, the core technology was added to other Empirix products such as Empirix's Hammer XMS to monitor thousands of VoIP media streams in real time to determine their quality. PacketSphere was Empirix's most successful new platform introduction. Companies purchased the PacketSphere product to emulate an Internet Protocol network to see the effects of deploying their product on the Internet prior to launch. PacketSphere received several industry awards.

20. During my time at Empirix, I presented lectures on VoIP and data network testing to companies including Lucent Labs (formerly AT&T Bell Labs). I was also invited to present several guest lectures in a software engineering course at MIT. Since then, I have also participated twice as a unit lecturer

(two weeks) in an experimental course that was taught by an Institute Professor (the highest award that a MIT Professor can achieve) and sponsored by the Chairman of the MIT Corporation (MIT's board of trustees).

21. From 2004 to 2008, I was employed by BBN Technologies Corp., a technology research and development company located in Cambridge, Massachusetts. BBN Technologies is a world-renowned company with expertise in acoustics, speech recognition, and communications technology. BBN Technologies staff have pioneered many internetworking technologies and Internet applications, and built some of the world's largest government and commercial data networks.

22. My duties and responsibilities at BBN Technologies generally included commercialization of the technologies developed by BBN Technologies, which included spinning off companies and growing commercial businesses in-house. More particularly, I was involved in utilizing the award-winning AVOKE STX speech recognition technology to create the public audio/video search engine EveryZing (formerly known as PodZinger) which was spun out into a stand-alone company now known as RAMP, Inc. PodZinger won the 2006 MITX Technology Award for best Web 2.0

Application and was also named the 2006 Forbes Favorite Video & Audio Search Engine, beating out Google, Yahoo, and other companies. After managing the creation of the initial prototype system, PodZinger built out a full streaming audio and video search solution when I was the Vice President of Operations and Technology there. I was also involved in the Boomerang Mobile Shooter Detection project as the Vice President of Engineering for the program. The Boomerang system was deployed to Iraq and Afghanistan, and was credited with saving many lives.

23. From 1989 to 1990, I worked at AT&T Bell Laboratories on optical computers. This work generated six peer-reviewed papers, and multiple U.S. and European patent applications in which I was named as a co-author or inventor. I also interned at AT&T Bell Laboratories in 1987 and 1988. Additional relevant experience in the field of optical computers is listed in my *Curriculum Vitae*.

24. I am a named inventor on several patents and published patent applications that are related to the VoIP technology including: U.S. Patent No. 6,967,963, entitled “Telecommunication Method for Ensuring On-time Delivery of Packets Containing Time- Sensitive Data”; U.S. Patent Application Publication No. 20020015387, entitled “Voice Traffic Packet

Capture and Analysis Tool for a Data Network”; U.S. Patent Application Publication No. 20020016708, entitled “Method and Apparatus for Utilizing a Network Processor as Part of a Test System”; U.S. Patent Application Publication No. 20020016937, entitled “Method and Apparatus for Utilizing a Network Processor as Part of a Test System”; and U.S. Patent No. 7,590,542, entitled “Method of Generating Test Scripts Using a Voice-Capable Markup Language.”

25.[This paragraph left intentionally blank].

III. MATERIALS CONSIDERED FOR THIS DECLARATION

26.In addition to my general knowledge, education, and experience, I have considered all of the materials referenced in my declaration as well as the materials listed in **Exhibit B** in forming my opinions.

IV. SUMMARY OF OPINIONS

27.Based on my review of the ‘657 patent and its prosecution history (including that of the parent ‘899 patent), the other materials I have considered, and my knowledge and experience, my opinions are as follows. The prior art renders the claims obvious for the ‘657 patent as identified below.

28.In my analysis herein, I have considered the various disclosures (e.g., the ‘657 patent, the parent ‘899 patent, the prior art, ...) from the view of an person of

ordinary skill in the art (“POSITA”) at the time of the invention, which I have also referenced as the “priority date” herein.

29. I have presented my opinion below regarding the challenged claims based on the following grounds:

Ground	Claims	Basis	Reference
1	1, 5, 6, 8, 9, 15, 16	§ 103	Banerjee
2	1, 5, 6, 8, 9, 15, 16	§ 103	Rompaey alone or in combination with CoWare I

V. OVERVIEW OF THE ‘657 PATENT, ITS PATENT FAMILY, AND ITS PROSECUTION HISTORY

A. Overview of the ‘657 Patent.

30. The application that became the parent ‘899 patent was filed May 23, 2013 as non-provisional application 13/901,466. The application that became the ‘657 patent (14/584,897) asserts itself to be a continuation of the parent ‘899 patent’s application.

31. At a high level, the ‘657 patent is generally directed to “[a] system and method for providing an integrated circuit that integrates with and controls a device wherein the integrated circuit design is developed based on a

selection of characteristics of the device. The system and method also provide software for establishing interoperability between the integrated circuit and a controller.” EX-1001, Abstract. The patent purports to solve the problem of different remote controls for different devices as discussed in its Background:

For some time now, technology has advanced to the point where developers and manufacturers have had the ability to provide remote control over their products. A very simple example is a television and its remote controller. A remote controller uses a wireless signal, usually infrared, to send user provided control instructions to a television set. The television set then executes those instructions, such as turning off the television or changing the channel. In the case of the TV and the remote controller, manufacturers had to design, develop and program the remote controller such that it could interact with the proprietary specifications of their television system. A remote controller setup for one television system would not work for another television system or a different system all together, such as a DVD player.

With the wide proliferation of the devices connected to the internet, developers have had the ability to develop software and controllers to control devices connected to the internet. One example is a personal computer with remote desktop software. This software enables a personal computer (with the correct software) to control another personal computer which is connected to the internet and running the remote desktop software. Another example is a mechanical device which is fitted with integrated circuits (ICs) that can be controlled by smartphones such as AR.Drone, Sphero and iHelicopter. For situations like these, developers currently have to create or develop proprietary software to provide control functionality of proprietary hardware.

...

All currently available automation, integration and control products and technologies are nothing more than hardware and software acting as figurative ropes and pulleys designed to manipulate devices and constructs with abstraction for the actual state of the device itself. Consequently, these existing solutions require considerable customization and development on a case by case basis for the

establishment of individual/closed automation and control ecosystems. **These solutions are neither universal** nor capable of ad-hoc deployment scenarios. From complex proprietary based control and automation systems to low cost, Do-It-Yourself (DIY) approaches, all currently available solutions require varying degrees of hardware and software abstraction to accomplish their respective deployment goals.

In general, all competing solutions fail to offer universal control as they all are predicated upon deployment through an abstraction layer and are geared towards specific markets of which there is no cross-communication or service. For the most part, competing solutions to the above described problems introduce automation of devices as “after thoughts” and attempt to enable integration through a “bolt on” metaphor. No competing technology has yet produced a universal “plug and play” model for any third party device manufacture to employ.

...

True, universal automation and control is predicated upon the development of the integrated technology piece which will deliver

plug and play for all devices and constructs that can be envisioned within an automated ecosystem. Historically, we have accepted that information and its movement through the internet have been viewed as unrestricted through the establishment of communication, messaging and interaction standards such as TCP, SMTP, and HTTP and we have viewed access to the same as commonplace due to the ubiquity of the web browser. These standards and related technologies not only assure access to information but are largely taken for granted as being available anywhere and through anything at any time.

EX-1001, 1:16-2:58.

32. The “Summary of The Invention” states: “In its most general aspect, the invention includes a system and method **for providing universal and integrated control** to constructs and devices which may be controlled by micro control units (MCU) or other controlling hardware. In another general aspect, **the invention provides a mechanism and methodology for delivering true universal automation and control of any device** or construct that can be imagined within an automation and control ecosystem.” EX-1001, 3:45-58.

33.The ‘657 Title is “System and method for universal control of electronic devices.” The Challenged Claims depart from the expressed solution of a “universal control.”

34.The Challenged Claims depart from the purported problem and solution and do not expressly recite the “controller” found in, for example, claims 2, 3, and 4.

35.As issued, the ‘657 patent includes 16 claims. I have been asked to opine on the Challenged Claims in my analysis, which I have included below:

1. A method for designing an integrated circuit which controls a device when the integrated circuit is integrated into the device comprising:

selecting a value for a control type parameter associated with an electronic device from a hierarchal classification database;

selecting a value for a display type parameter associated with the electronic device from the hierarchal classification database;

selecting a value for an input type parameter for the
electronic device from the hierarchal classification
database;

selecting a value for an output type parameter for the
electronic device from the hierarchal classification
database;

receiving the selected values for the control type, display
type, input type and output type parameters into a
memory of a processor programmed by software
commands to design a hardware integrated circuit and
associated operating software for controlling the
hardware integrated circuit to control the electronic
device;

analyzing by the processor the selected values for the
control type, display type, input type and output type
parameters of the electronic device;

outputting by the processor a hardware integrated circuit
design based on the selection of input and output
characteristics of the electronic device, the hardware

integrated circuit including a communication bus for communicating with the electronic device; and outputting by the processor software programming commands to be .executed by the hardware integrated circuit for controlling the electronic device *in accordance* with the selection of categorized input and output characteristics of the electronic device through the communication bus of the hardware integrated circuit using a communication protocol compatible with the electronic device and the hardware integrated circuit.

5. The method of claim 1, wherein the hardware integrated circuit design is based on at least one predetermined hardware integrated circuit component which is able to control one of the selected values for the control type, display type, input type and output type parameters.

6. The method of claim 5, wherein the hardware integrated circuit design is provided as a software schematic, layout, or both.

8. The method of claim 1, wherein the hierarchal classification database is a component of a fabless development kit.

9. A system comprising:

one or more processors;

a design user interface provided by programming commands

executed by the one or more processors, the design user

interface allowing a user to characterize an electronic device by

selecting a value for a control type parameter, a value for a

display type parameter, a value for an input type parameter and

a value for an output type parameter, all of which are associated

with the operation and control of the electronic device;

logic encoded in one or more non-transitory computer-readable

media that, when executed by the one or more processors, is

operable to:

receive the selected control type, display type, input type

and output type parameter values selected by the user

into a memory associated with the one or more

processors, the logic executed by the one or more

processors operable to design a hardware integrated

circuit design based on the selected control type, display

type, input type and output type parameter values

selected by the user, the hardware integrated circuit design including a communication bus; and output software programming commands to be executed by the hardware integrated circuit for controlling the device in accordance with the selected control type, display type, input type and output type parameter values selected by the user using a communication protocol compatible with the device and the hardware integrated circuit through the communication bus of the hardware integrated circuit.

15. The system of claim 9, wherein the one or more non-transitory computer-readable media further comprises logic that, when executed by the one or more processors, is further operable to simulate the hardware integrated circuit's functionality based on the hardware integrated circuit design.

16. The system of claim 9, wherein the design user interface is in communication with a hierarchal classification database, and the user selects the selected control type, display type, input type and output type parameter values from the hierarchal classification database.

**B. Overview of the Prosecution History of the ‘657 Patent and
Related Applications**

- i. Prosecution History of the application leading to the 657 Patent
(application 14/584,897)

36.I have reviewed the prosecution history of the ‘657 patent including application 14/584,897 as well as the parent ‘899 patent application (13/901,466) (or the “‘466 Application”) in forming my opinions for purposes of this declaration.

37.As shown in my declaration for IPR2024-00467, the ‘466 Application prosecution confirms that the parent ‘899 claims are neither novel nor non-obvious. The ‘657 prosecution history shows that the ‘657 claims were not patentably distinct from the parent ‘899 claims. Thus, the ‘657 claims are neither novel nor non-obvious.

38.In the ‘897 Application, the applicant submitted claim 1 which recited the following:

1. A computer implemented method for designing an IC which controls a device
when integrated into the device comprising:

receiving a selection of categorized input and output characteristics of the
electronic device;

providing an IC design based on the selection of input and output characteristics
of the device; and

providing software for controlling the device through an IC based on the IC
design.

This claim was immediately amended to recite:

Declaration of Henry Houh, Ph.D. Under 37 C.F.R. § 1.68 in Support of
Petition for *Inter Partes* Review of U.S. Patent Nos. 9,367,657

Claims 1-20 are canceled.

21. (New) A method for designing an integrated circuit which controls a device when the integrated circuit is integrated into the device comprising:

selecting a value for a control type parameter associated with an electronic device from a hierarchal classification database;

selecting a value for a display type parameter associated with the electronic device from the hierarchal classification database;

selecting a value for an input type parameter for the electronic device from the hierarchal classification database;

selecting a value for an output type parameter for the electronic device from the hierarchal classification database;

receiving the selected values for the control type, display type, input type and output type parameters into a memory of a processor programmed by software commands to design a hardware integrated circuit and associated operating software for controlling the hardware integrated circuit to control the electronic device;

analyzing by the processor the selected values for the control type, display type, input type and output type parameters of the electronic device;

outputting by the processor a hardware integrated circuit design based on the selection of input and output characteristics of the electronic device, the hardware integrated circuit including a communication bus for communicating with the electronic device; and

outputting by the processor software programming commands to be executed by the hardware integrated circuit for controlling the electronic device in accordance with the selection of categorized input and output characteristics of the electronic device through the communication bus of the hardware integrated circuit using a communication protocol compatible with the electronic device.

39. See '897 Application, December 29, 2014 Claims and Preliminary Amendment.

40. The Examiner issued a double patenting rejection of these claims. See '897 Application, Rejection, July 27, 2015.

41. The Examiner determined the '897 application claims were not patentably distinct from the parent '899 patent claims.

Although the conflicting claims are not identical, they are not patentably distinct from each other because the subject matter in the instant Application are similar to the subject matter in copending Application and substantially read on the claims in the copending Application and vice versa, the claims in the copending Application is similar to the claims in the instant Application and read on the claims in the instant Application.

42. The applicant filed a terminal disclaimer and made a minor amendment to the last claim element of claim 1:

selection of categorized input and output characteristics of the electronic device through the communication bus of the hardware integrated circuit using a communication protocol compatible with the electronic device and the hardware integrated circuit.

2015-11-25 Amendment at 1-3.

43. The examiner subsequently allowed the claims.

44. This paragraph left intentionally blank

45. This paragraph left intentionally blank.

VI. LEVEL OF ORDINARY SKILL IN THE PERTINENT ART

46.I have been advised that there are multiple factors relevant to determining the level of ordinary skill in the pertinent art, including the educational level of active workers in the field at the time of the invention, the sophistication of the technology, the type of problems encountered in the art, and the prior art solutions to those problems.

47.It is my opinion that a person having ordinary skill in the relevant art at the time of invention would have a Bachelor's or Master's Degree in computer science, electrical engineering, or a related field, with at least three years of practical experience in circuit design, digital design, and/or automation tools. Additional experience could substitute for the advanced degree.

VII. CLAIM CONSTRUCTION

48.I understand that both parties have submitted claim construction briefing to the Court. I have considered both parties' constructions in my analysis below.

49.The prior art discussed below discloses the claim elements under any reasonable construction of the terms and under either parties' proposed construction.

VIII. UNDERSTANDING OF THE LAW

50. In expressing my opinions and considering the subject matter of the claims of the '657 patent, I am relying upon certain basic legal principles that have been explained to me. I have been informed by counsel that a patent claim is unpatentable as anticipated under 35 U.S.C. § 102 if each element of that claim is present either expressly or inherently in a single prior art reference. I have also been informed that, to be an inherent disclosure, the prior art reference must necessarily disclose the limitation, and the fact that the reference might possibly practice or contain a claimed limitation is insufficient to establish that the reference inherently teaches the limitation.
51. I have been informed that a claimed invention is unpatentable under 35 U.S.C. § 103, if the differences between the invention and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which the subject matter pertains. I have also been informed by counsel that the obviousness analysis takes into account factual inquiries including the level of ordinary skill in the art, the scope and content of the prior art, and the differences between the prior art and the claimed subject matter.

52.I have been informed by counsel that the Supreme Court has recognized several rationales for combining references or modifying a reference to show obviousness of claimed subject matter. Some of these rationales include the following: (a) combining prior art elements according to known methods to yield predictable results; (b) simple substitution of one known element for another to obtain predictable results; (c) use of a known technique to improve a similar device (method, or product) in the same way; (d) applying a known technique to a known device (method, or product) ready for improvement to yield predictable results; (e) choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success; and (f) some teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention.

53.Finally, I understand that secondary factors may be considered in an obviousness inquiry. I understand that these are also referred to as secondary considerations and may include evidence of long-felt need, failure of others, skepticism of those of skill in the art, licensing, commercial success, recognition in the industry, acquiescence, evidence of

copying, and unexpected results. While these secondary considerations must be taken into account, they do not necessarily control the obviousness determination.

54. Moreover, for objective evidence of nonobviousness to be accorded substantial weight, I understand that the proponent of such evidence must establish a “nexus” between the evidence and the merits of the patent at issue. Specifically, I understand that the secondary consideration must be tied to the merits of the claimed invention of the patent at issue. Where the alleged secondary consideration results from something other than what is both claimed and novel, I understand that there is no nexus to the merits of the claimed invention and the secondary considerations are thus not indicative of nonobviousness. I also understand that it is the patentee’s burden to come forward with evidence that a nexus exists.

55. [This paragraph left intentionally blank].

IX. SCOPE AND CONTENT OF THE PRIOR ART

A. Rompaey

56. Rompaey was filed on October 23, 1996 and issued on February 9, 1999.

Rompaey is prior art under 35 U.S.C. § 102(a) (post-AIA). Rompaey is attached as Exhibit 1004 to the IPR.

57. I have been advised that, because the earliest filing date related to the ‘657 patent is after March 16, 2013, the “first-to-file” provisions of the American Invents Act (“AIA”) apply. Thus, the ‘657 priority date is the date of the first application that provides Section 112 support for the claims.

58. Rompaey is prior art under at least the pre-AIA version of 35 U.S.C. § 102(a), (b), (e), and (g) and the post-AIA version of 35 U.S.C. § 102(a)(1), (a)(2) and (d).

59. Rompaey teaches a “Design environment and a design method for hardware/software co-design.” Rompaey at Title. Like the ‘657 patent Rompaey teaches that designers may specify the functional characteristics of devices including the inputs and outputs and provides a tool (called CoWare) to automatically generate hardware integrated circuits and software to implement the designer’s selections and specifications:

A hardware and software co-design environment and design methodology based on a data-model that allows one to specify, simulate, and synthesize heterogeneous hardware and software architectures from a heterogeneous specification. The environment and methodology of the invention allow for the interactive synthesis of hardware and software interfaces. The environment defines

primitive objects to represent a specification of an essentially digital system. The primitive objects are defined by describing the specification of the system in one or more processes, each process representing a functional aspect of the system. Further, each of the processes have ports which are connected to ports of other processes with a channel. The ports structure communication between the processes.

Rompaey at Abstract.

60.Rompaey teaches a design tool system (“CoWare”). Using CoWare, a system designer inputs a device specification, defines and specifies inputs of “models” from libraries that define hardware and software pre-existing components, refines those components interactively, converts the design into hardware and software, and then outputs the integrated circuit design (using formats such as VHDL and net lists) and the associated software that executes on the integrated circuit. Rompaey teaches a specific example of designing the circuits for a “pager” device.

61.Rompaey’s pager example renders the Challenged Claims unpatentable. As detailed below, Fig. 10 depicts a system diagram of the components within a pager. EX-1004, 21:33-22:35.

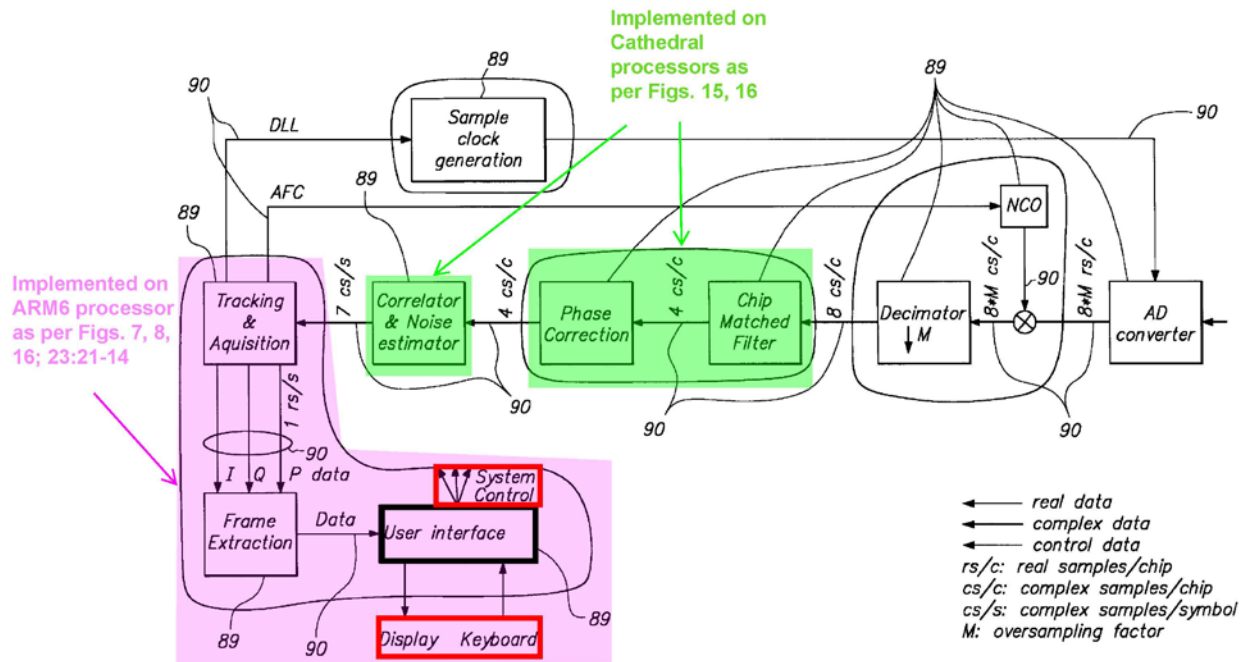
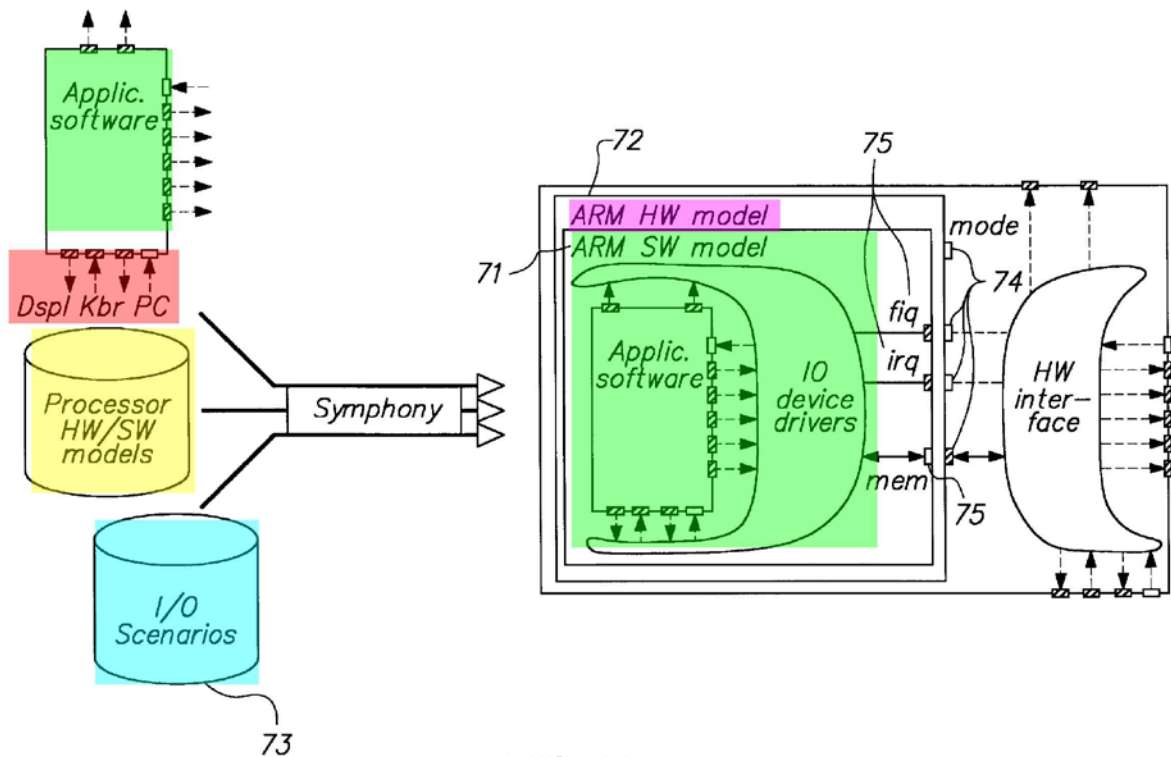


FIG. 10

62.

63.EX-1004, Fig. 10 (annotated). The “Tracking Acquisition [], Frame Extraction [], and User Interface are ... mapped on ... an ARM6 processor” (EX-1004, 23:21-23) as shown in Fig. 7:



64.

FIG. 7

65.EX-1004, Fig. 7 (annotated).

66.In Fig. 7, a user previously specified the overall system design to CoWare.

including components that perform input and outputs such as the display (“**Dspl**”), keyboard (“**Kbr**”) and program controls (“**PC**”). EX-1004, Figs 9-14. CoWare uses the system design, along with “**Processor HW/SW models**” and “**I/O scenarios**” to generate the “**ARM HW model**,” and the “**ARM SW/Model**” (which includes **application software** and **IO device drivers**). EX-1004, 26:42-43. The “user interface” (Fig. 10) controls device inputs / outputs such as the keyboard, display and program controls.

In CoWare, the “SYMPHONY [function], adds RAM and ROM blocks to store the program code and data.” EX-1004, 20:22-23.

67. After the user has refined and simulated the system design, the design is synthesized into VHDL or a “net list describing the final implementation of the system” (for the hardware). EX-1004, 28:29-30. For software, “[t]he I/O device driver processes are merged with the original software application code.” EX-1004, 26:42-43. CoWare then outputs the hardware and software designs.

68. Rompaey discloses, or renders obvious, the Challenged Claims.

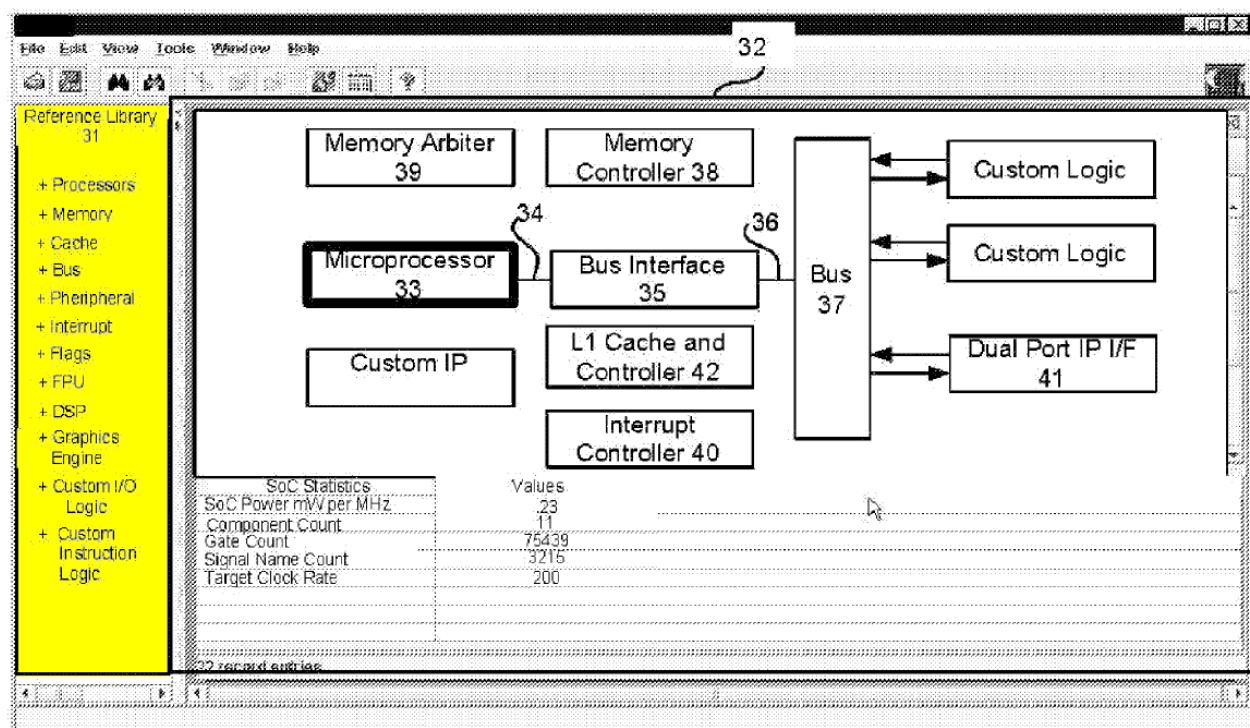
B. Banerjee

69. Banerjee was filed on March 9, 2007, and published on September 11, 2008.

70. Banerjee discloses, or renders obvious, every element of the Challenged Claims. Banerjee teaches a software tool for designing hardware integrated circuits and software that executes on the hardware. The designer enters “parameters,” “characteristics,” and “attributes” to tailor the designed hardware and software for a specific “application” (functions of the device controlled by the hardware circuit and software) such as devices that require

“real-time processing of video or audio” or “media players” or “signal processing,” “embedded applications,” or “custom I/O logic.”

71. In one example, a “designer” uses a GUI with selectable categories – each with their own sub-menus of selectable components and parameters thereof.



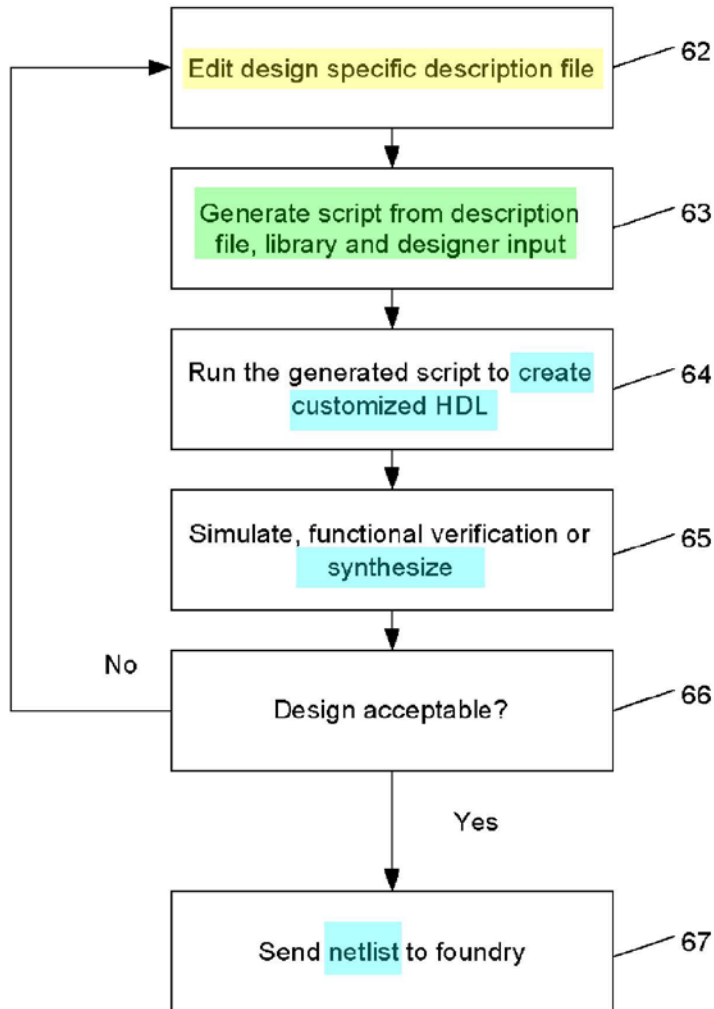
72. EX-1005, Fig. 3.

73. The high-level hardware circuit design process is exemplified in Figure 6.

The Fig. 3 GUI is used to “**edit design specification description file.**”

Then, Banerjee’s software tool receives characteristic “**from description file, library, and designer input,**” and then “**creates customized HDL**”

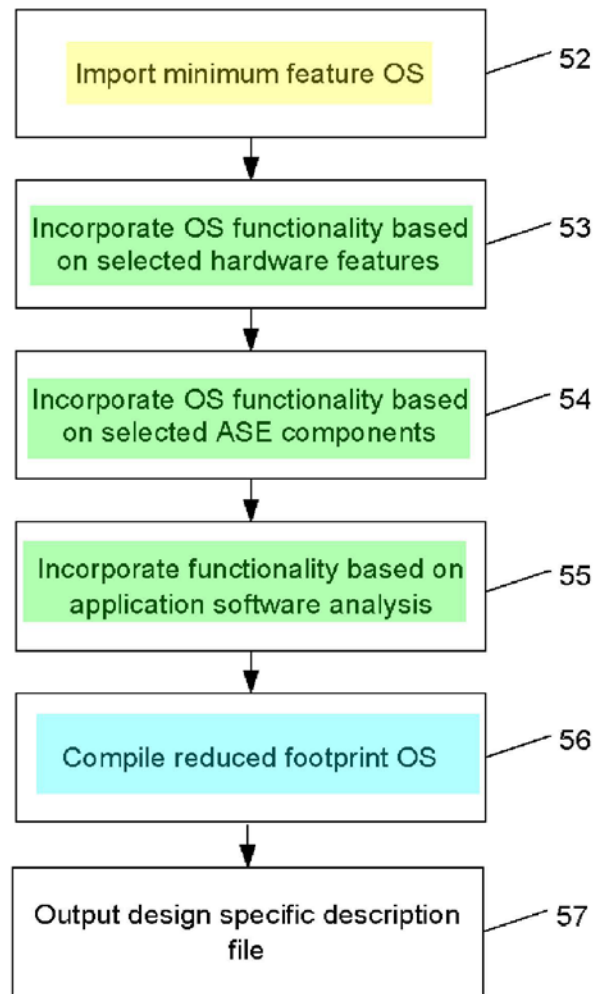
which can be “**synthesized**” into a “**netlist**” (which shows the hardware integrated circuit design).



74.EX-1005, Fig. 6 (annotated).

75. Fig. 5 exemplifies how one software portion (the “operating system” / “OS”) is designed and output. Banerjee’s tool starts with a “**minimum feature OS**” from a library. The tool analyzes the system “hardware

features” to “**incorporate**” additional “**OS functionality.**” Then, “the optimized operating system is **compiled into a reduced footprint operating system** that is instantiated in the device, preferably in a non-volatile computer readable storage medium.” EX-1005, 0083. Similar processes occur for the “application software” and “middleware” that Banerjee’s tool outputs. EX-1005, ¶¶0072, 0078.



76.EX-1005, Fig. 5 (annotated).

77. [This paragraph left intentionally blank]

78. Banerjee discloses every aspect of the Challenged Claims.

C. Background Prior Art

79. A number of additional other references demonstrate that all of the aspects of the claimed invention were well-known to a POSITA long prior to the filing of the ‘657 patent.

1. Sugimoto

80. U.S. Pat. No. 5,197,016 (“Sugimoto”) was filed in 1989 – 24 years before the ‘657 patent – and issued on March 23, 1993. Sugimoto teaches “[a] computer-aided system and method [] for designing an application specific integrated circuit (ASIC) whose intended function is implemented both by a hardware subsystem including hardware elements on the integrated circuit and by a software subsystem including a general purpose microprocessor also on the integrated circuit.” EX-1009, Abstract. Sugimoto’s Fig. 2 depicts the system called “ISCC” which corresponds with Sugimoto’s Title of “Integrated Silicon-Software Compiler.”

81. ISSC was a tool for designing an ASIC as shown in Figure 2 and described at

The overall system flow in accordance with the present invention is illustrated in FIG. 1. The computer-aided system and method of the

present invention is represented by the block 10, and for simplicity is referred to herein by the acronym ISSC (integrated silicon-software compiler). The ISSC 10 receives as its input (which are technology and architecture independent) specifications 11, and which define on a functional or behavioral level the functions which are to be performed by the target application specific integrated circuit (ASIC). As illustrated in FIG. 1, the user enters the functional specifications of the desired target ASIC into the integrated silicon-software compiler (ISSC) 10 in the form of a flowchart 11. The ISSC 10 then generates a netlist 15 from the flowchart. The netlist 15 includes architecture specific definitions of the hardware cells required to implement the functions of the hardware subsystem, a definition of the microprocessor used to implement the functions of the software subsystem, and definitions of a system controller, data paths and control paths for interconnecting the various components. The netlist 15 can be used as input to any existing VLSI layout and routing tool 16 to create mask data 18 for geometrical layout. The ISSC system 10 also generates a program 19 from the flowchart information for implementing the functions of the software subsystem. The program

19 can be stored in an external or internal memory associated with the microprocessor on the target ASIC.

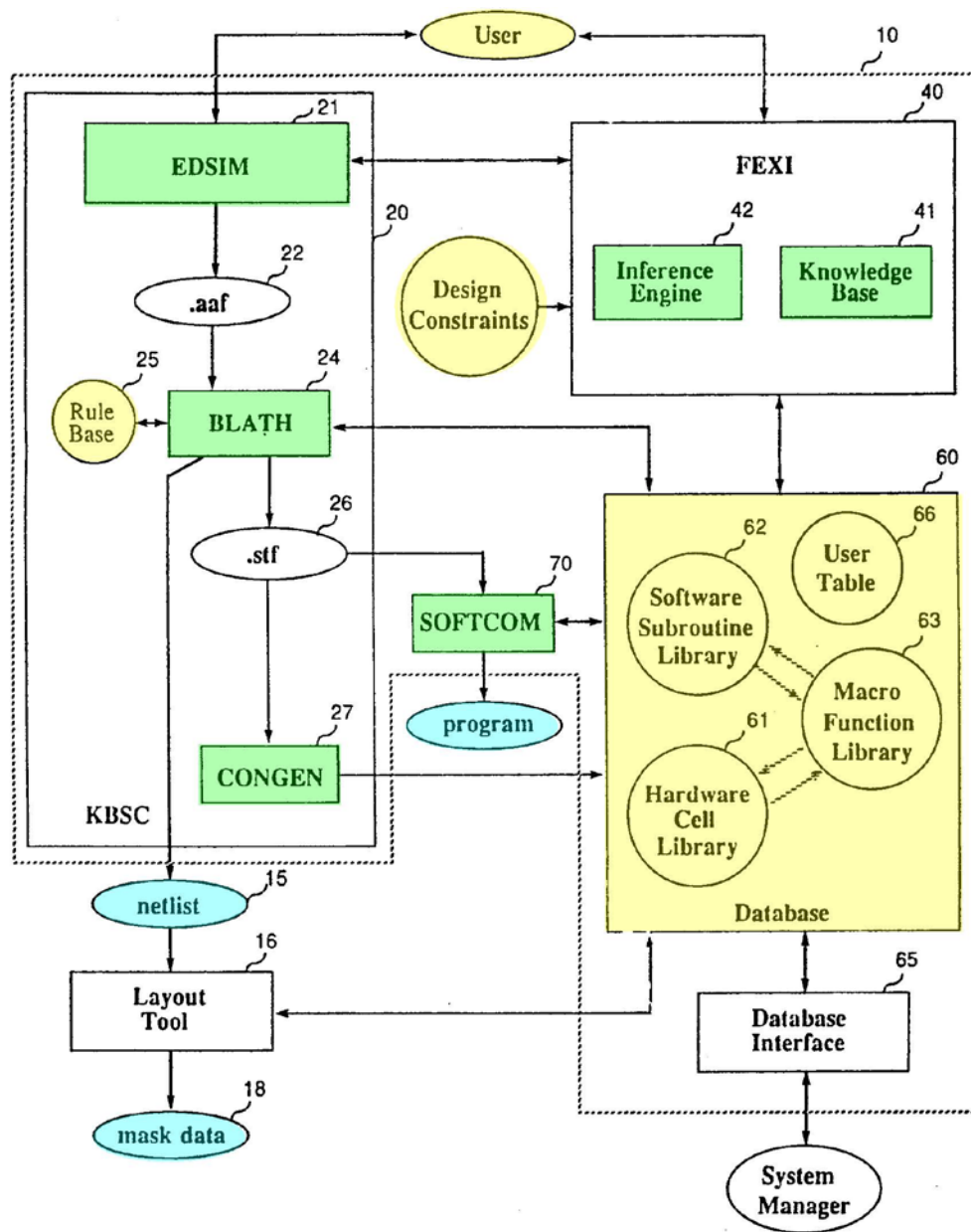
The ISSC system 10 can be operated on a suitable programmed general purpose digital computer. By way of example, one embodiment of the system is operated in a workstation environment such as Sun3 and VAXStation-II/GPX running UNIX Operating System and X Window Manager. The software uses C programming language and a data base such as INGRES or Gbase. The human interface is mainly done by the use of a pointing device, such as a mouse, with pop up menus, buttons, and a special purpose command language. The permanent data of the integrated circuit design are stored in a database for easy retrieval and update. Main memory temporarily stores the executable code, design data (flowchart, logic, etc.), data base (cell library), and knowledge base. The CPU performs the main tasks of creating and simulating flowcharts and the automatic synthesis of the design.

The primary elements or modules which comprise the ISSC system are shown in FIG. 2. In the embodiment illustrated and described herein, these elements or modules are in the form of software

programs, although persons skilled in the appropriate art will recognize that these elements can be easily embodied in other forms, such as in hardware for example.

Referring more particularly to FIG. 2, it will be seen that the ISSC system 10 includes a module or subsystem 20 called KBSC (Knowledge Based Silicon Compiler). The KBSC subsystem 20 corresponds to the system and method described in copending U.S. application Ser. No. 143,821, filed Jan. 13, 1988, the subject matter of which is incorporated herein by reference. As shown in FIG. 2, the KBSC subsystem 20 includes a number of modules or programs which collectively provide an interface with the user for receiving input of the functional specifications for the particular target ASIC and which provide as output a netlist 15 and a program 19. Other major subsystems or modules of the ISSC 10 include a front-end expert system interface (FEXI) 40, a database 60 and a software compiler (SOFTCOM) 70.

82.EX-1009, 3:58-4:58.



EX-1009, Fig. 2 (annotated).

83. Sugimoto's ISSC tool receives inputs from **users**, **rule bases**, **design constraints**, and **libraries of software and hardware components**. ISSC performs the **analysis** to merge the components into a device system design. ISSC

then outputs the hardware design (“**netlist**” and “**mask data**”) and the software (“**program**”) to control the device using the designed ASIC. EX-1009, 3:58-5:30.

2. **Kobayashi**

84. Sugimoto is a continuation-in-part of U.S. Patent No. 4,922,432 (“Kobayashi”) filed in 1988 which issued on May 1, 1990. Kobayashi has similar content to Sugimoto and also presents its ASIC design in the context of a specific controlled “device”: “soft drink vending machine controller design example.”

Soft Drink Vending Machine Controller Design Example

The following example illustrates how the previously described features of the present invention are employed in the design of an application specific integrated circuit (ASIC). In this illustrative example the ASIC is designed for use as a vending machine controller. The vending machine controller receives a signal each time a coin has been deposited in a coin receiver. The coin value is recorded and when coins totalling the correct amount are received, the controller generates a signal to dispense a soft drink. When coins totalling more than the cost of the soft drink are received, the controller dispenses change in the correct amount.

This vending machine controller example is patterned after a textbook example used in teaching digital system controller design. See Fletcher, William I., *An Engineering [sic] Approach to Digital Design*, Prentice-Hall, Inc., pp. 491-505. Reference may be made to this textbook example for a more complete explanation of this vending machine controller requirements, and for an understanding and appreciation of the complex design procedures prior to the present invention for designing the hardware components for a controller.

FIG. 10 illustrates a flowchart for the vending machine controller system. This flowchart would be entered into the KBSC system by the user through the flowchart editor. Briefly reviewing the flowchart, the controller receives a coin present signal when a coin is received in the coin receiver. State0 and cond0 define a waiting state awaiting deposit of a coin. The symbol CP represents "coin present" and the symbol -CP represents "coin not present". State1 and cond1 determine when the coin has cleared the coin receiver. At state20, after receipt of a coin, the macro instruction ADD3.1 (lc, cv, sum) instructs the system to add lc (last coin) and cv (coin value) and store the result as sum. The macro instruction associated with state21 moves the value in the

register sum to cv. The macro CMP.1 at state22 compares the value of cv with PR (price of soft drink) and returns signals EQ, GT and LT.

The condition cond2 tests the result of the compare operation CMP.1.

If the result is "not greater than" (-GT.CMP.1), then the condition

cond3 tests to see whether the result is "equal" (EQ.CMP.1). If the

result is "not equal" (-EQ.CMP.1), then control is returned to state0

awaiting the deposit of another coin. If cond3 is EQ, then state4

generates a control signal to dispense a soft drink (droppop) and the

macro instruction CLR.1(cv) resets cv to zero awaiting another

customer.

If the total coins deposited exceed the price, then state30 produces the

action "returncoin". Additionally, the macro DECR.1 (cv) reduces the

value of cv by the amount of the returned coin. At state31 cv and PR

are again compared. If cv is still greater than PR, then control passes

to state30 for return of another coin. The condition cond5 tests

whether the result of CMP.2 is EQ and will result in either dispensing

a drink (droppop) true or branching to state0 awaiting deposit of

another coin. The macros associated with the states shown in FIG. 10

correspond to those defined in Table 1 above and define the particular actions which are to be performed at the respective states.

Appendix A shows the intermediate file or "statelist" produced from the flowchart of FIG. 10. This statelist is produced as output from the EDSIM program 20 and is used as input to the PSCS program 30 (FIG. 3).

85.EX-1010, 12:37-14:30. See also Figs. 10-15 explaining the soft drink vending machine example.

86.As noted in the quote above, Kobayashi's "vending machine controller example is patterned after a textbook example used in teaching digital system controller design. See Fletcher, William I., *An Engineering Approach to Digital Design*, Prentice-Hall, Inc., pp. 491-505." EX-1010, 12:51-55 ("Fletcher").

3. **Fletcher**

87.Fletcher was published in 1984 and is prior art under section 102(a). Fletcher contains detailed teachings regarding the use of tools to provide automatic design of the characteristics and functionality of devices and then to implement the hardware integrated circuits and software that implement such functionality.

88. The Fletcher passages referenced by Kobayashi date from 1984, when Fletcher was published. EX-1011.

89. Tools to design hardware integrated circuits and software to control devices existed decades prior to the ‘657 patent. A multitude of references show such systems. Thus, by May 2013, a POSA would have known that tools existed for electronic device system designers to select from categories of functional components to be used for the device, such as control, display, input and output components, to specify those components’ parameters, and to have the tools generate and output the hardware integrated circuits and software to control the desired electronic device.

90. [This paragraph left intentionally blank].

X. DETAILED UNPATENTABILITY ANALYSIS FOR THE ‘657

91. For the ‘657, I have been asked to provide an opinion as to whether the Challenged Claims are obvious over the Rompaey (alone or in combination with CoWare I) and Banerjee.

92. As part of my obviousness analysis, I have considered the scope and content of the prior art, described above and below. Additionally, I considered whether any differences between the alleged invention and the prior art are such that the subject matter, as a whole, would have been obvious to a person having ordinary skill in the art at the time of the alleged

invention. I have also considered the level of ordinary skill in the pertinent art in performing my analyses.

93.I noted that Ground 1 and Ground 2 are reversed below from the ordering that was presented in the IPR and my declaration regarding the ‘899 patent.

This is to correspond to the Petition for the ‘657 patent.

A. Ground 2: The Challenged Claims Are Obvious In View of Rompaey alone or in combination with CoWare I.

94.As detailed below, Rompaey alone, or in combination with CoWare I, discloses, or renders obvious, the Challenged Claims and thus renders these claims unpatentable.

95.Rompaey provides extensive details regarding a system a designer uses the CoWare tool to design a “system” such as the exemplary pager device by selecting various functionality parameters for the system to have (including display, input and output parameters such as buttons, displays, and keyboards) and refine that system design specification by continuing to select various parameters, attributes and characteristics (the claimed parameters) from hierarchal classifications databases including I/O scenarios defining the inputs and outputs and their associated functionality and control circuitry. The designer then uses CoWare to analyze the selections specifying the design, generate hardware integrated circuit

designs (including the use of predefined existing components implement various functionality specifications and parameters) and then generate software to execute on the hardware design. The CoWare tool with then simulate the designs (as desired) and output the hardware integrated circuit designs and software.

1. **Rompaey Discloses, And/ Or Renders Obvious, The Challenged Claims**

a) **Claim 1**

Preamble: “A method for designing an integrated circuit which controls a device when the integrated circuit is integrated into the device comprising:”

96. Whether or not the preamble is limiting, Rompaey discloses this element.

Using the CoWare tool in Rompaey the system designer will design integrated circuit(s) using functional specifications and various libraries with the designed circuit(s) ultimately controlling a device into which it is incorporated. Rompaey teaches the exemplary design of the circuits and software to be incorporated in, and control, a “pager” device. *E.g.*, EX-1004 at 6:6-18; 8:4-48; 10:53-62; Figs. 1-2, 7-13.

97. For example, Rompaey’s claim 6, which depends on claim 1, teaches such a system:

1. A database compiled on a computer environment, adapted for access by executable programs on said environment for generating the implementation of an heterogeneous essentially digital system, comprising:

a plurality of objects representing aspects of said digital system wherein said objects comprise primitive objects representing the specification of said digital system and hierarchical objects being created by said executable programs while generating the implementation of said digital system, said primitive objects comprising ports, protocols, and channels, at least one primitive port having at least one primitive protocol, said hierarchical objects being successive refinements in a chain of refinements of said primitive objects, having more detail and preserving any one or all of said aspects to thereby generate said implementation of said digital system;

relations in between said primitive objects and in between said hierarchical objects and between said primitive objects and said hierarchical objects; and

functions for manipulating said objects and said relations.

6. The database as recited in claim 1 wherein said implementation is a heterogeneous implementation comprising hardware subsystems and software subsystems, said software subsystems being executed on one or more of said hardware subsystems.

EX-1004, claim 6.

98. As detailed above, the “implementation” is hardware systems, which correspond to the claimed IC, and software systems executed on the hardware. Claim 1 recites designing the “implementation of a heterogeneous essentially digital system” which Rompaey identifies as a pager. EX-1004, 10:16-17 (“FIG. 10 is a block diagram of a typical heterogeneous digital system: a pager application.”).

99. If the Patent Owner argues that Rompaey does not expressly or inherently disclose any aspect of this claim, the discussion of the elements below explains my opinions on why any such disputed aspect is obvious to a POSITA at the time of the invention.

- i. 1[a] “selecting a value for a control type parameter associated with an electronic device from a hierarchal classification database;

selecting a value for a display type parameter associated with an electronic device from a hierarchal classification database

selecting a value for an input type parameter for the electronic device from the hierarchal classification database;

selecting a value for an output type parameter for the electronic device from the hierarchal classification database;

receiving the selected values for the control type, display type, input type and output type parameters into a memory of a processor programmed by software commands to design a hardware integrated circuit and associated operating software for controlling the hardware integrated circuit to control the electronic device”

100. Rompaey discloses this element. Rompaey describes three broad examples of these “selecting” and “receiving” elements: (1) the designer selecting, and the CoWare tool receiving the initial specification / selection of components from the designer; (2) selections from libraries and repositories made by the designer and the CoWare tool that are received into CoWare with the selections including the hardware and software predefined modules or elements that correspond to the implementations of the designer system design; and (3) receiving from the additional selections by the designer and the CoWare tool of component and the characteristics of such components through the process of refining and optimizing the design.

(a) Rompaey’s Selection Actions From A Hierarchal
Classification Database

101. Rompaey’s designer selects and specifies the functionality of the designed system (a device such as a pager) and the components (hardware

and software) implementing that functionality for that specific device. In each of the three selections, the designer is selecting parameters of the functionality (including the control, display, inputs and outputs) of the designed “system” such as the disclosed pager device. The designer and CoWare make those selections from the various hierarchal classification databases described in Rompaey.

102. As I have noted in the Banerjee discussion below, the ‘657 patent uses the terms “function,” “parameters,” and “characteristics” largely interchangeably with respect to the control type, display type, input type, and output type selections. That same discussion applies to the Rompaey’s disclosure.

103. The ‘657 patent describes selecting the claimed parameters by a designer selecting that a particular device will have a desired functionality of a particular category. For example, the ‘657 describes selecting a functionality of the device:

In one aspect, the invention includes a system and method for designing integrated circuits (IC) to integrate into an electronic device for control using IC components that are developed for generic characteristics of electronic devices. This way, IC devices components may be standardized and designed based on functionality thus removing the necessity of designing a proprietary device in house. A

developer can select the functionality of their device and an IC can be designed off the standard components.

104. EX-1001, 4:46-56.

105. In the ‘657 patent, the categorized types of functions are stored in libraries such as the “classification library” described in the specification:

Diving further into the classification library provides a category describing the functions of the blender, **which functions may be further categorized by control type, display type, input type, and output type**. For example, a blender may be controlled by a rotary dial, with a display that amounts to a word or graphic or symbol that displays the state of the blender when the input type is actuated. In another example, the input device may be push buttons or touch sensitive elements, and the display type may be a LCD or other type of electronic display that informs the user what state the blender is in when an input type is actuated. The output of the input may also be discerned from the display type, as well as experientially, such as by hearing the increase or decrease of sound generated by the blender as the speed of the motor is increased or decreased, pulsed, or stopped upon actuation of the input type. As one skilled in the art will immediately understand, any device, such as a blender, can be defined by defining its control type, display type, input type, and output type.

106. EX-1001, 9:11-9:29.

107. The '657 patent teaches that the claimed parameters (referenced as “characteristics”) correspond to these “function types” (or the “functionality of the device”) that can be selected as per the quoted passages immediately above.

Referring again to FIGS. 2-4, additional details of an embodiment of a method using the FDK to design an ASIC in accordance with the principles of the invention when executed by a computer is described. As described previously, the FDK may provide a menu of categorized devices for a user to select from. In one embodiment, the menu may start with a limited number of broad categories of devices which drills down into a more specified category of a device with each selection until a final device is selected. By way of example and not limitation, a menu may have, for example, the classes: appliance, automotive, environment. Selecting the class appliance, may provide the additional categories: kitchen appliance, office appliance, entertainment appliance. Once selecting kitchen appliance, a menu of kitchen appliances may be provided such as blender, coffee machine, stove, etc. Once the user selects a device, **the user may be provided with a selectable set of characteristics of the user's device. By way of example, the type of characteristics may include control, display, input and output, communications.**

Characteristics may include multiple types within each characteristic. For example, types of control may include wheel, joystick, button, knob, switch, etc. Display type references the form of visual

communication, such as for example, alpha numeric, base ASCII, extended ASCII, bitmap, or even more complex displays such as HTML decoding engines. These display types may be communicated to a user of the device by LCD, LED, OLED, and the like. In other embodiments input and output of the device may be audio, video, etc. For example, the **blender's display references an “output” of the vendor (what it is doing) and the blender's buttons references an “input,” a command or instruction to modify or initiate a new action on the blender.**

Communications may include Wi-Fi, Bluetooth, serial, public switch telephone network, WiMax, LTE, etc. In other embodiments the FDK may include more or less characteristics and/or use different categorizations.

A user may choose multiples of each characteristics and/or types within each characteristic. In some cases, a user may be able to add additional specifications related to the environment which the IC would be connected into. These specifications may include, for example, real estate (chip size and dimensions), clock speed; interrupts; operating voltages, current, power; communication types such as HDL; and the like. In some embodiments, the FDK may dictate the minimum necessary elements for the designed IC to run properly.

Each characteristic and/or type may be linked with a predetermined standardized IC for performing and controlling those characteristics and types on a device. The FDK may design and develop a complete IC design layout and/or schematic that could be sent to a IC

manufacturer for manufacturing. The FDK may have a library of standard IC parts which each selection is linked to. Once all selections are made the FDK may combine the standardized IC parts and provide a CAD design of a resulting IC. Similar to how CAD software may provide a library of standard cells or blocks when designing an IC, the FDK may provide standardized IC components that fully function and can control a device based on the user's chosen characteristics.

108. EX-1001, 14:32-15:22.

109. As quoted and bolded above, the ‘657 patent teaches several specific control, display, input and output parameters. These include the teaching that selecting the components for the device to have a “display” is a selection of output parameters and also display parameters and selecting the components to have “buttons” is a selection of input type parameters.

110. The ‘657 passages above also provide a more generic statement of the parameters. As noted in the quote above, “output” parameter includes a component showing “what [the device] is doing.” An “input” parameter type includes a component reflecting “a command or instruction to modify or initiate a new action.” EX-1001, 14:51-63.

111. Rompaey discloses such selection of the parameters through selection of functions that specify the parameters of the device in the same way as in the ‘657 patent. For example, Rompaey’s device design includes

selection/specification that the device will have a “display” and selection of the components for the “display” (just as the ‘657 patent references selecting that the device will have the function of a “display” and the components for a “display”).

112. Similarly, Rompaey’s device design includes selection of “inputs” by specifying/selecting that the device will have a “keyboard” and control buttons and selecting specific components to implement that input function for the device. This is a selection of the function for “a command or instruction to modify or initiate new action on the [pager]” as identified in the ‘657 patent for an “input.” EX-1001, 14:60-63. For example, the user pushes / types on the pager’s keyboard or pushes the pager control button – an “input” parameter / functionality akin to the ‘657 patent designer selecting that the designed blender device will have the function type of a button and components to interface with such a button. EX-1001, 9:4-29. The keys on a keyboard are effectively buttons. When pushed (or depressed) by a user, it causes an electrical signal that is recognized by the interface to the button / key. So, in Rompaey, the depressing of a key (or button on keyboard) causes an electrical signal that is recognized ultimately by the “User Interface” which a POSITA would recognize can be

implemented by the “UART” which controls “peripherals” as described in Rompaey and which can be selected as a component. EX-1004, 4:38-49, Figs. 10, 11, 12, 13, 17.

113. Such selections are shown through Rompaey for each of the components as detailed below. First, the specification of the functionality is selected. Then the components reflecting that functionality are selected and input. Then the system is refined by further selections as necessary for specific aspects of the functionality (such as protocols).

114. As I noted above, Rompaey’s CoWare tools perform the claimed “selecting” and “receiving” of the selections for the different parameter types (control, display, input and output) using at least three different ways of selecting and receiving data into the memory of the workstation that runs Rompaey’s CoWare tool: (1) a designer-specified “description of the system;” (2) library files representing categories of different reusable components that are mapped onto the system design selected by the designer or CoWare; and (3) designer or CoWare selections with “tools” that refine the specific functions of the components within the designed device.

115. Figure 18 exemplifies CoWare's system design process and the ways in which CoWare receives selections into memory.

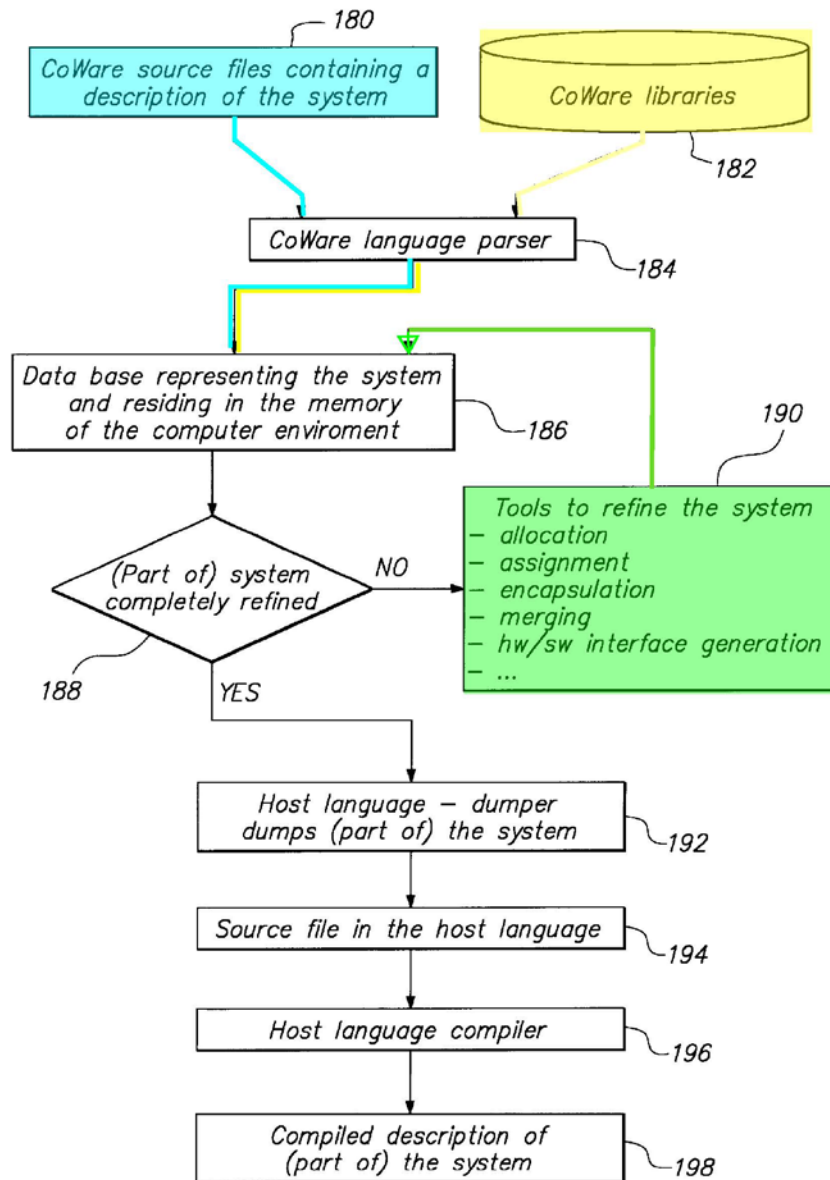


FIG. 18

EX-1004, FIG. 18.

FIG. 18 is an operational flowchart for a presently preferred method of specifying a hardware/software codesign for an embedded system known as CoWare. At state 180, the **system designer creates software source files containing a specification of the system.** **This initial specification is parsed, in conjunction with libraries of predefined routines 182, to result in a database of objects which is stored in the computer memory as shown at states 184, 186.** The CoWare libraries contain processor models, I/O scenario models, protocol definitions, process definitions for use as hierarchical port or hierarchical channel, and so forth. The database could be construed using the Standards Components Library for Graphs, C++ Standard Components, Release 3.0, Copyright (c) 1991, 1992 AT&T and UNIX System Laboratories, Inc. This defines classes for each object in a graphical representation of the system undergoing codesign and has API functions that are used to build graphs, access elements of a graph, and so forth. **In addition, the database requires library functions or tools that implement graph algorithms which correspond to the described merger, flattener, and other tools of CoWare.**

116. EX-1004, 26:50-27:3. The **blue**, **green**, and **yellow** blocks/passages are examples of distinct actions of the claimed “receiving.”

117. The source files, libraries, database, and “classes for each object” are “hierarchical” and categorized which I have detailed below. The passage above references that different categories of functionality (such a processor

models, I/O scenario models...) can be construed using the “Standard Components” library as specified.

118. This “Standards Component” library was known to have a hierarchical classified structure in which system designers could select from different categories of components to implement various features and functionalities – as is expressly stated in this quote. As noted from the name, one of these “Standard Components” provided the ability to create functionality such as “Graphs” (referenced in the description of Figure 18 above). This functionality “defines classes for each object in a graphical representation of the system undergoing codesign and has API functions that are used to build graphs, access elements of a graph, and so forth..” EX-1004, 26:64-67.

119. “Classes” in this context would be recognized by a POSITA as a form of categories. For example, Rompaey teaches “classes” of each of the components (as noted in the quote above) and the hierarchal structure of these components allows for selecting of subclasses (such as a processor type) and further selections thereafter to continue to refine the specific components functionality and operation.

120. For example, these classes would allow the different categories identified (“processor models, I/O scenario models, protocol definitions, process definitions for use as hierarchical port or hierarchical channel, and so forth”) to be organized and classified in a hierarchy. The protocols are stored in the library databases and “Primitive protocols provide a classification of all hierarchical protocols.” EX-1004, 15:29-30.

121. For example, using the disclosure of Rompaey, the classification of “processor models” would be constructed to have at least the classification of two different processors identified in Rompaey underneath it (the “Arm6” and “Cathedral” processors shown on, for example, Figure 2, 6, 7, and 15) while the “I/O scenario models” would include at least the different classifications of I/O scenarios in Rompaey at 19:29-61. These classifications are stored in a library and database as noted in the quotation from 26:50-27:3 above.

122. Thus, as detailed in Figure 18 (and the description above), a designer would select from the categories in a hierarchal classification library the initial specifications with components reflecting the device functionality. The designer and CoWare then uses the “Standards Components” library to select (and receive) from categories in a hierarchal classification database

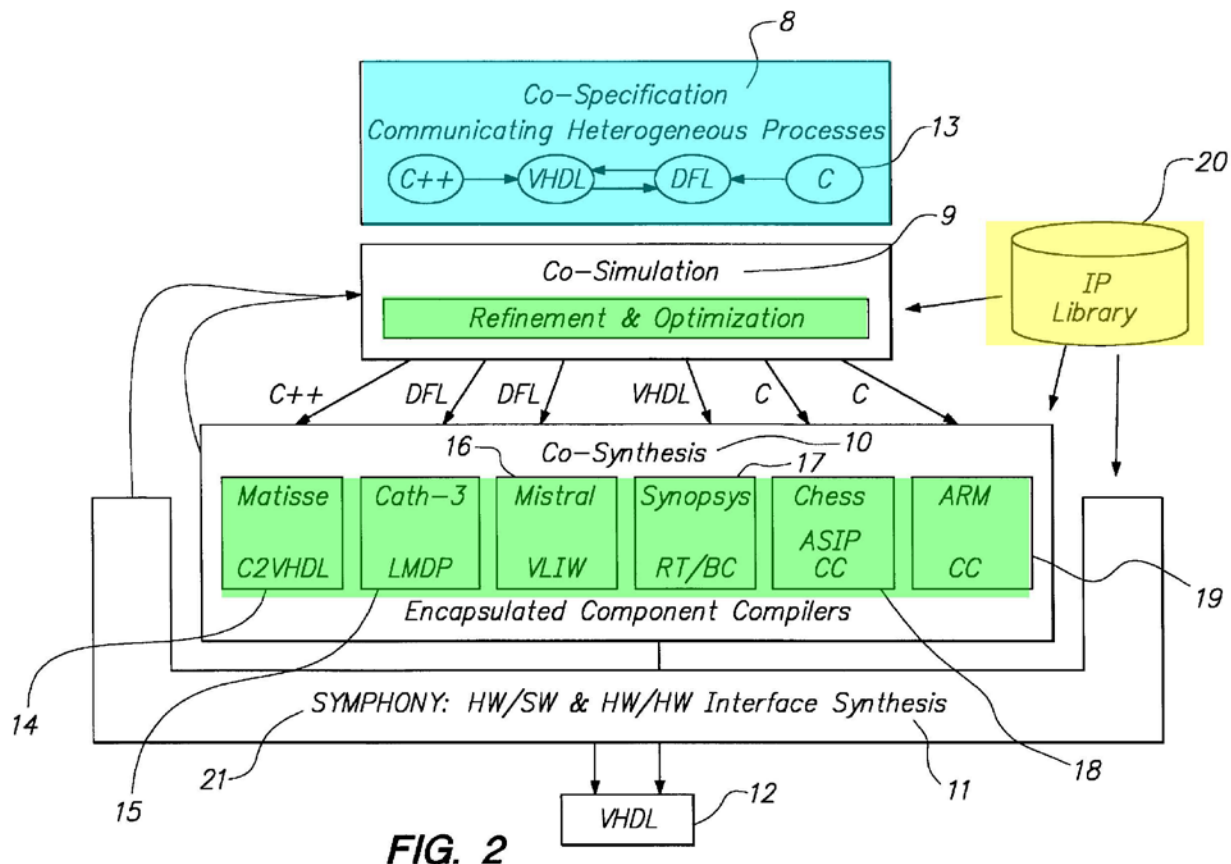
(e.g., the Standard Components Library) the components that correspond to initial selected specification. The designer would then refine this selection using “functions or tools that implement graph algorithms” (as described above and below) which allowed the user to select further parameters and components for the design and its functionality from various classification libraries including the Standard Component library. These hierarchal objects stored in the database are used to select these parameters that include control type, display type, input type and output type as expressly disclosed in Rompaey and discussed below.

123. A POSITA would recognize that this teaching is not specific to the “Standards Component” libraries, but rather encompasses the general approach in Rompaey for a designer to select from pre-existing functionalities and components implementing those functionalities, select from parameters / characteristics to refine those functionalities and use those selections to generate hardware and software to implement designed devices / systems.

124. Each of these types of selections are “received” into the “database representing the system and residing in the memory of the computer environment.” EX-1004, Fig. 18, element 186. This corresponds to, for

example, the requirements in the claims of “a memory of a processor programmed by software commands.” As shown in Rompaey, this is a single memory which performs all of the recited actions of the memory in the claims.

125. The process summarized in Figure 18 and its description is shown in the other Figures as well. Fig. 2 depicts this overall process in showing the “**co-specification**” of the system, the “**refinement and optimization**” of the system, and the “**IP Libraries.**”



EX-1004, Fig. 2.

126. In the first step, the designer creates a system design using a “data model” with categorized hierarchical objects with “processes,” input/output “ports” and “channels” from a database.

FIG. 2 shows the architecture of the CoWare system. The CoWare system supports four major design activities: co-specification 8, co-simulation 9, co-synthesis 10 and interface synthesis 11. The input is a heterogeneous specification of an electronic system, the output 12 may include a net list for prior-art commercial tools for the generation of the implementation layout. This output preferably comprises structural VHDL or Verilog and machine code for the programmable processors.

The CoWare design environment is implemented on top of a data model in which modularity is provided by means of processes. Processes contain host language encapsulations which are used to describe the system components. Communication between processes takes place through a behavioral interface comprising ports. For two processes to be able to communicate, their ports must be connected with a channel. The inter-process communication semantics is based on the concept of the Remote Procedure Call (RPC). The data model is hierarchically structured and allows the refinement of channels, ports, and protocols into lower level objects, and add detail. The most abstract object is referred to as a primitive object. An object that contains more implementation detail than a primitive object is referred to as a hierarchical object.

EX-1004, 11:12-35.

127. Objects, ports and channels define the inputs and outputs of processes.

Rompaey summarizes the designer's specification options:

We first discuss the primitive objects. The hierarchical objects are used to refine the communication behavior of the system and are discussed afterwards.

A process is a container for a number of host language encapsulations of a component. A single process can have multiple host language encapsulations describing different implementations for the same component, or for the same component represented at different abstraction levels.

A host language encapsulation describes a component in a specific host language. Preferably C, C++, DFL, VHDL and Verilog are supported host languages. A CoWare language encapsulation is used to describe the system's structure. In a CoWare language encapsulation, one can instantiate processes and connect their ports with channels. Other host language encapsulations comprise context and a number of threads. The context and thread contain code written in the host language of the encapsulation. The context contains code that is common to all threads in the encapsulation, i.e., variables/signals and functions as allowed by the semantics of the host language. As such, the context provides for inter-thread (intra-process) communication.

Each primitive CoWare process (symbolized by an ellipse 13 in FIG. 2) encapsulates concurrent program threads in a host language of choice. Concurrent threads communicate over shared memory inside a

process. Inter-process communication is over uni-directional channels using a Remote Procedural Call (RPC) protocol. The reasons of this choice will be explained below. Notice that in this way heterogeneous specification is supported: both hardware and software aspects, structural and behavioral aspects, and different specification paradigms (data-flow, control-flow, and so forth) can be combined. Co-specification allows to describe a functional specification based on the concept of communicating CoWare processes.

EX-1004, 11:21-12:5.

128. A POSITA would recognize that this classification of design selections as corresponding to the selection of control, display, input and output parameters and functionality as recited in the '657 claims. For example, the description above is generic enough to allow for any "host language" to describe the overall system design as reflected by the different types of hierarchical objects used to represent the design.

129. A POSITA would recognize a hierarchical object as corresponding to one of the categories of selections that can be made from the classification library of different representations for the different functionalities within the designed system. The hierarchical object is progressively defined through user selections of features / characteristics / parameters /

functionalities of that object which also reflect characteristics of the designed device.

130. Rompaey then provides a further description of the design process using CoWare.

In refinement for co-synthesis the designer performs an interactive coarse partitioning of the specifications over a user allocated architecture. This leads to a merger of component compiler consistent processes to be mapped on the same component. Component compiler consistent processes have an encapsulation in the same host language. Merging consists of in-lining the RPC calls between the processes and leads to two subproblems: the mapping of the concurrent threads in the processes on a processor re-using existing component compilers 14, 15, 16, 17, 18, 19 and the refinement of the communication between processes into hardware and software communication protocols that implement it. **The implementation of concurrent threads and intra-process communication must be taken care of by using Real-Time Operating Systems (RTOS), micro-kernels or software synthesis in case of programmable processors or by providing a library based communication protocol shell around the existing hardware synthesis tools.** Refinement of the inter-process communication means again a refinement of the primitive RPC communication by expanding the communication ports into implementable protocols available in a protocol library 20. It is also possible to assign channel processes to abstract channels.

In principle all of this is open to the user who can add his own library for communication protocols. On the other hand, CoWare provides in the SYMPHONY toolbox 21 a methodology for interface synthesis whereby every communication channel is refined by selection of a communication scenario. In this way, automated synthesis of hardware/hardware and hardware/software interfaces, including the generation of the software drivers in programmable processors is possible. This is an essential part of hardware/software co-design.

After the compilation of all components, all hardware is available as structural VHDL and all software for the processors is in C which can be compiled on the host compiler of the programmable components. The final step is to link all the synthesis and hardware descriptions to drive commercial back-end tools to generate layout.

EX-1004, 12:12-50

131. This passage regarding refinement reflects that a designer will use the libraries and repositories (including the Standard Component library referenced above) which will select from the categories of components in the hierarchical classification database of the library to implement the initially selected functionality. For example, for the “component compiler” “the mapping of the concurrent threads in the processes on a processor re-using existing component compilers 14, 15, 16, 17, 18, 19” (as shown in Figure 2) reflects a selection of pre-existing components to perform the

functionality selected in the initial system specification. These different components are stored in the pre-existing classification libraries.

132. Further refinement of the selected functionality is done using the “protocol library” which a POSITA would also understand to correspond to a hierarchical classification library containing different protocols used for communication. The designer refines the design by selecting from those different categories of protocols to implement various parameters / functionalities of the designed device including control, display, inputs and outputs. “Refinement of the inter-process communication means again a refinement of the primitive RPC communication by expanding the communication ports into implementable protocols available in a protocol library 20.”

133. As noted in the quote above, designers can store their refined designs back into the libraries for use in subsequent design projects for devices that may have similar functionalities.

134. Rompaey extensively details the specific options of inputs, output, and processing for each component. EX-1004, Figs 3, 4, 5; 12:51-15:49, claims 1, 6-12.

135. Figures 3 and 4 show examples of the user's design process. For example, Figures 3 and 4 show the design, refinement and implementation of components that reflect an RS-232 port. RS-232 is well known to a POSITA as a component that implements inputs and outputs for a device.
136. For example, in the 1980s and 1990s, RS-232 ports were included on the back of desktop computers so that peripherals, such as keyboards, could be plugged into the device and used for input or outputs.
137. Figure 3 shows an exemplary initial design. "FIG. 3 is an illustration of exemplary primitive objects in a co-design database and the relations in between the primitive objects." EX-1004, 6:61-63.
138. Figure 4 shows how the design in Figure 3 can be implemented in hierarchical objects. "FIG. 4 is an illustration of the exemplary hierarchical objects in the database and the relations in between the hierarchical objects and between the primitive objects shown in FIG. 3 and the hierarchical objects." EX-1004, 9:64-67.
139. The hierarchical categorized objects are used to select and define the parameters associated with, or of, the device, including control, display, input and output characteristics.

140. Figure 3 shows a sample “process subsystem” within the overall category of “process systems.” It reflects the selection of two processes and the communication between those two processes (“inter-process communication”).

141. Figure 3 shows a form of pseudo-code which is intended to convey the functionality of the processes. There are different functionalities / parameters within the overall model that the user may select. For example, Rompaey teaches:

Protocols define the communication semantics of a port. A primitive protocol is one of master, in master, out master, inout master, slave, inslave, outslave, inoutslave. Each primitive protocol indicates another way of data transport. The in, out, and inout prefix indicates the direction of the data. The master, slave postfix indicates the direction of the control: whether the protocol activates an RPC (master) or services an RPC (slave). In the remainder of this text, ports with a slave/master protocol are usually referred to as slave/master ports.

In FIG. 3 master ports (29, 30) are represented by the small shaded rectangles on a process' perimeter. Slave ports (27, 28) are represented by small open rectangles on the perimeter. The data direction of a port is represented by the arrow that connects to a port. In FIG. 3 port p1 (29) is an outmaster port and port p2 (27) is an inslave port.

A protocol may further have an index set. The indices in the index set are used to convey extra information about the data that is transported. For example the primitive protocol used to model the memory port of a processor will have an index to model the address of the data that is put on the memory port.

EX-1004, 12:64-13:18.

142. For example, as noted above, the designer would have to select from different categories of input / output parameters such as “master, in master, out master, inout master, slave, inslave, outslave, inouts slave.” Similarly, the designer selects from the categories of the parameters specifying a “way” (direction) of transport from the categories of in, out, in/out: “Each primitive protocol indicates another way of data transport. The in, out, and inout prefix indicates the direction of the data.” Similarly, the designer must select from the “direction of control” categories of parameters (master or slave).

143. As further reflected above, the port must be assigned a protocol in the first place. There may be further selections such as the “index set” which reflects extra information about the data as noted above and all constitute selected input and/or output parameters.

144. Similarly, as shown in Figure 3, there are “channels” that have various parameters that the user must select from categories as reflected in the pre-defined components available for selection including:

A channel is a point-to-point connection of a master port and a slave port. Two ports that are connected by a channel can exchange data. Channels can be uni- or bi-directional. A primitive channel provides for unbuffered communication. It has no behavior: it is a medium for data transport. In hardware it is implemented with wires. In software it is implemented with a (possibly in-lined) function call. In this way, primitive channels model the basic communication primitives found back in software and hardware description languages.

EX-1004, 42-51.

145. The channel selection is related to the selection of ports to implement that channel and the communication behaviors selected for those ports. The designer can select from different categories of pre-defined functionality to implement any of the different components and their own functionality:

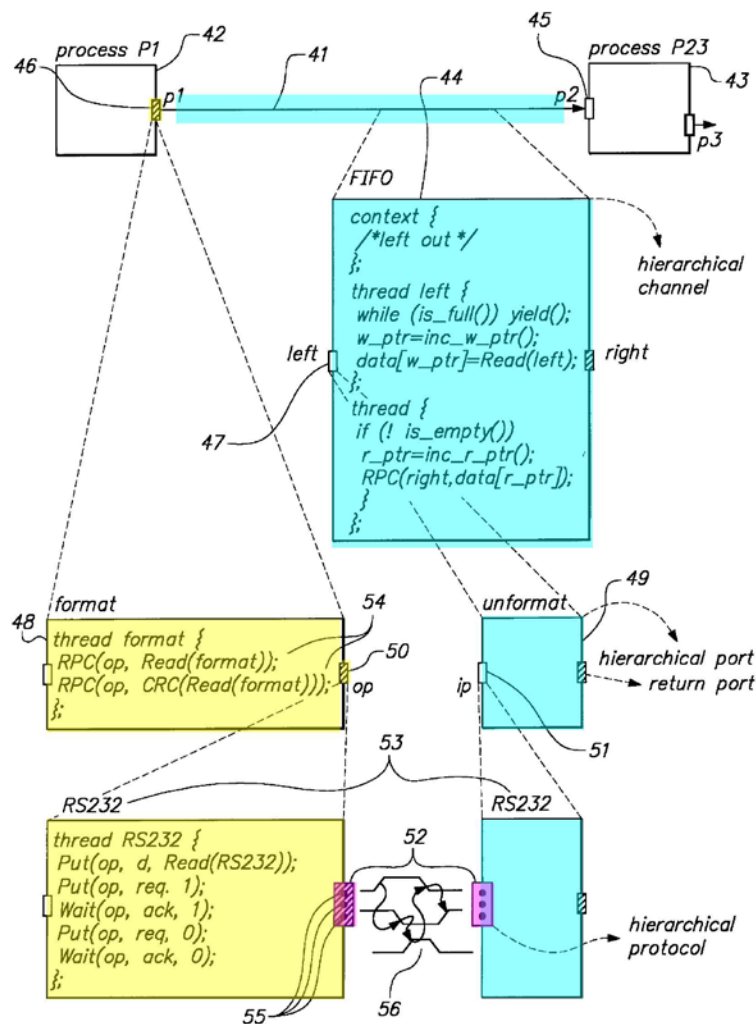
Hierarchical channels are processes that assign a given communication behavior to a primitive channel. The behavioral interface of a hierarchical channel is fixed by the ports connected by the primitive channel. Making a channel hierarchical, can drastically change the communication behavior of two connected processes. It can, for example, parallelize (pipeline) the processes by adding

buffers. The one property that is preserved by making a channel hierarchical is the direction of the data transport.

EX-1004, 14:49-57.

146. Figure 4 below details further examples of refinement from Figure 3 in which the designer can select, define and refine the parameters of the design in Figure 3 using an “RS232” port. EX-1004, 14:58-15:49.

147. As shown in Figure 4, the processes have certain functionality defined at their inputs and outputs (such as “input/output ports” p1 and p2). That functionality is shown in the yellow and blue boxes below. The user will select the specific types of protocols (in this instance, RS-232) and then select further parameters to implement those protocols (which, in this instance, is an input / output parameter as reflected in the type of protocol and the specific parameter implementing that protocol).

**FIG. 4**

EX-1004, Fig. 4.

148. As noted above for Figure 3, Figure 4 reflects the user selections for many different categories of the functionality and characteristics of the designed device. This is exemplified in the following passage:

Hierarchical ports are processes that assign a given communication behavior to a primitive port. The behavioral interface of the hierarchical port is partially fixed by the primitive port it refines. The

hierarchical port process should have one port, which we call the return port, that is compatible with the primitive port. Making a primitive port hierarchical, preserves the data direction (in/out). Two ports are compatible if their primitive protocols are compatible, if they have equal data type, and if they have equal protocol indices. The following primitive protocols are compatible: (master, slave); (inslave, outmaster); (inslave, inoutmaster); (outslave, inmaster); (outslave, inoutmaster); (inoutslave, inmaster); (inoutslave, outmaster); (inoutslave, inoutmaster). Two hierarchical protocols are compatible if their primitive protocols are compatible and they have the same name.

In FIG. 4 we impose a certain data formatting for the data transported over the channel (41) between process P1 (42) and the FIFO hierarchical channel (44). This is achieved by making the primitive ports p1 (46) and left (47) hierarchical. The format process (48) that refines port p1 (46) might for example add acyclic redundancy check to the data that is transported. The unformat process (49) that refines port left (47) of the FIFO hierarchical channel (44) then uses this cyclic redundancy check to determine whether the received data is valid. The actual data and the cyclic redundancy check are sent sequentially over the same primitive channel between ports op (50) and ip (51). As a consequence, the data rate between the format (48) and unformat (49) process is twice the one of process P1 (42).

Primitive protocols provide a classification of all hierarchical protocols. A primitive protocol determines the communication semantics, but not the communication implementation: it does not fix

the timing diagram used in the communication. Hierarchical protocols refine primitive protocols with a timing diagram and the associated I/O terminals. Hierarchical protocols are high level models for alternative implementations of a primitive protocol: they preserve both data direction and control direction of the primitive protocol.

EX-1004, 15:1-39.

149. Figure 4 shows the hierarchy of the categorized data model in which a process (42) includes an **input/output port** (p1) associated with a “**FIFO hierarchical channel (44)**” which selects sub-parameters such as the “protocol,” “data format, data direction and “data rate.” EX-1004, 15:16-43; *see also* EX-1004, 9:64-67.

In FIG. 4, the primitive protocol of the port (50) and ip port (51) of the format (48) and unformat (49) process are refined into an RS232 protocol (52). In the RS232 hierarchical port (53), an RPC (54) issued in the format process (48) on the op port (50) is converted into manipulations of the terminals (55) according to a timing diagram (56).

150. EX-1004, 15:44-49. Thus, the designer has selected, and the CoWare tool has received, the claimed input and output parameters by selecting the port (RS-232) expressed through the “terminals” that perform the input and output according to the sub-parameters for that RS-232. As RS-232 ports implement display, input and output for components such as displays,

keyboards and controls such as control buttons on a pager, such designer specifications also are the display, input and/or output parameters of the overall device. *See also* EX-1004, Fig. 1 (element 5); 3:19-25.

151. Additionally, Rompaey teaches the design of an exemplary device – a pager – in which the designer selects the display, input and output parameters by selecting that the device will include display, input / output functions of a display and keyboard and the details of the implementation of those functions in the design for the device.

152. Figures 7-17 show the various aspects of the design of the pager example.

153. In some instances, Rompaey references its designed devices as a “system” or “application.” “FIG. 10 is a block diagram of a typical heterogeneous digital system: a pager application.” EX-1004, 10:16-17. Fig. 10 shows an example of an initial selection of functionality and parameter types using the pager example presented in Rompaey.

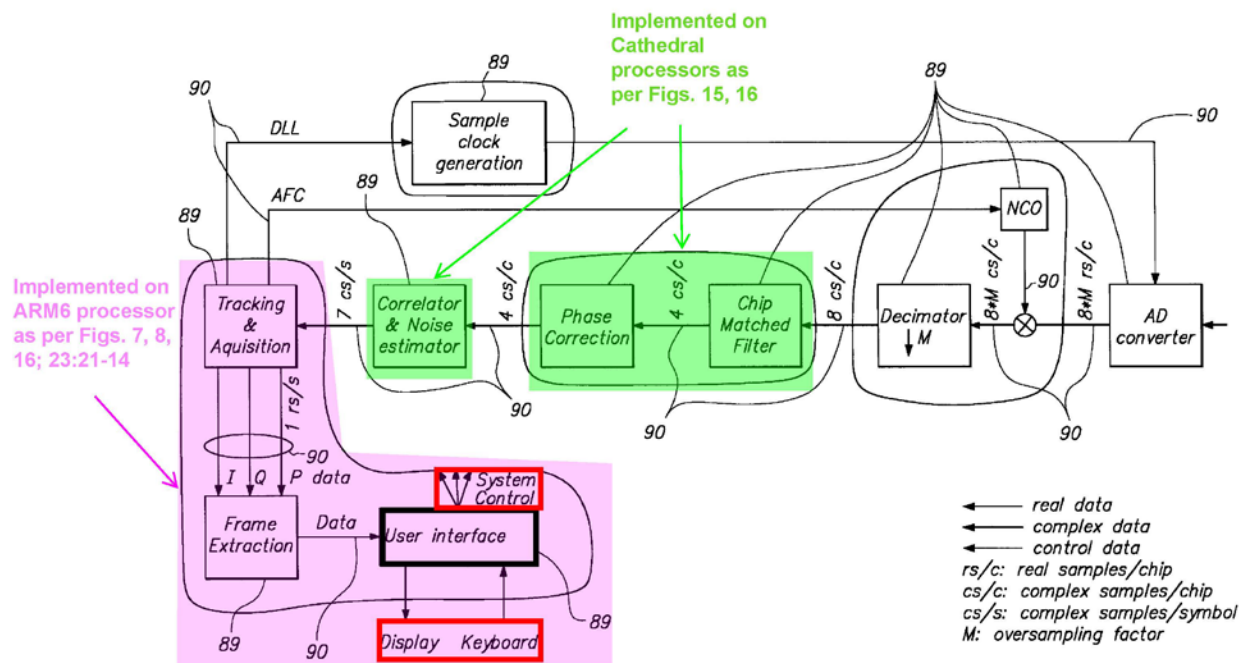


FIG. 10

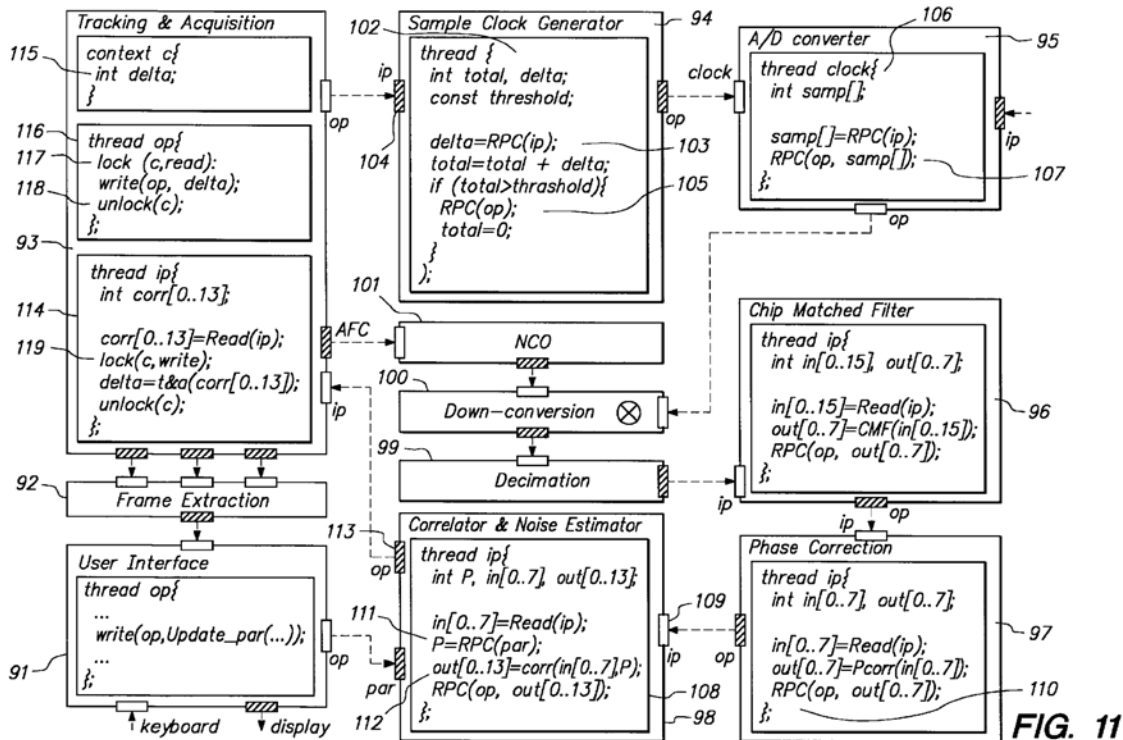
154. EX-1004, FIG. 10. In designing the pager device in Figure 10, the designer has selected that the device has a “user interface” with selected and designed functionality for a “display,” a “keyboard,” and “system control[s]” – which are further examples of claimed control, display, inputs and outputs of the device.

155. The designer selects specific functions and components corresponding to the functionality of the device (or “system”). For example, “FIG. 11 is a schematic representation of the pager application, introduced in FIG. 10.” EX-1004, 10:18-20.

156. Fig. 11 is in the format of the designer's specification discussed above. Each of the blocks 91-101 in Figure 11 show details of the implementation of the various different blocks labeled as "89" in Figure 10. "FIG. 11 shows the RPC communication in detail for part of the pager design. The blocks (91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101) in FIG. 11 correspond to the processes (89) from FIG. 10." EX-1004, 21:41

157. Fig. 11 shows that the designer selects that the device will have a user interface that includes the specific characteristics of the device in selecting a display (display and output type) (and the requisite functionality to support it) and a keyboard (input type). Figure 11 shows predefined blocks arranged and selected to implement the display, keyboard, and components for receiving and processing wireless signals. These components related to the wireless signal include the down-conversion, A/D converter, Chip Matched Filter, Phase Correction, Decimation, and Correlator & Noise Estimator. In selecting these blocks, the user has selected control type parameters (in the various RF control blocks referenced above), display type parameters (in the blocks that interface with the display), input type parameters (in both the RF input blocks and in the

user interface interfacing with the keyboard) and output type parameters (in, for example, the components implementing the display).



EX-1004, FIG. 11.

Each block (89) in FIG. 10 corresponds to a process implementing a specific function of the pager. This functional decomposition determines the initial partitioning. The lines with arrows (90) in between the processes represent primitive channels with RPC semantics. FIG. 11 shows the RPC communication in detail for part of the pager design. The blocks (91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101) in FIG. 11 correspond to the processes (89) from FIG. 10.

EX-1004, 21:34-41.

158. In Figure 11, the designer has selected from different components to implement the various functionalities with parameters of the device. Rompaey describes examples of a portion of this activity at 21:43-22:45.
159. In a CoWare design, a designer may select functionalities through the use of “processes” in the system specification. “Each process is described in the language that is best fit for the characteristics of the function it implements” such as the “C” language. EX-1004, 22:26-33.
160. Thus, as depicted in the examples above, Rompaey allows for the designer to select from various libraries the functionalities and components that reflect the overall system design and represent those functionalities in block diagram formats and in simplified formats such as in portions of Figured 3, 4, 10, 11 and further selected additional parameters that define and specify the operation of the functionalities and components selected.
161. The second type of selecting action is CoWare “receiving” the data as selected by the designer and CoWare itself from the libraries and “I/O scenarios” as shown in Figs. 2 (“IP library”); 6 (“I/O scenarios”); 7 (“I/O scenarios”) and 18 (“CoWare libraries”). “The CoWare libraries contain processor models, I/O scenario models, protocol definitions, process

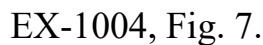
definitions for use as hierarchical port or hierarchical channel, and so forth.” EX-1004, 26:56-27:3.

162. CoWare receives the selection of the implementations of the system block design from these libraries which provide for categories of different components arranged in a hierarchical classification database as described above.

163. “FIG. 6 is a flowchart of a specific embodiment of the implementation process for hardware/software co-design according to the method of FIG. 2.” EX-1004, 10:3-5.

164. Figure 6 shows an iterative process. The designer’s initial selections are used to receive further selections of components from the libraries. As part of this process, the tool analyzes the current design, allows the user to select further refinements, then analyzes the current design and allows the user to repeatedly select further changes for the design.

165. Rompaey’s Figure 7 gives a specific example of the receipt of those categorized selections of inputs and outputs corresponding to the components in the device / system design.



167. These further implementation / components from the functionality are received from the libraries as shown in Figure 7 above in blue and yellow.

88

available in the hardware section of the data sheet of the programmable processor core. The HDL host-language encapsulation of the hardware model is characterized by a behavioral interface that conforms with the hardware interface of the programmable processor. All hardware ports (74) have hierarchical protocols: they consist of terminals and a timing diagram. The hardware model may also contain a HDL description for either a black box, a simulation model, or the full description of the processor core.

The software model (71) of the programmable processor core consists of a software host-language encapsulation that formalizes the information that is available in the software section of the data sheet of the programmable processor. The software host-language encapsulation of the software model is characterized by a behavioral interface that conforms with the software interface of the programmable processor core. All software ports (75) have primitive protocols. The software model identifies, for example, what ports can be used to get data in or out of the processor core (memory mapped, co-processor port, . . .), what ports can be used as interrupt ports and what their characteristics are (interrupt priority, maskable interrupt, . . .). In addition, the software model contains a behavioral description that allows compilation of a software host language encapsulation into machine code. For example: functions to manage processor specific actions such as installing an interrupt vector, enabling/disabling interrupts, etc.

In FIG. 7, the software model of the ARM-6 RISC processor is shown along with a number of its ports (75). The memory port mem is

modeled as a (bi-directional) slave port. This slave port is accessed by the device drivers, by means of an RPC, to write/read data to/from the external hardware. The slave thread, modeled in the software model, attached to the mem slave port translates the incoming RPC to a memory access. SYMPHONY makes the connection between the device drivers and the mem port. The fiq port, an interrupt port on a programmable processor, is modeled as a master port. The software model of the ARM processor ensures that an RPC to this port is performed every time the processor detects that an interrupt has occurred. SYMPHONY connects the fiq port to a slave thread that serves as the interrupt service routine so that routine is started automatically.

In FIG. 7, the hardware model of the ARM-6 RISC processor is shown along with a number of its ports (74). The memory port mem is modeled as a (bi-directional) master port. The hardware model of the ARM- performs an RPC to port every time that it wants to write/read data from the RAM, ROM or memory mapped hardware. SYMPHONY connects the mem port to a slave thread in the hardware interface, which does the address decoding and forwards the RPC to the appropriate hardware block (RAM, ROM, memory mapped hardware).

The fiq port is modeled as a slave port in the hardware model. This slave port is activated by an RPC that is performed to the port by the hardware interface. The slave thread attached to the fiq port (and modelled in the hardware model) sets the appropriate flag in the status register to the appropriate value, signalling the interrupt request.

The link between events (RPC to ports, starting of slave threads attached to slave ports) in the hardware model and events in the software model, is handled by the processor hardware or, in the case of simulation, by the instruction-set simulator for the ARM.

EX-1004, 17:65-18:67.

168. Categories of selections from the databases stored in the libraries include the types of processors which implement the control of the device (using software, for example) and the I/O scenarios which implement the displays, inputs and outputs for the devices depending on the designer's selections of particular components and their functionality and parameters.

169. For example, the "I/O scenarios" are stored in libraries along with models of other components. "FIG. 7 illustrates the tool SYMPHONY. SYMPHONY makes use of a software model (71) and hardware model (72) of the target processor and a library of I/O scenarios (73) for that processor." EX-1004, 17:65-18:1.

SYMPHONY is based on the observation that programmable processor share a number of common communication methods to get data in or out of the processors. These communication methods are modeled by I/O scenarios. An I/O scenario describes one way of using the ports of a specific processor core to map a particular port of a software host language encapsulation to an equivalent port in hardware, thereby crossing the processor core boundary while

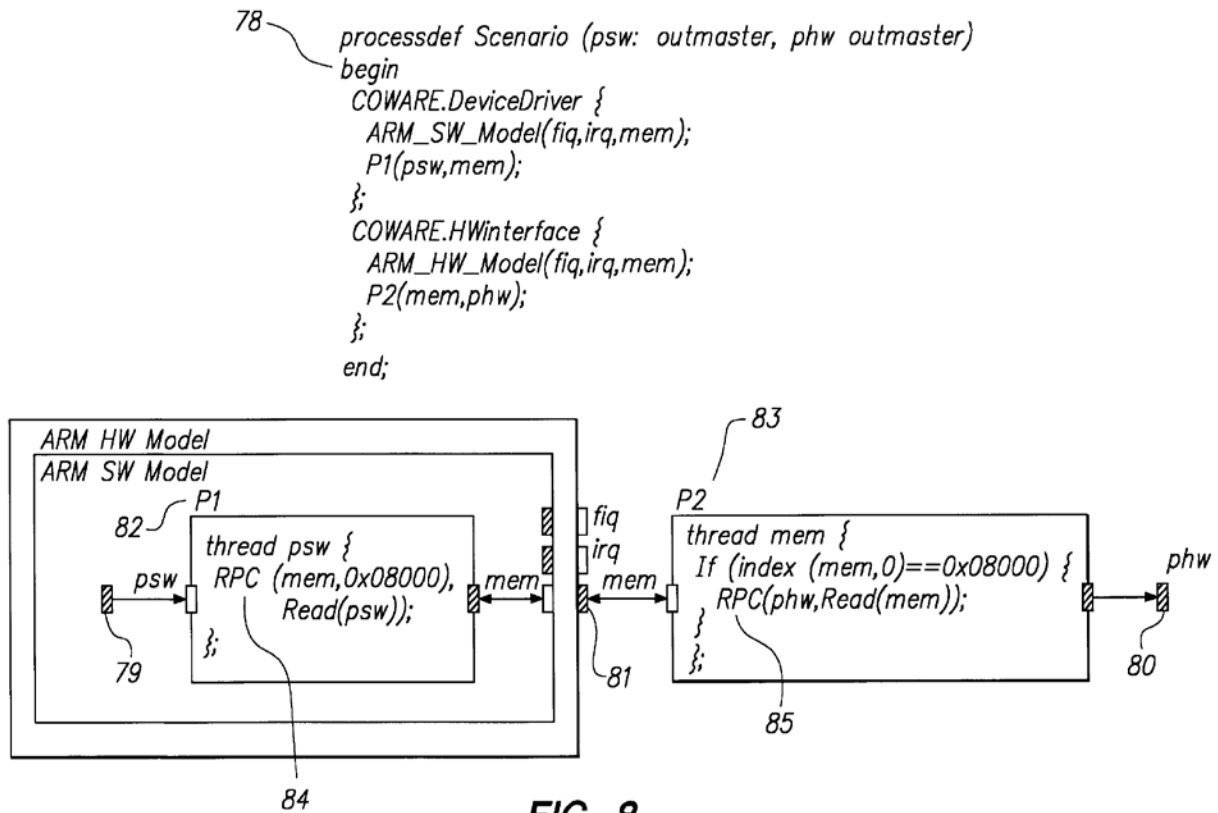
maintaining the communication semantics. FIG. 8 shows an example of an I/O scenario. It consists of a software host-language encapsulation and a hardware host-language encapsulation that describe a software I/O driver and the hardware counterpart, respectively. **An I/O scenario is also tagged with some performance figures that will allow the designer or SYMPHONY to make a decision about what I/O scenario to use for which port.**

The I/O scenario (78) of FIG. 8 shows how an outmaster port psw (79) in software can be mapped to an outmaster port phw (80) in hardware, thereby using the memory port (81) of an ARM-6 RISC processor core. The software process encapsulation P1 (82) represents the software I/O driver and copies data from port psw (79) to a specific memory address 0×08000 via an RPC call (84). The hardware process encapsulation P2 (83) represents the hardware counterpart and checks whether the memory address bus of the ARM (modeled by the protocol indices of the memory port (81)) equals 0×08000. If this is the case, data that is residing on the memory data bus of the ARM is copied to port phw (80) via an RPC call (85).

EX-1004, 19:1-27.

170. This passage teaches is a very explicit example of how a designer makes a selection (using CoWare) of one or more specific I/O scenario(s) (from the categories of different I/O scenarios identified below) that is in an hierarchical classification database (the libraries such as the “Standard Components” that store the I/O scenarios as I have discussed above).

171. There are several different categories of I/O scenarios that the designer can select from in the library / database to implement the desired functionality for the device. “The library of I/O scenario comprises:” (1) “memory-mapped I/O scenarios” which define characteristics for an input/output via memory-mapped I/O ports; (2) “Instruction-Programmed I/O scenarios” defining characteristics for “specially communications ports of the processor ... connected to the external channels via the I/O unit”; and (3) “hardware interrupt controls” that define the characteristics of the interrupts (inputs) of different processors. EX-1004, 19:29-20:19. This is at least three separate “categories” of parameters that can specify input, output, display and control types for the device – each with their own set of categorized parameters selected after the selection of the top-level category.
172. “Fig. 8 shows an example of an I/O scenario.” EX-1004, 19:9-10. It shows the specific hardware interfaces and associated software to for control, display, input and output hardware for an exemplary ARM6 processor and the parameters of those inputs and outputs for this particular I/O scenario. EX-1004, 19:10-28.

**FIG. 8**

173. EX-1004, Fig. 8. The “fiq” and “irq” are interrupts and the “mem” is a “memory-mapped I/O” – each being an input / output parameter of the device along with specific sub-parameter.

174. CoWare receives further selections when the designer “creat[es]” and “select[s]” various processor models and I/O scenario models of the ports from objects in the library / database to implement the designed system/device using, for example, selected protocols based on libraries of re-usable components.

175. Rompaey expressly identifies the I/O scenarios as being “selected:”

In an aspect of the present invention, the design environment further includes means for creating processor models of the hardware components as objects in the database, the models comprising software models representing the software views on the hardware components and hardware models representing the hardware views on the hardware components. The design environment further includes means for creating I/O scenario models of the ports as objects in the database, the I/O scenario models representing the implementation of the ports on the hardware components, the implementation comprising software subsystems, hardware subsystems, and processor models with connections therebetween. The implementation of the communication between a first software subsystem and a first hardware subsystem results in the first software subsystem with a first port being replaced by a second hardware subsystem with a second port, where the first port and the second port represent an essentially identical communication. The design environment further comprises **means for selecting I/O scenario models** for the ports of the first software subsystem; means for combining the software subsystems of the **selected I/O scenarios**; and means for combining the hardware subsystems of the **selected I/O scenarios**. A first I/O scenario model represents the connection of the first port to the second port, the connection comprising a connection of the first port to the software subsystems of the I/O scenario model, connections of the software subsystems of the I/O scenario model to the software model, connections of the hardware model to the hardware subsystems of the I/O scenario model, and a connection of the hardware subsystems of

the I/O scenario model to the second port. The I/O scenario models comprise any one or more of memory mapped I/O scenarios, instruction programmed I/O scenarios, and interrupt based I/O scenarios.

176. EX-1004, 8:5-38.

In principle all of this is open to the user who can add his own library for communication protocols. On the other hand, CoWare provides in the SYMPHONY toolbox 21 a methodology for interface synthesis whereby every communication channel is **refined by selection of a communication scenario**. In this way, automated synthesis of hardware/hardware and hardware/software interfaces, including the generation of the software drivers in programmable processors is possible. This is an essential part of hardware/software co-design.

177. EX-1004, 12:36-40.

The port containing the hierarchical protocol, is made hierarchical to add the required communication behavior that implements the timing diagram of the **selected hierarchical protocol**. Again this is achieved without modifying the description of any of the processes involved. Because of this property it is feasible to construct libraries of functional building blocks and libraries of communication blocks that are re-usable: they can be plugged together without modifying their description. After blocks have been plugged together, any communication overhead (chains of remote procedure calls) can be removed by in-lining the slave threads that serve the RPCs. The result is a description of the component in which function and

communication are interleaved seamlessly and which can be compiled into software or hardware as efficiently as a description in the traditional design process.

The above method reduces the amount of protocol conversions needed at the system level and allows to postpone the selection of the communication protocol and its implementation until late in the design process, in this way achieving the requirements of "design for re-use". The concept of hierarchical protocols is also useful to model off-the-shelf components ("re-use of designs"), because the timing diagrams according to which a processor communicates are abstracted in it.

178. EX-1004, 16:50-17:7.

Once an I/O scenario is selected for every port of the software host-language encapsulation, SYMPHONY generates the necessary communication software and the corresponding hardware I/O unit by combining the selected I/O scenarios.

179. EX-1004, 20:10-14.

Communication Mechanism Selection

After the partitioning of the system has been verified by simulation and before the actual implementation takes place, the designer may choose to refine the communication mechanism between the processors. This can be achieved by making explicit the behavior of the channels between the processors.

In the running example, the processors can, in principle, operate concurrently because each processor has its own thread of control. By

refining the RPC based communication scheme we can pipeline the processors: all processors operate concurrently and at I/O points they synchronize. This refined communication scheme is called Blocked/UnBlocked Read/Write communication. FIG. 13 shows the pager with the refined communication mechanism. The inputs and outputs of the processors have been labeled with BW for Blocked Write, BR for Blocked Read, and UBR for UnBlocked Read.

BW-BR communication guarantees that no data is ever lost. When the writing process has data available, it will signal that to the reading process. If the reading process is at that moment not ready to receive the data (because it is still processing the previous data), the writing process will block until the reading process is ready to communicate. Alternatively, if the reading process needs new data, it will signal that to the writing process. If the writing process is at that moment not ready to send the data (because it is still computing the data), the reading process will block until the writing process is ready to communicate. The BW-BR scheme is used in the main signal path.

A BW-BR scheme, however, is not used for the parameter and mode setting for the main signal path. If an accelerator uses BR to read a parameter value, it will be blocked until the parameter is provided. Since the parameter setting is done in software, this will slow down the computations in the main signal path considerably. Therefore parameter setting is done via a BW-UBR scheme. This makes sure that every parameter change is read by the accelerators, but it leaves it up to the accelerator to decide when to read the parameter.

In the CoWare design environment, the refinement of the communication mechanism is performed by making use of a hierarchical channel. A hierarchical channel replaces a primitive channel by a process that describes how communication over that channel is carried out.

The introduction of BW-BR communication is shown in detail for the Chip Matched Filter & Phase Correction (135) and Correlator & Noise Estimator process (136) in FIG. 14.

The BWBR channel (137) contains an autonomous thread (138) and a slave thread (139) that communicate with each other via the shared variable `tmp 0..7!` in the context (140). The slave thread (139) is activated by an RPC (141) from the CMF & Phase Correction process (135) and it tries to update the context (140) with new values. The autonomous thread (138) continuously tries to read the values from the context (140) and send them to the output port (144) via an RPC (142), in this way activating the Correlation & Noise Estimator process (136) that is attached to that output (144). The blocking character of the communication is taken care of by the use of a binary semaphore `rw` (143). This guarantees that the input thread (139) will block until the previous data has been read by the autonomous thread (138) (no data is overwritten before it has been read), and that the autonomous thread (138) will block until new data is available (no data is read twice). When the input slave thread (139) is blocked, the CMF & Phase Correction process (135) that requested its service via an RPC (141) is also blocked because the RPC (141) will only return after the slave thread (139) has completed. When the autonomous

thread (138) is blocked, there are no RPC requests to the Correlator Noise Estimator process (136), so that process (136) is blocked automatically.

In the case of Blocked-Write, UnBlocked Read communication, the code for the autonomous thread is slightly modified. The thread always sends the value stored in the context, without checking whether it is updated. The same value can be sent more than once, but the thread will never be blocked. The input slave thread is identical to the BWBR case, and will block until the data has been read.

In both cases, locking and unlocking of the context is required to avoid concurrent accesses to the shared variable in the context and, as such, has nothing to do with the blocking character of the communication.

180. EX-1004, 23:50-24:67 (section entitled “Communication Mechanism Selection”).

The hardware interface and the software I/O device driver is generated automatically with SYMPHONY. To generate these interfaces, SYMPHONY analyses the ports of the software process.

For each of these ports, SYMPHONY scans the library of I/O scenarios for an applicable scenario. The user is asked to select the most appropriate scenario amongst the applicable ones. SYMPHONY then combines the selected scenarios into a software I/O device driver and a hardware interface.

181. EX-1004, 25:36-45 (asking user to select I/O scenarios from categories).

182. Rompaey even claims the selection:

31. The design environment as recited in claim 30 further comprising:
means for **selecting I/O scenario models** for the ports of said first
software subsystem;
means for combining the software subsystems of **said selected I/O
scenarios**; and
means for combining the hardware subsystems of **said selected I/O
scenarios**.

EX-1004, claim 31.

183. Thus, Rompaey discloses selecting from categories of components to
implement functionality (including the claimed parameters) with
components including processor models and I/O scenarios. Then a further
selection of specific I/O scenarios is performed. Then a further selection of
the parameters and functionality of the selected I/O scenarios is performed.
Each of these corresponds to the claimed selection of parameters.

184. The third type of action showing the claimed “selecting” and
“receiving” happens during the interactive refinement process. After the
library and the initial design specification are received by CoWare
including, for example the selection of I/O scenarios, the refinement
process continues with additional selections. The designer will select
functionality and adjustments to that functionality, the system will analyze

the selections including running simulations, and then the designer will have the option to continue selecting functionality and changes to existing functionalities including functionalities related to the input and output characteristics of the designed device such as the pager. These changes to the functionality are reflected by selecting the different parameter types.

185. This is shown in, for example, by the description in which “the designer performs an interactive coarse partitioning of the specifications over a user allocated architecture” to “refine” the architecture using additional “tools” such as “component compilers.” EX-1004, 12:11-34. These “[t]ools, which are independent software functions selectively used by the designer, operate on the database to refine the system representation.” EX-1004, 27:4-7. The tools “include[]: allocation tools (see FIG. 5 and related discussion), assignment tools (see FIG. 5 and related discussion), ... and hardware/software interface tools (see FIG. 7 and related discussion on SYMPHONY).” EX-1004, 27:7-19; *see also* 27:20-28:13 (exemplifying how the refinement process receives inputs related to various characteristics of the design system) including protocols, “communication between the processor and its environment,” “I/O device

drivers” and “hardware,” input/output port definitions and implementations such as “FIFO” and “implement[ing] ... communication.”).

186. These tools take selected “inputs,” perform processing, and select additional parameters for each of the inputs – such as selecting / assigning processor types, or communication channels, or software such as device drivers. This is described below for the various types of parameter types.

187. The selections above are from a hierarchal classification database. In each of the three “selecting” actions above, Rompaey’s designer (or CoWare) selects from existing functionality arranged in “libraries” (*e.g.*, databases). The source files, libraries, database, and “classes for each object” are “hierarchical” and classified according to functionality. EX-1004, 26:50-27:3.

188. Rompaey teaches different “classes” of objects defining components such as programmable processors, ASICs, hardware accelerators, micro controllers, communications blocks and memory, and peripherals (DMA, UART...). EX-1004, 26:50-27:3; 4:36-49. Within each component category, Rompaey allows for further selection refinement to a particular subclassification of that component (from many) such as an “ARM6” or

“Cathedral” processor or a “RS-232” communication block or categorized I/O Scenarios as discussed above.

189. As Rompaey notes, different classes of processors provide for different functionality and may be configured by various parameters for different levels of functionality.

The input to the implementation refinement process is a functional specification: a CoWare language encapsulation consisting of a number of process instances (i.e. host language encapsulations), exhibiting both intra-process and inter-process communication behavior. In a first step, allocation is performed. In this step the number and type of processors are selected that will serve as the target for implementing the input specification. After allocating the necessary processor resources, an assignment step is performed. In this step each process instance of the input specification is assigned to one of the allocated processors.

190. EX-1004, 17:8-18. See also EX-1004, Fig. 6 (showing “assign process to processors” and listing different categories of processors at the top). So, for example, there are “programmable processors” (e.g., ARM6) and more application specific processors (e.g., Cathedral). *Id.*, 4:9-14. There are “processors with different protocols” to be interfaced. *Id.*, 5:4-6. See also EX-1004, 5:35-45, 9:21-51, 19:1-28 (identifying different

processors), 23:1-45; 19:56-60 (special “instruction programmed I/O scenarios [with] special instructions for accessing special I/O ports...).

191. Rompaey’s selection uses a “hierarchical” set of different classes of objects and functions to implement different types / categories of device-level functionality. EX-1004, 6:19-34. The objects and functions “are refined into second hierarchical objects for adding implementation detail to the communication behavior...” EX-1004, 7:10-13, 9:10-13. Rompaey’s “data model is hierarchically structured and allows the refinement” of the specified components and functions. EX-1004, 11:31-44. The ‘657 patent used similar “hierarchal” language as Rompaey to describe its selections. EX-1001, Fig. 4, 4:20-23, 6:17-19.

192. In each of the three selecting and “receiving” actions above, a POSITA would understand that Rompaey’s different functions and components are categorized to effectuate Rompaey’s “reuse” goal which is a “key.” EX-1004, 4:64-5:2. “In modular designs, the complete system functionality is split into communicating components of manageable complexity. The advantage of this approach is that the components can be reused....” EX-1004, 5:18-21.

193. To be able to re-use such components, the storage of such components would be most optimally in a form that allowed the designer to readily identify and access those stored designs in the future. Thus, beyond Rompaey’s explicit reference to such categories classification databases, a POSITA would understand that the various aspects of Rompaey that are used in the system specification would be stored in a database or a library arranged by categories and those categories would include various hierarchies of categories. Such hierarchical classification (by category) would optimally allow for such re-use because the arrangement allows for the designer to quickly and consistently identify and re-use the designed components. A POSITA would recognize such a categorized hierarchical classification database as an obvious implementation of Rompaey’s system – particularly with the emphasis on re-use of previously used components because it provides efficiency in storage and in accessing the materials quickly through the classification.

194. Rompaey uses “hierarchical” protocols and systems (categorized) to model the components received by the software tools received as a “functional specification.”

The concept of hierarchical protocols is also useful to model off-the-shelf components (“re-use of designs”).... The input to the implementation refinement process is a functional specification

195. EX-1004, 16:66-17:18. This allows for “off-the-shelf” components performing specific input / output functions for the designer to select. EX-1004, 5:42-67, 9:49-51.

196. Rompaey describes many categories of types of functional components that may be selected for the device functionality including “programmable processors, application specific processors with hardwired controller, application specific processors with specialized instruction set, hardware accelerators, micro controllers, communication blocks and memory, peripherals (DMA, UART, and so forth).” EX-1004, 4:38-49.

197. UARTs are a well-known component to implement functions for display, input and output types for “peripherals” such as displays, buttons and keyboards since at least 1980. For example, as discussed herein CoWare I shows the UART implementing the user interface to devices such as the pager display and control buttons

198. For just one example, U.S. Pat. No 4,430,731 (filed in 1980) notes that: “An UART is also connected to data bus system 19, and as is well known translates data addressed thereto from parallel to serial, with an

asynchronous format, having a start bit, address sequence, data sequence, parity bit, and termination bit. The UART is connected via asynchronous bit 23 to one or more addressable UARTs 22. Various types of keypads, keyboards, displays, etc., are connected to the addressable URATs [sic]....”
EX-1004, 4:36-46.

199. Each form of input (from the selection and receiving) above is received into the claimed “a memory of a processor” This would include that the inputs (the selections) are received into a single memory of a single computer. Rompaey identifies that the entire CoWare tool can be implemented in a single workstation and in a memory. EX-1004, 15:50-61, 26:50-27:23, Fig. 18.

200. [This paragraph is left intentionally blank].

(b) Rompaey’s “Parameter” Types

201. Rompaey teaches the claimed control, display, input, and output parameter types. As the ‘657 patent acknowledges, the standard components that implement the functionality for the devices are “developed for generic characteristics of electronic devices.” EX-1001, 4:46-56. The designer specifies the functionality based on these standard components. EX-1001, 4:46-56. As noted herein, this functionality (the “functions”) is

referenced in the ‘657 as the control type, display type, input type and output type parameters / characteristics / functions. EX-1001, Fig. 2, Fig. 3, 8:28-34, 9:1-4, 9:11-29, 9:50-57, 10:34-39, 14:20-63.

202. I note that characterizing electronic devices by control, inputs, and outputs is not novel or non-obvious and has not been for long prior to the priority date of the ‘657 patent. The “display type” is merely a sub-category of the “output” as the ‘657 states. E.g., EX-1001, 14:61-62 (“the blender’s display references an ‘output’”); 9:22-23 (“The output of the input may also be discerned from the display type.”). These functions (effectively input, output, processing) are the basic computer science / processor functions that every first year computer science class has taught since at least the 1980s.

203. Thus, characterizing an electronic device by these parameter types was not novel or nonobvious as of the priority date.

204. Rompaey describes classes of functional components that may be selected for the device functionality including “programmable processors, application specific processors with hardwired controller, application specific processors with specialized instruction set, hardware accelerators, micro controllers, communication blocks and memory, peripherals (DMA,

UART, and so forth).” EX-1004, 4:38-49. UARTs implement functions for display, input, and outputs for “peripherals” such as displays and keyboards as I have detailed above. See also EX-1004, 6:49-53(“hardware subsystems comprise any one or more of processor cores, off-the-shelf components, custom components, ASICs, processors, and boards.”), claims 7, 37, 43 (same)

205. These lists and the Figures showing the different components exemplify Rompaey’s selecting of parameters for control, display, input and output as claimed.

206. As noted above, the ‘657 describes “parameters” in terms of the selection of components that implement different functionality. Rompaey describes the claimed parameters in at least the following manner.

“Control Type Parameters”

207. Control type parameters include at least the first five components in Rompaey’s list above: programmable processors, application specific processors with hardwired controller, application specific processors with specialized instruction set, hardware accelerators, and micro controllers. EX-1004, 4:37-49. This list is reflected in the “hardware subsystems comprise any one or more of processor cores, off-the-shelf components,

custom components, ASICs, processors, and boards” citations above (e.g., 6:49-53, claims 7, 37, 43).

208. Each of these different component / processor types provides control of the device’s functionality – which is its intended purpose. These all represent different “classes” of processors that can be selected to provide different control. By selecting a particular type of processor which can be used for a specific particular functionality, Rompaey is teaching (including rendering obvious) the selection of a control parameter associated with the device as claimed with the parameter being the processor type.

209. Rompaey gives specific examples of selecting different sub-classes of processors to implement control of different functionality. In the pager example, an “ARM6” “programmable processor” is selected to control certain aspects of the pager whereas two “Cathedral” “application specific DSP processors” are selected to control other aspects as shown in Fig. 10 below:

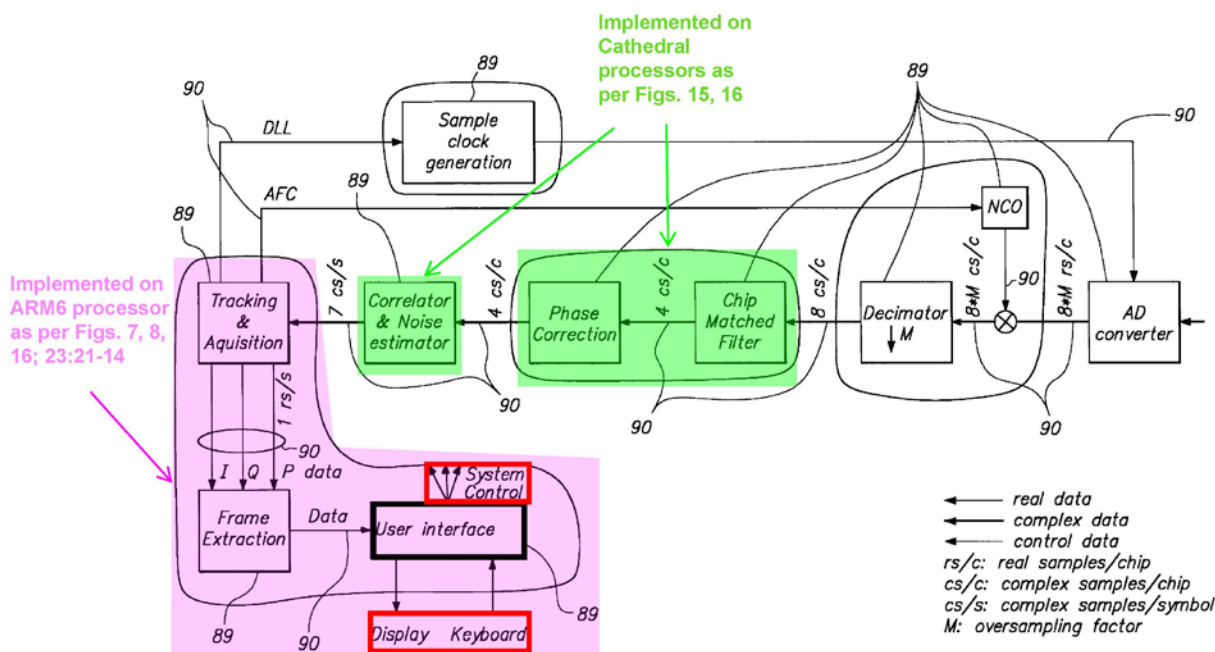


FIG. 10

210. EX-1004, Fig. 10,

Each block (89) in FIG. 10 corresponds to a process implementing a specific function of the pager. This functional decomposition determines the initial partitioning. The lines with arrows (90) in between the processes represent primitive channels with RPC semantics. FIG. 11 shows the RPC communication in detail for part of the pager design. The blocks (91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101) in FIG. 11 correspond to the processes (89) from FIG. 10.

The Sample Clock Generator process (94) contains an autonomous thread (102). This thread runs continuously. It performs an RPC (103) over its input port ip (104) to the Tracking & Acquisition process (93) to obtain a new value for delta. The autonomous thread (102) of the process (94) adds the delta parameter to some internal variable until a threshold is exceeded. In this way it implements a sawtooth function.

When the sawtooth exceeds the (fixed) threshold, an RPC call (105) is issued to the A/D converter process (95). The autonomous thread (102) of the Sample Clock Generator (94) performs an RPC (105) (gives a sample clock tick) every threshold/delta iterations (real clock cycles).

The slave thread clock (106) in the A/D converter process (95) samples the analog input, and sends the result to the Down-conversion process (100) via an RPC call (107). This in turn will activate the Decimation process (99) via an RPC call, etc.

The Correlator Noise Estimator process (98) contains a slave thread (108) associated with port ip (109) to compute the correlation values. This slave thread (108) is activated when the Phase Correction process (97) writes data to the Correlator Noise Estimator process (98) (i.e., when the Phase Correction process (97) performs an RPC (110) to the ip (109) port of the Correlator Noise Estimator process (98)). The slave thread (108) reads in the data and then performs an RPC (111) to the User Interface process (91) to obtain a new value for the parameter par that it requires for computing the correlation values. Finally, the new correlation results are sent to the Tracking Acquisition process (93) via an RPC call (112) on its op port (113).

The slave thread (114) in the Tracking Acquisition process (93) updates the delta value for the sawtooth function implemented by the Sample Clock Generator process (94). It puts the updated value in the context (115), where it is retrieved by the slave thread op (116) which serves RPC requests from the Sample Clock Generator process (94). In this way, the Tracking Acquisition process (93) influences the

frequency of the clock generated by the Sample Clock Generator process (94). This example shows how the context (115) is used for communication between threads inside the same process whereas the RPC mechanism is used for communication between threads in different processes. The locking (117) and unlocking (118) of the context (115) is required to avoid concurrent accesses to the variable delta. The lock (117) in the slave thread op (116) locks the context (115) for read: other threads are still allowed to read from the context (115), but no other thread may write the context (115). The lock (119) in the slave thread ip (114) locks the context (115) for write: no other thread is allowed to write or read the context (115) until it is unlocked again.

Each process is described in the language that is best fit for the characteristics of the function it implements. The data-flow blocks (NCO (101), Down-conversion (100), Decimation (99), Chip Matched Filter (96), Phase Correction (97), Correlator Noise Estimator (98), and Sample Clock Generator (94)) are described in DFL. The control oriented blocks (Tracking Acquisition (93), Frame Extraction (92) and User Interface (91)) are described in C. The code in FIG. 11, is pseudo-code meant for illustration and does not correspond to the actual code.

211. EX-1004, 21:34-22:35. As noted above and in the figures, these different control blocks are allocated to different processor types. Within each control block there are additional parameters for that processor block.

212. Different processors are selected based on functional requirements.

This selection depends on the particular needs of a processing block and a “target” processor (or processors) is selected for each of the different processing. As noted below the “selection” to place the control processing over multiple processors is itself a form of a selection of a control type parameter.

At this moment it is not yet decided what process will be implemented on what kind of target processor nor is it defined how the RPC communication will be refined. However, the choice of the specification language for each process restricts the choice of the component compiler and in that sense partly determines the target processor. Hence, studying possible alternative assignments of a process to a target processor may require the availability of a description of the process in more than one specification language or a clear guess of the best solution.

Allocation and Assignment

This step determines what processes will be implemented on what target processor. The initial specification shows the finest grain partitioning: a process in the initial specification will never be split over several processors. However, it may be worthwhile to combine a number of processes inside a single processor. This is achieved by merging these processes into a single process that can then be mapped on the selected target processor by a host language compiler. Merging of processes is only allowed when the processes are described in the

same specification language. Hence, studying possible alternative mergers may require that for a number of processes (e.g., Correlator Noise Estimator process) a description is available in more than one specification language. After allocation and assignment, one obtains a description with a one-to-one mapping of merged processes to processors.

In the pager example (FIG. 12), the following allocation, merging and assignment takes place. The NCO (120), Down-conversion (121), and Decimation (122) processes are merged and mapped in hardware onto an application specific DSP processor (123) because the sample rate of the merged processes is identical which implies that they can be clocked at the same frequency. The advantage is that only one clock tree needs to be generated per merged process (one per original process). An additional advantage is that the scan-chains for the processes that are merged can be combined.

The Chip Matched Filter (124), and Phase Correction (125) processes are merged and mapped onto a CATHEDRAL-3 processor (126) because their sample rates are identical. The Correlator Noise Estimator process (127) is mapped onto a CATHEDRAL-3 processor (128). It is not merged with the Phase Correction process (125) because it operates at a four time slower frequency. The Sample clock generator (129) is mapped onto a CATHEDRAL-3 processor (130).

Tracking Acquisition (131), Frame Extraction (132), and User Interface (133) are merged and mapped on a programmable processor (134). For this design an ARM6 processor is chosen. The Hardware/Software tradeoffs are based on the following observations.

To obtain a maximal degree of flexibility as much of the functionality as possible is implemented in software on the ARM6 (134). However, due to performance constraints of the ARM6 processor (134), there is a limit to what can be implemented in software. The two main factors that play a role in this problem are:

(1) The Tracking Acquisition process (131) has to be implemented in software because the algorithm used to perform tracking and acquisition may be modified depending on the application domain of the pager system.

(2) The Correlator Noise Estimator process (127) is not included in software because the input rate for the Correlator Noise Estimator (127) is too high to realize a real-time communication between the ARM6 and the Phase Correction process (125). In addition, an estimation of the number of cycles required to execute each function on the ARM6 shows that the implementation of Correlator Noise Estimator process (127) in software leaves insufficient time to perform tracking and acquisition in between every two symbols.

After merging, each of the merged processes can now be implemented on a separate target processor by the appropriate compiler. The communication between the merged processes is still done via primitive ports and channels.

213. EX-1004, 22:41-23:45. As noted above, the additional “selections” for control type include the specification of the interactions between the different processors, as well as the clocking frequencies and the sampling rate for the RF processing.

214. Furthermore, for each processor, additional parameters are selected (by the designer or CoWare) to define the specific flow and control of data processing between the processors.

After merging, each of the merged processes can now be implemented on a separate target processor by the appropriate compiler. The communication between the merged processes is still done via primitive ports and channels.

Communication Mechanism Selection

After the partitioning of the system has been verified by simulation and before the actual implementation takes place, the designer may choose to refine the communication mechanism between the processors. This can be achieved by making explicit the behavior of the channels between the processors.

In the running example, the processors can, in principle, operate concurrently because each processor has its own thread of control. By refining the RPC based communication scheme we can pipeline the processors: all processors operate concurrently and at I/O points they synchronize. This refined communication scheme is called Blocked/UnBlocked Read/Write communication. FIG. 13 shows the pager with the refined communication mechanism. The inputs and outputs of the processors have been labeled with BW for Blocked Write, BR for Blocked Read, and UBR for UnBlocked Read.

BW-BR communication guarantees that no data is ever lost. When the writing process has data available, it will signal that to the reading process. If the reading process is at that moment not ready to receive

the data (because it is still processing the previous data), the writing process will block until the reading process is ready to communicate. Alternatively, if the reading process needs new data, it will signal that to the writing process. If the writing process is at that moment not ready to send the data (because it is still computing the data), the reading process will block until the writing process is ready to communicate. The BW-BR scheme is used in the main signal path.

A BW-BR scheme, however, is not used for the parameter and mode setting for the main signal path. If an accelerator uses BR to read a parameter value, it will be blocked until the parameter is provided. Since the parameter setting is done in software, this will slow down the computations in the main signal path considerably. Therefore parameter setting is done via a BW-UBR scheme. This makes sure that every parameter change is read by the accelerators, but it leaves it up to the accelerator to decide when to read the parameter.

In the CoWare design environment, the refinement of the communication mechanism is performed by making use of a hierarchical channel. A hierarchical channel replaces a primitive channel by a process that describes how communication over that channel is carried out.

The introduction of BW-BR communication is shown in detail for the Chip Matched Filter & Phase Correction (135) and Correlator & Noise Estimator process (136) in FIG. 14.

The BWBR channel (137) contains an autonomous thread (138) and a slave thread (139) that communicate with each other via the shared variable tmp 0..7! in the context (140). The slave thread (139) is

activated by an RPC (141) from the CMF & Phase Correction process (135) and it tries to update the context (140) with new values. The autonomous thread (138) continuously tries to read the values from the context (140) and send them to the output port (144) via an RPC (142), in this way activating the Correlation & Noise Estimator process (136) that is attached to that output (144). The blocking character of the communication is taken care of by the use of a binary semaphore rw (143). This guarantees that the input thread (139) will block until the previous data has been read by the autonomous thread (138) (no data is overwritten before it has been read), and that the autonomous thread (138) will block until new data is available (no data is read twice). When the input slave thread (139) is blocked, the CMF & Phase Correction process (135) that requested its service via an RPC (141) is also blocked because the RPC (141) will only return after the slave thread (139) has completed. When the autonomous thread (138) is blocked, there are no RPC requests to the Correlator Noise Estimator process (136), so that process (136) is blocked automatically.

In the case of Blocked-Write, UnBlocked Read communication, the code for the autonomous thread is slightly modified. The thread always sends the value stored in the context, without checking whether it is updated. The same value can be sent more than once, but the thread will never be blocked. The input slave thread is identical to the BWBR case, and will block until the data has been read.

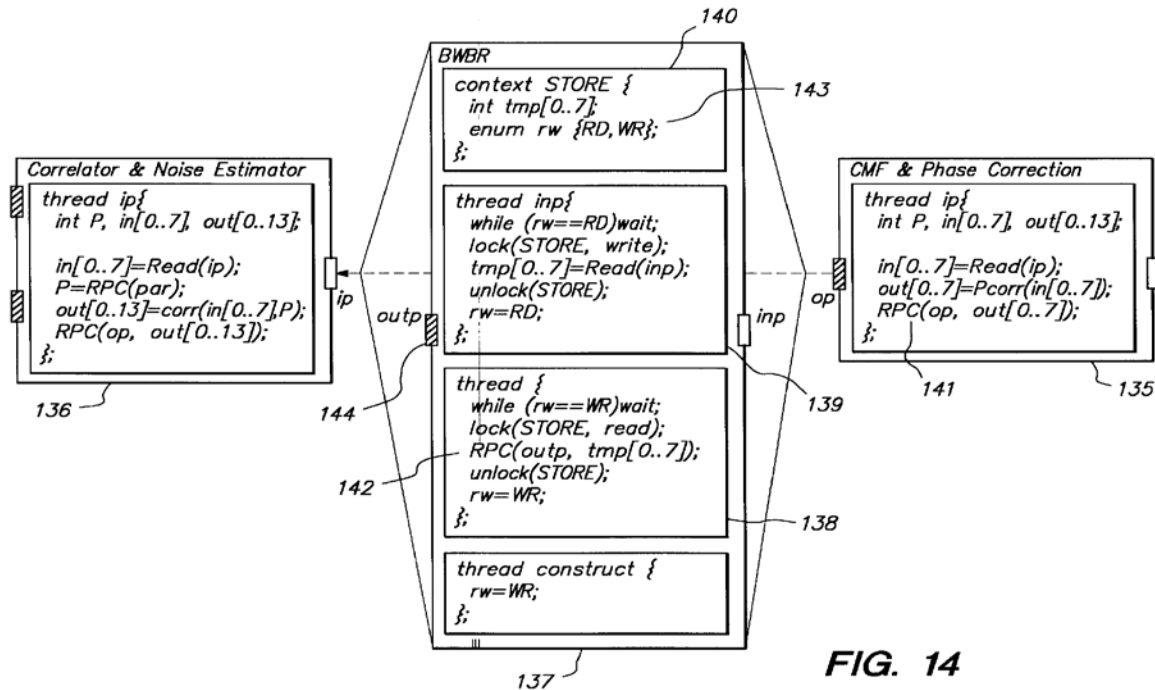
In both cases, locking and unlocking of the context is required to avoid concurrent accesses to the shared variable in the context and, as

such, has nothing to do with the blocking character of the communication.

215. EX-1004, 23:46-24:67. As shown above, these additional parameters for the various processors include the communications such as the BW-BR communication scheme from the Blocked/UnBlocked Read/Write schemes and the allocation of processors to particular functionality along with particular inputs and outputs for that functionality.

216. As a further example, different processors may have a selected control parameter of different clock frequencies depending on the implemented functionality. EX-1004, 23:12-19 (“The Chip Matched Filter (124), and Phase Correction (125) processes are merged and mapped onto a CATHEDRAL-3 processor (126) because their sample rates are identical. The Correlator Noise Estimator process (127) is mapped onto a CATHEDRAL-3 processor (128). It is not merged with the Phase Correction process (125) because it operates at a four time slower frequency. The Sample clock generator (129) is mapped onto a CATHEDRAL-3 processor (130).”). This teaches, including rendering obvious, the control type parameter selection.

217. Figures 14-16 describe the additional details of the parameters selected for the Cathedral processors to implement the functionality for which they were selected. EX-1004, 24:27-59, 25:1-26:30.

**FIG. 14**

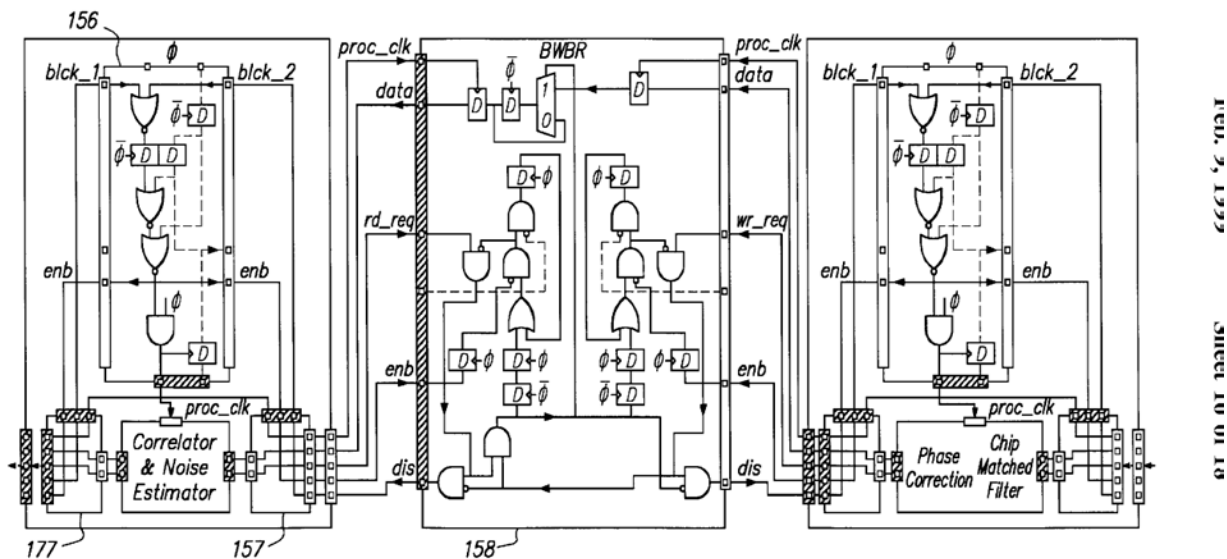
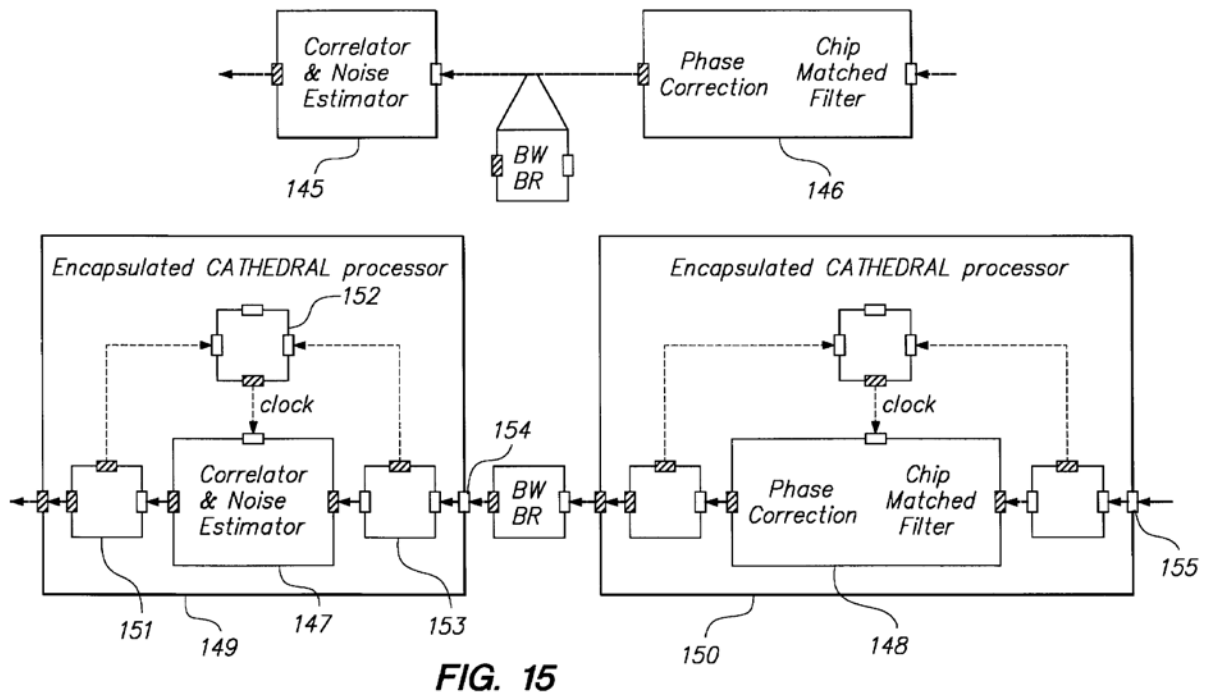


FIG. 16

218. Figures 14-16 show the different communications and control parameters specified to be exchanged across the different blocks. The

specification and selection of these parameters (for example, the type of data block and the allocation of functionalities across the blocks) is a form of selection of the control type parameter because it controls the functionality and operation of the overall device and is specific to that device (in this case, specific functionality for a pager device) implemented in various processor types as discussed above.

219. Also, the description of the types of selections (above) includes Figs. 7-8 (and their associated descriptions) that exemplify the selection of the ARM6 processor type and the associated parameters within that processor type to implement control of the device. The ARM6 is shown as controlling various aspects including the user interface, Frame Extraction, and Tracking and Acquisition blocks. As shown in Figure 10, these blocks exchange various control parameters and the specification and allocation of functionality to the processor shows the selection of control type parameters.

“Display Type Parameters”

220. Rompaey’s pager example show selecting the functionality of having a “display” which is a form of the claimed selecting of display type parameters associated with the device. Merely selecting that the device has

a display is apparently one of the few types of display type parameters disclosed in the ‘657 patent.

221. Selecting the type of display and the specific implementation are also display type parameters as claimed. For example, in 1995, a pager would typically have had an alphanumeric display using an LCD – both forms of parameter types as claimed. The selection of such a display would be understood to be a part of Rompaey in light of the repeated showings of the associated display and would be, at a minimum, obvious to a POSITA in light of the express notations in the Figures of a display.

222. Rompaey’s selectable components include “communication blocks and memory, peripherals (DMA, UART, and so forth).” EX-1004, 4:38-49. Rompaey’s disclosed “display” would be implemented as a selected “peripheral” component using, for example, the UART or RS232 or similar components well-known to implement display (none of which are actually disclosed in the ‘657 patent). A POSITA would understand that the selection of the display type would result in further selections of parameters regarding that display such as update frequency or allocation for the size of information to be displayed on the screen including the display resolution and the type of information (word or graphics) displayed.

223. The “display” interfaces with the User Interface block. EX-1004, Figs. 10-13. Rompaey discloses both “UART” and RS-232 (as also discussed herein) which are forms of interfaces that a POSITA would have known to implement the functionality required for the disclosed “display.” As noted below for the CoWare I, it expressly confirms that a POSITA would understand the display to interface with the UART disclosed as one of the selectable types of peripheral components (parameters). Moreover, the specific protocols for the peripherals would also be selected from the hierarchal classification database as is shown herein in the discussion of the RS232 port that Rompaey shows and describes in Figures 3 and 4 (also discussed herein). The selection of the various parameters for the type of peripheral port (such as UART or RS232) including the protocols for data exchange, the direction of the data exchange, the timing / frequency of the data exchange – all of the types of parameters discussed with respect to the RS232 example given here. These examples in Rompaey provide far more detail than any display type parameter discussed in the ‘657 patent.

“Input / Output Type Parameters”

224. The selecting of input and output type parameters is largely detailed in the section describing the “selecting” actions above. The description above

includes: (1) selecting I/O scenarios, selecting the type of I/O scenario and selecting aspects / functionality of each I/O scenario and (2) the extensive details above for Figs. 7 and 8 describing the ARM6 functionality to implement the user interface for input and outputs. *See also* EX-1004, 8:5-38; 12:36-40; 16:50-17:7; 20:10-14; 23:50-24:67 (section entitled “Communication Mechanism Selection”); 25:36-45 (asking designer to select I/O scenarios from categories); claim 31. EX-1003, ¶¶173-182. Each is “selecting” the input and output type parameters (including as appropriate the display type parameter as one of the outputs). A designer or CoWare selects the I/O scenario and then refines that with further selections on parameters such as the direction of the input/output, the protocols and timing of signals. Each is the claimed selection.

225. I also note that the ‘657 patent identifies “display” type parameters (discussed immediately above) as also being “output” type parameters. The ‘657 patent teaches that “input and output of the device may be audio, video, etc. ... the blender's display references an ‘output’” EX-1001, 14:59-61; EX-1001, 9:22-24. Thus, each “display type parameter” immediately above also corresponds to an “output type parameter.”

226. Moreover, because many devices are known to have multiple different forms of inputs and outputs (e.g., such as the display, keyboard, control buttons... on a typical wireless device such as a pager or cell phone), Rompaey's disclosure would be understood by a POSITA to be flexible enough to encompass a single devices with these multiple different input and output type parameters.

227. Using Fig. 4, Rompaey teaches selecting the parameters of an "RS232" port. EX-1004, 14:58-15:49. RS-232 is a well-known input / output component as discussed above.

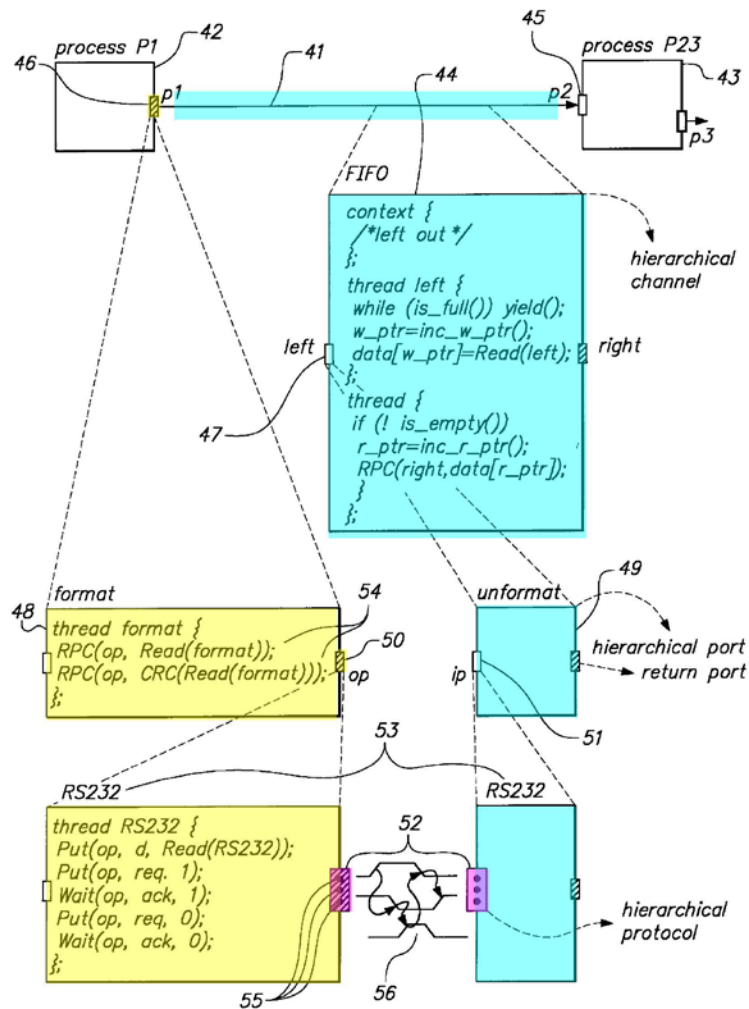


FIG. 4

228. EX-1004, Fig. 4. Figure 4 shows the hierarchy of the classified data model in which a process (42) includes an **input/output port** (p1) associated with a “**FIFO hierarchical channel (44)**” which selects sub-parameters such as the “protocol,” “data format, data direction and “data rate” implemented through “terminals ... according to a timing diagram.” EX-1004, 15:16-49; *see also* EX-1004, 9:64-67. “Primitive protocols provide a classification of all hierarchical protocols.” EX-1004, 15:29-31.

“A primitive port is characterized by a protocol and a data type parameter.”

EX-1004, 12:58-59; *see also* EX-1004, 15:63-67.

229. The specific protocols and data type parameter are selected from a “classification” library (hierarchical classification database) for each of the I/O scenarios such as the RS-232 port discussed herein.

230. Thus, the designer has selected the claimed input and output parameters by selecting the port (RS-232) expressed through the “terminals” that perform the input and output according to the selected parameters for that RS-232 (such as protocol, timing, direction, and data type). As RS-232 ports implement input and output components such as displays, keyboards and controls such as control buttons on a pager, such designer specifications also are the input / output parameters of the device. EX-1003, ¶¶135-148. *See also* EX-1004, Fig. 1 (element 5); 3:19-25.

231. Furthermore, the “selection” of a particular processor type may also correspond to selecting an input type parameter because it may define the type of inputs that may be processed. For example, Rompaey specifically describes that a particular “application specific DSP processor” is selected for processing the input wireless signals received into the pager. EX-1004, 24:1-19, 21:43-22:34. *See also* EX-1004, 3:51-67 (describing functionality

to process “incoming signals”). This selection can be performed by the designer or CoWare. The types of functions such as “down-conversion” and “decimation” relate to “frequency down- or up-conversion, bit to symbol modulation, data-compress and error correction encoding” (EX-1004, 3:28-32) which all relate to input data signals. Selecting these functionalities and a processor type that corresponds to these functionalities is the claim selection of input type parameters. These are shown in Figures 10-17 which each demonstrate the type of parameters that would be specified to process the incoming and outgoing radio signals (input parameter types and output parameter types).

232. For example, as shown in Fig. 10 below, the “pager” example is receiving wireless transmissions which are shown as being received into the AD converter. Those input transmissions are then passed to various portions of the designed system as shown in the arrows (using the legend) showing the “sampling” of the data and the “chip” information. For each of these components, specific operating “parameters” would be selected by the designer so that the system would operate as desired. For example, the designer would specify operating parameters for the “rs/c”, “cs/c”, “cs/s” and “M” parameters specified in the legend and shown as being passed

between the functional blocks. Such parameters are “input” parameter types because they define the operation on the inputs to the device (the wireless signals). Similarly, for a two-way device (that transmit), a POSITA would understand that effectively the opposite procedure would be done to convert the transmitted data into the output radio signals. The designer would be specifying output type parameters for that opposite procedure.

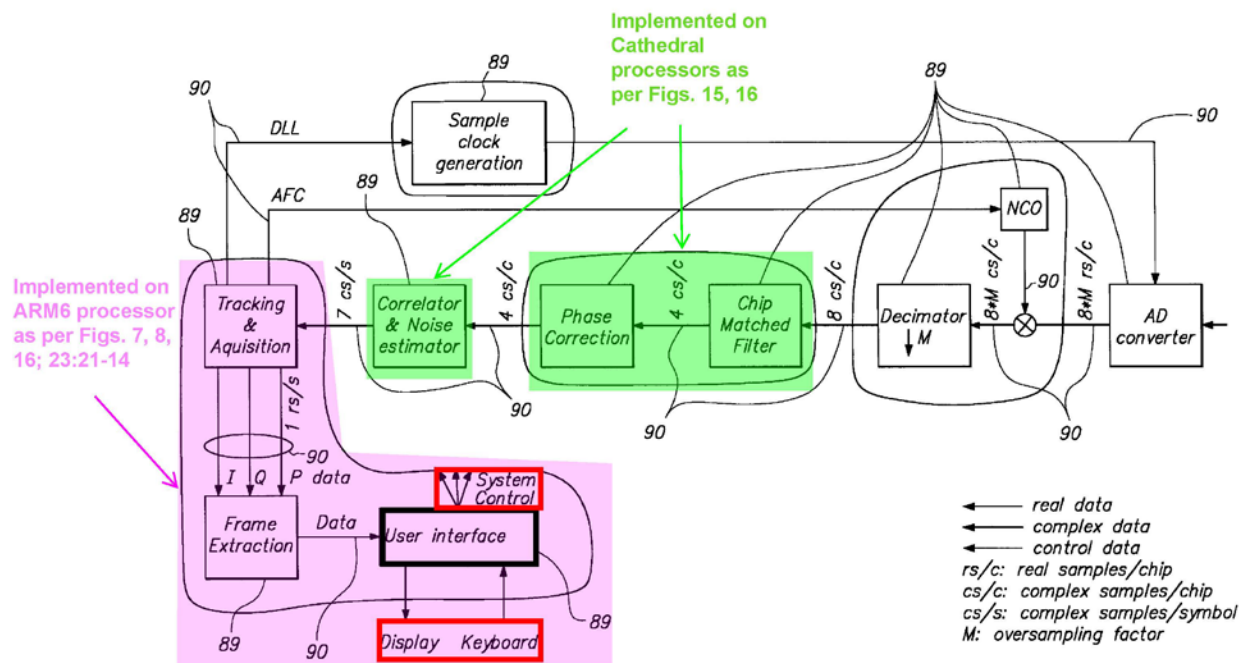


FIG. 10

233. [This paragraph is left intentionally blank]

(c) Rompaey Renders This Element Obvious Alone Or In
Combination With CoWare I.

234. If the Patent Owner argues that Rompaey does not expressly disclose any aspect of this element, Rompaey renders obvious every aspect of this element.

235. For example, if the Patent Owner argues that Rompaey discloses receiving characteristics of a hardware integrated circuit (as opposed to a device containing that circuit), Rompaey at a minimum renders obvious receiving the device-level characteristics.

236. For example, at a minimum, Rompaey discloses the type of selection that the Patent Owner asserts corresponds to the claims. For further example, the Patent Owner asserts that the claimed selection and receipt of parameters is met when a designer:

selects and specifies the configuration options and parameters for the integrated circuit's hardware components and operating software, including options and parameters for the integrated circuit that correspond to those of the electronic devices into which the integrated circuits will be integrated, such as a control type parameter (e.g., system control signals), a display type parameter (e.g., multimedia or graphics aspects), an input type parameter (e.g., memory and other

system inputs), and an output type parameter (e.g., system outputs) for an integrated circuit design.

237. EX-1008, 00025.

238. It is not clear what “system control signals” or “system outputs” references as these as seemingly just vague open ended statements.

239. However, to the extent PO’s assertions are understandable, as detailed above, Rompaey discloses such configuration and attributes of both the designed integrated circuits and the associated device (pager) that the circuits (and software) controls including their “subsystems” and “chips” of those devices. Rompaey teaches in detail the method in which the designed components (including subsystem) comprise the functionality of the “system” (device) being designed.

240. Second, Rompaey discloses (and claims) specifying parameters related to the functionality of “application specific integrated circuits” (and programmable processors). EX-1004, 4:36-49, claim 37. The purpose of “application specific” integrated circuits (ASICs) is exactly as the name states – to be specific to a particular type of application. The type of application is the type of device and defines the parameters for, or associated with, that device. *E.g.*, EX-1004, 1:18-45 (discussing applications / devices such as “consumer electronics, portable and personal

communication, broadband networks, multi-media, and automotive applications.”). Rompaey’s list of “devices” is similar to that identified in the ‘657 patent. EX-1001, 8:62-64. For example, the “I/O scenarios” implement specific components that are used as the display, inputs and outputs of the controlled device as discussed above. Similarly, as discussed above, the various processors exhibit different types of “control” that are specific to the applications (e.g., as discussed above, the Cathedral processor being used for certain functionalities related to signal processing and the ARM6 processor being used for other functionalities).

241. As detailed in Rompaey, the design is intended to be tailored to the “heterogeneous digital system” which, as I have noted, Rompaey expressly associates with the “pager” and also calls an “application” which is a device as used in the ‘657 patent. Thus, Rompaey’s disclosure of the pager as an “application” and the design of the “application specific integrated circuit” shows that Rompaey’s disclosure expressly teaches, and renders obvious, the teaching of a device (pager) specific integrated circuit which has control, display, input and output type parameters and functionality of the device-level (“application” or level or “system” level as used in Rompaey).

242. Third, Rompaey expressly labels his pager “device” to be a “pager application.” EX-1004, 10:16-41, 21:29-31. Given these factors, there is no patentable distinction between receiving the parameters of Rompaey’s pager “application” / device and its associated “application specific integrated circuits,” and any receiving of “device” parameters.

243. Similarly, if PO argues that Rompaey does not disclose selections from a hierarchal classification database, Rompaey renders obvious such a database. As noted above, Rompaey stores components and the related parameters and functionality in a library (database) arranged in a fashion to be re-used. Such re-use would require classification and a hierarchal classification sorted by functionality (e.g., processors, I/O) is an obvious design choice. Thus, at a minimum, Rompaey’s “hierarchical” classification renders obvious the claimed selecting and receiving of parameters.

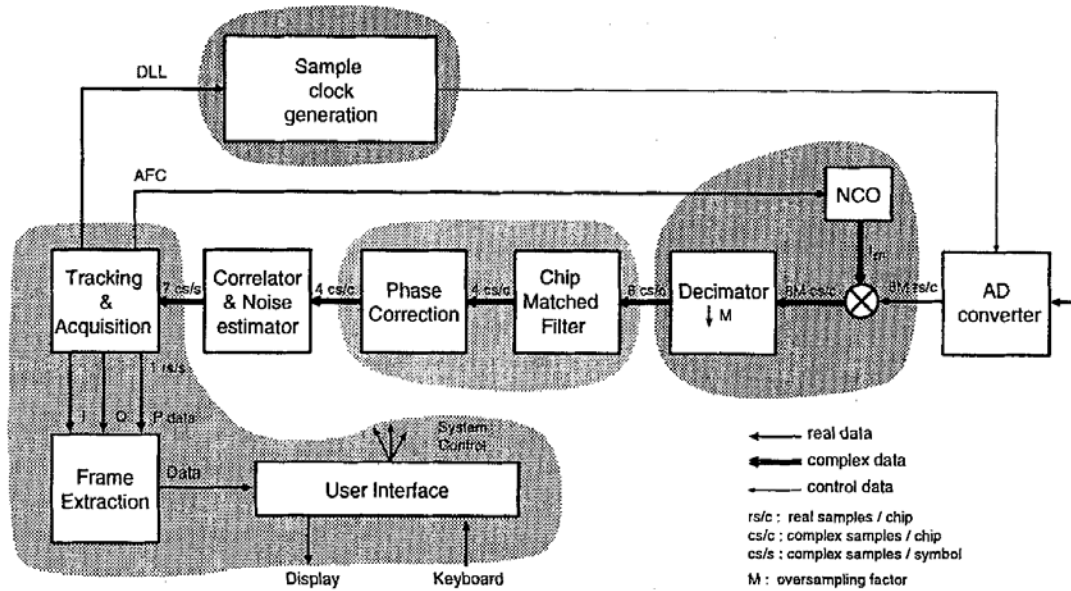
244. Furthermore, as discussed above, Rompaey teaches the ability to specify specific different types (classes) of components such as processors, ASICs, hardware accelerators, micro controllers, communications blocks and memory, and peripherals (DMA, UART...) from various databases. EX-1004, 4:36-49. Within each component category, Rompaey allows for

further selection refinement to a particular type of that component (from many) such as an “ARM6” or “Cathedral” processor or a “RS-232” communication block or categorized I/O Scenarios. Such refinements render obvious the selection and receiving of the different parameters from a hierarchal classification database. Finally, Rompaey’s system for inputting data uses a “hierarchical” set of different types (classes) of objects and functions to implement different types / categories of device-level functionality. EX-1004, 6:19-34. The objects and functions “are refined into second hierarchical objects for adding implementation detail to the communication behavior...” EX-1004, 7:10-13, 9:10-13. Rompaey’s “data model is hierarchically structured and allows the refinement” of the specified components and functions. EX-1004, 11:31-44. The ‘657 patent used similar “hierarchal” language as Rompaey (seventeen years after Rompaey) to describe its “classified” inputs. EX-1004, Fig. 4, 4:20-24, 6:16-18, 7:51-53, 8:44-46, 9:42-48. Thus, at a minimum, Rompaey’s “hierarchical” classification renders obvious the claimed “receiving” of the “selected” parameters from a hierarchal classification database.

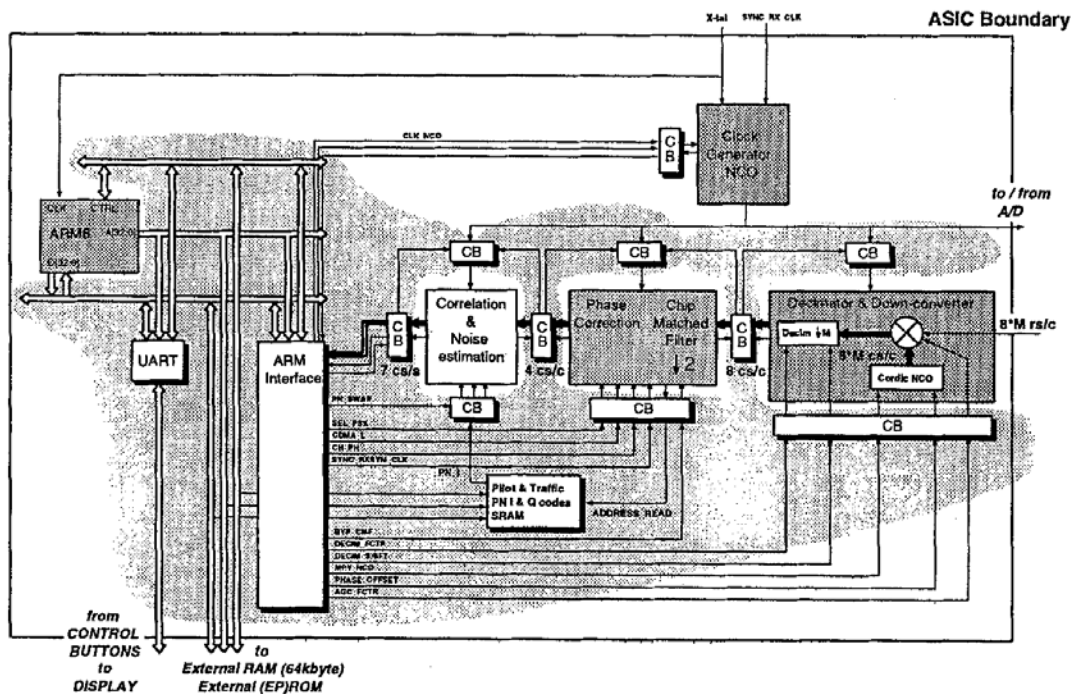
245. Thus, Rompaey discloses this element.

246. Rompaey combined with CoWare I further discloses this element.

CoWare I is a published paper that largely mirrors much of Rompaey's
figures and disclosure. CoWare I includes the following Figure:



(a)



(b)

247. EX-1007, Figs. 6(a), 6(b). Figure 6(a) is a copy of Rompaey's Fig. 10.

Fig. 6(b) adds relevant disclosure. First, Fig. 6(b) shows that the pager has

“Control Buttons” that are part of the input / output for the device. This control buttons are essentially identical to the “buttons” identified in the ‘657 patent as corresponding to input parameter types including their operation and specification of the components implementing this functionality as detailed above. Moreover, these buttons correspond to the “keyboard” disclosed in Rompaey (and Figure (a) above) and/or the “system controls” in Rompaey.

248. Second, Fig. 6(b) shows that the “UART” is a component that is selected for the display, input, and output type parameters (display and buttons). Rompaey discloses a “UART” as one of its classified selectable functionalities for peripherals as I have discussed above. EX-1004, 4:38-49. This UART shows a specific example of how Rompaey shows the selection of the user interface, to implement the interaction with the device components such as the control buttons and display to effectuate the device display, input and output functionalities and parameters. This would correspond to, for example, the discussion of selection of “I/O scenarios” above.

249. A POSITA would be motivated to combine Rompaey and CoWare I with a reasonable expectation of success. The references show largely the

same content including the description of the generic design process and the specific description of the pager example. The references are authored by the same people (the three inventors on Rompaey.). The references were filed / published contemporaneously with Rompaey's application filed on October 23 1996 and CoWare publishing in 1996. Finally, I note that the inventors expressly submitted CoWare I to the PTO while prosecuting Rompaey as shown on the face of Rompaey:

Verkest et al., "CoWare—A Design Enviroment for Heterogenous Hardware/Software Systems," *Design Automation for Embedded Systems*, 1, pp. 357–386 (1996).

Van Rompaey et al., "CoWare—A design enviroment for heterogenous hardware/software systems," *IEEE*, pp. 252–257 (1996).

250. EX-1004, cover

251. Given these factors and the heavy overlap in content, a POSITA would have been motivated to combine references with the same/similar overlapping content such as these references. For example, a POSITA would have wanted to understand any additional details that might exist in CoWare I beyond the patent disclosure. The color diagrams above are one such example of additional information that would have been useful to, and sought by, a POSITA in viewing Rompaey.

252. Thus, a POSITA would have found that Rompaey, alone or in combination with CoWare I, discloses and/or renders obvious this claim element for the reasons discussed above.

253. [This paragraph is left intentionally blank]

- ii. 1[b]: “analyzing by the processor the selected values for the control type, display type, input type and output type parameters of the electronic device;”

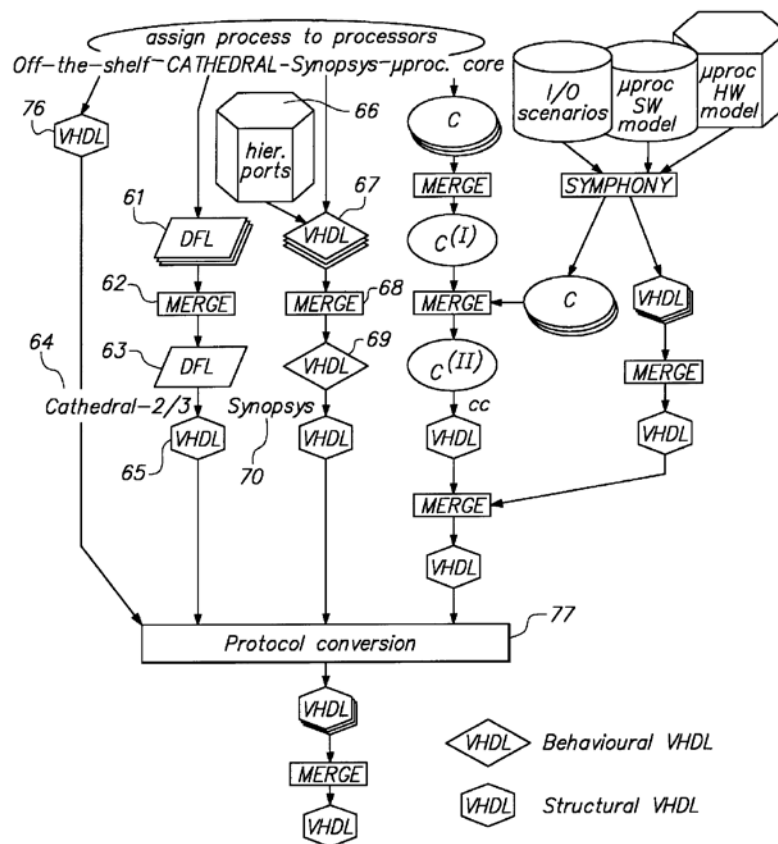
254. Rompaey discloses this element.

255. Once the CoWare designer specifies the system design and libraries for a desired implementation, the CoWare tool analyzes those selections to generate the hardware circuits and software corresponding to the selections. This is shown in the processing of the inputs in Figures 2, 6, 7, 11, 18 (showing the receiving of several different forms of selection of the claimed parameters including selections using libraries, scenarios, and models that include elements 180, 182, 186, and 190 in Figure 18).

256. Fig. 2 (*supra*) shows one example of the claimed analyzing in the blocks that perform “co-synthesis 10” as well as the “SYMPHONY” tool. The co-synthesis analyzes the designer’s design specification, including all of the parameters, and libraries to iteratively and interactively map the design onto hardware and software components. EX-1004, 12:12-43.

257. For example, the designer can select from classified hierarchal databases the various control, display, input and outputs such as by selecting I/O scenarios as described above. CoWare (including SYMPHONY) will then analyze the existing design and potentially simulate the design. The designer could then change, modify, or augment the previous selections for the design to refine the design and CoWare would then perform the same analysis. After the designer is satisfied with the design, a final analysis is performed and the hardware integrated circuit design and associated software are output.

258. Fig. 6 shows another example of the claimed “analyzing.” “FIG. 6 is a flowchart of a specific embodiment of the implementation process for hardware/software co-design according to the method of FIG. 2.” EX-1004, 10:3-5.



259. EX-1004, Fig. 6.

260. The various “merge” and “assign process to processors” blocks are all forms of the claimed “analyzing.”

261. For example, the “assign process to processors” analyzes the system design to determine which processor (or category of processor) might best meet the design goals.

262. Allocation and Assignment

This step determines what processes will be implemented on what target processor. The initial specification shows the finest grain partitioning: a process in the initial specification will never be split over several processors. However, it may be worthwhile to combine a number of processes inside a single processor. This is achieved by merging these processes into a single process that can then be mapped on the selected target processor by a host language compiler. Merging of processes is only allowed when the processes are described in the same specification language. Hence, studying possible alternative mergers may require that for a number of processes (e.g., Correlator Noise Estimator process) a description is available in more than one specification language. After allocation and assignment, one obtains a description with a one-to-one mapping of merged processes to processors.

EX-1004, 22:51-67.

263. As noted above, part of the “analysis is “studying possible alternative” implementations for the process in the design. The specific pager example shows such allocation and assignment:

In the pager example (FIG. 12), the following allocation, merging and assignment takes place. The NCO (120), Down-conversion (121), and Decimation (122) processes are merged and mapped in hardware onto an application specific DSP processor (123) because the sample rate of the merged processes is identical which implies that they can be clocked at the same frequency. The advantage is that only one clock

tree needs to be generated per merged process (one per original process). An additional advantage is that the scan-chains for the processes that are merged can be combined.

The Chip Matched Filter (124), and Phase Correction (125) processes are merged and mapped onto a CATHEDRAL-3 processor (126) because their sample rates are identical. The Correlator Noise Estimator process (127) is mapped onto a CATHEDRAL-3 processor (128). It is not merged with the Phase Correction process (125) because it operates at a four time slower frequency. The Sample clock generator (129) is mapped onto a CATHEDRAL-3 processor (130).

Tracking Acquisition (131), Frame Extraction (132), and User Interface (133) are merged and mapped on a programmable processor (134). For this design an ARM6 processor is chosen. The Hardware/Software tradeoffs are based on the following observations. To obtain a maximal degree of flexibility as much of the functionality as possible is implemented in software on the ARM6 (134). However, due to performance constraints of the ARM6 processor (134), there is a limit to what can be implemented in software. The two main factors that play a role in this problem are:

- (1) The Tracking Acquisition process (131) has to be implemented in software because the algorithm used to perform tracking and acquisition may be modified depending on the application domain of the pager system.
- (2) The Correlator Noise Estimator process (127) is not included in software because the input rate for the Correlator Noise Estimator (127) is too high to realize a real-time communication between the

ARM6 and the Phase Correction process (125). In addition, an estimation of the number of cycles required to execute each function on the ARM6 shows that the implementation of Correlator Noise Estimator process (127) in software leaves insufficient time to perform tracking and acquisition in between every two symbols.

EX-1004, 23:1-45.

264. As described above, Rompaey’s “merge” process analyzes the design and libraries to implement the designer’s specification. EX-1004, 17:19-58; 22:51-24:63 (detailing an example of how processes are allocated to specific hardware components based on various parameters); 25:1-26 (and Fig. 15) (detailing the implementation of portions of the pager).

265. The CoWare tool will then use the allocated processors to “compile” into implementations.

After merging, each of the merged processes can now be implemented on a separate target processor by the appropriate compiler. The communication between the merged processes is still done via primitive ports and channels.

EX-1004, 23:46-49

266. The “SYMPHONY” block also analyzes the selections. As shown in Fig. 6 above, symphony receives the “I/O scenarios,” and the hardware and software models. EX-1004, 25:36-44 (“The hardware interface and the software I/O device driver is generated automatically with SYMPHONY.

To generate these interfaces, SYMPHONY analyses the ports of the software process. For each of these ports, SYMPHONY scans the library of I/O scenarios for an applicable scenario. The user is asked to select the most appropriate scenario amongst the applicable ones. SYMPHONY then combines the selected scenarios into a software I/O device driver and a hardware interface.”). “[T]he implementation of inter-process communication comprises the steps of generating device drivers and of generating hardware interfaces. A software tool is used to achieve this. This tool is called SYMPHONY....” EX-1004, 17:49-63; *see also* 12:37-43 (“CoWare provides in the SYMPHONY toolbox 21 a methodology for interface synthesis whereby every communication channel is refined by selection of a communication scenario. In this way, automated synthesis of hardware/hardware and hardware/software interfaces, including the generation of the software drivers in programmable processors is possible.”). Each of these actions by CoWare is a form of “analyzing” the selections received by CoWare as an operation of the tool to further implement the system design.

267. “FIG. 17 is an illustration of the generation of an exemplary hardware/software interface between a software subsystem compiled on an

ARM processor core and a hardware subsystem.” EX-1004, 10:43-46. The figure shows two ports implemented by the SYMPHONY analysis (shown in Fig. 7). EX-1004, 25:45-26:49.

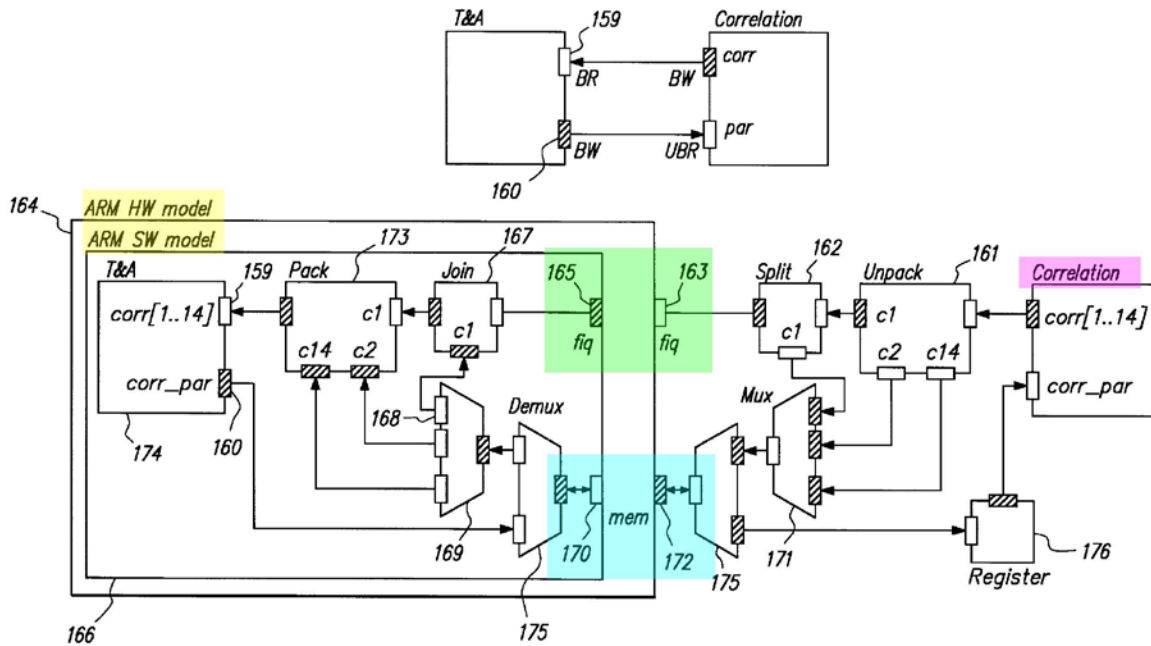
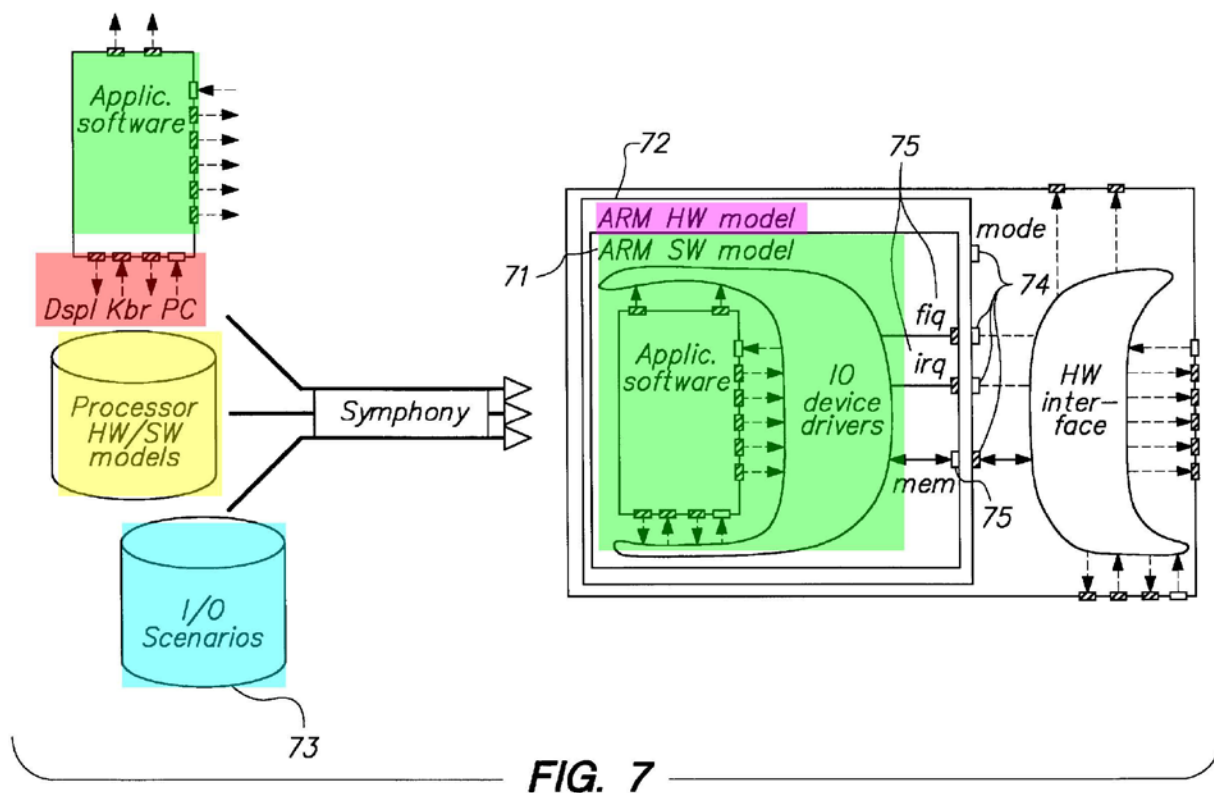


FIG. 17

268. Fig. 17 shows the input/output on two ports (**fiq** and **mem**) to the “**correlator**” which is used for the input / output of the wireless signals of the pager. EX-1004, 21:60-22:35, 25:45-25:49. These ports have various parameters such as protocols, data rates, and data directions as I have noted above.

269. As noted above, Fig. 7 shows an example of SYMPHONY’s analysis of the input selections to create a hardware integrated circuit design (“**ARM HW model**”), “**IO device drivers**” software; and “**Applic[ation]**

software” for the device (pager) that includes the control, display, keyboard, and program controls.



EX-1004, Fig. 7.

270. Rompaey’s claims detail this process of analyzing the selected parameters to output hardware and software for the designed device. EX-1004, claim 1 (analyzing input objects to “generate said implementation of said digital system”); claims 6-8 (implementations in hardware and software), claim 9 (system specification input); claims 10- 25 (describing features of database / libraries); claim 27 (means for specifying a system, means for implementing the system; means for allocating hardware

components for said implementation;...); claim 28 (creating models in database); claim 29 (creating I/O scenarios); claim 30 (implementation in hardware and software for input/output ports); 31-33 (various I/O scenario specification and usage); claims 34-46 (process of specifying characteristics, refining the characteristics, generating hardware and software subsystems implementing those characteristics).

271. If the Patent Owner argues that Rompaey does not expressly disclose this element, Rompaey renders obvious every aspect of this element. The term “analyzing” (or any form of “analysis”) is not in the ‘657 specification and was added without explanation in response to an Office Action in the parent ‘899 prosecution. The ‘657 patent teaches only one form of action on such parameters: “The parameters associated with the designers choices are then input into the FDK engine 65.” EX-1001, 9:57-58. As detailed above, Rompaey contains extensive disclosure regarding linkage of the selected parameters and the ability to link and use those parameters with the libraries of existing ICs for performing the functions associated with the parameters including that the parameters are “input” into the CoWare tool for analysis. Thus, at a minimum, Rompaey teaches the only example of

actions identified in the ‘657 as being performed using “parameters” and therefore renders obvious the claimed “analyzing.”

272. My discussion of “analyzing” above is consistent with that analysis provided by the Patent Owner in its allegations of infringement. This further supports my position that Rompaey renders obvious such analysis at a minimum.

273. PO asserts that the claimed “*analyzing...*” occurs when a designer “uses the [] [d]esign [t]ools to define the desired parameters and configure the selected IP and create... RTL, an IP microarchitecture, and executable files for the integrated circuit appropriate for the associated electronic devices.” EX-1008, 000029. As detailed above, Rompaey discloses such analyzing to create and output the RTL or VHDL defining the IP microarchitecture and the executable files (the application software and device drivers) for the integrated circuit appropriate for the pager.

274. Thus, Rompaey discloses this element.

- iii. 1[c] “outputting by the processor a hardware integrated circuit design based on the selection of input and output characteristics of the electronic device, the hardware integrated circuit including a communication bus for communicating with the electronic device; and”

275. As an initial matter, I note that, the “outputting” terms of claim 1 refer to “the” selection of “characteristics,” but the actual “selecting” in the earlier claim elements is of “parameters” (not “characteristics”).

276. My understanding is that during inter partes procedures, certain defenses such as indefiniteness of a claim are not raised in a petition. Thus, I take no position on whether this different language renders the claims indefinite.

277. Rompaey outputs the generated hardware subsystem (integrated circuit design) in well-known formats including HDL, VHDL, and netlist.

278. VHDL, HDL and netlist are commonly used terms to describe outputs (usually files of some sort) that correspond to hardware integrated circuit designs.

[M]ost specifications are first translated into English and then redesigned in a specific hardware description language such as Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL) or VERILOG for the hardware components and a software description language such as C or assembler for the software components.

EX-1004, 1:54-59.

279. Rompaey’s output of both the hardware integrated circuit design and software are shown throughout the specification and the figures.

FIG. 2 shows the architecture of the CoWare system. The CoWare system supports four major design activities: co-specification 8, co-simulation 9, co-synthesis 10 and interface synthesis 11. The input is a heterogeneous specification of an electronic system, the output 12 may include a net list for prior-art commercial tools for the generation of the implementation layout. This output preferably comprises structural VHDL or Verilog and machine code for the programmable processors

EX-1004, 11:12-20.

280. This passage expressly describes the output of both the hardware integrated circuit design and the software.

281. See also EX-1004, 17:46-48 (“The result is a VHDL net list (65) of the implementation....”);

In step (3) when two processors are not protocol compatible, a protocol conversion process is inserted. In FIG. 6, the VHDL net list representation of the processor (65) that was compiled with Cathedral-2/3 and the off-the-shelf processor (76) (VHDL net list representation) have incompatible protocols. Protocol conversion (77) is required to make them compatible.

EX-1004, 20:32-38

282. The above passage shows that as part of the analysis and output, the CoWare tool can use “off-the-shelf” components from, for example, the libraries discussed above in the “receiving” element 1[a].

FUNCTION: implement communication between two connected ports. The communication implementation can be inserting a FIFO channel, or implementing protocol conversion, or so forth.

INPUT: CoWare encapsulation with a channel between two ports.

OUTPUT: CoWare encapsulation with a hierarchical channel that implements the communication.

dump<language> where <language>={**VERILOG**, **VHDL**, C, DFL, C++}

FUNCTION: extract host language code from host language encapsulation.

INPUT: host language encapsulation.

OUTPUT: host language code.

At decision state 188 the system design determines if the system (or a selected portion of the system) is completely refined or is refined as far as required. If not, a loop comprising states 186, 188 and 190 is repeated until the system is completely refined, as determined at decision state 188, and the system designer continues at state 192. At state 192, the system (or system portion) is extracted from the data base by a host language "dumper". The result of the host language dumper is a source file of the system (as shown in state 194) in the corresponding host language, e.g., **VERILOG**, **VHDL**, C, DFL, or C++. Proceeding to state 196, the system designer can compile the source file using a traditional host language compiler to generate a compiled description of the system (or system portion) as shown in state 198. This compiled description may be, for example, an executable for a computer (for a simulation), a gate level net list (for a

VHDL simulation), or a net list describing the final implementation of the system.

EX-1004, 27:66-28:30.

283. This passage above shows that the output of the system from the analysis includes the various hardware integrated circuit design languages such as VERILOG and VHDL including for the “communication processes.”

284. Rompaey teaches this output throughout the specification. It is an integral part of the CoWare system and Rompaey’s disclosure. See EX-1004, 1:53-59; 4:19-29; 5:14-16, claims 38, 44; Figs. 1-2, 6-7, 9, and 18.

285. The hardware output includes “Register Transfer level” (RTL) code. EX-1004, 26:38-42; 3:35-39. RTL code teaches, or renders obvious, the output of the hardware integrated circuit design.

286. This element adds that the hardware integrated circuit design must include a “communication bus.” Rompaey teaches such a bus to a POSITA. First, Rompaey teaches generating software to be executed on well-known “off-the-shelf” processor models, such as “Cathedral” and “ARM6.” The designs of such well-known processor would disclose, or render obvious, a communications bus because such buses are well-known and pervasive throughout processors. Moreover, SYMPHONY generates the hardware and

software to effectuate the inputs and outputs as detailed in element 1[b] above.

E.g., EX-1004, 19:1-20:25. These scenarios included the use of “data that is residing on the memory data bus of the ARM [processor]”

SYMPHONY is based on the observation that programmable processor share a number of common communication methods to get data in or out of the processors. These communication methods are modeled by I/O scenarios. An I/O scenario describes one way of using the ports of a specific processor core to map a particular port of a software host language encapsulation to an equivalent port in hardware, thereby crossing the processor core boundary while maintaining the communication semantics. FIG. 8 shows an example of an I/O scenario. It consists of a software host-language encapsulation and a hardware host-language encapsulation that describe a software I/O driver and the hardware counterpart, respectively. An I/O scenario is also tagged with some performance figures that will allow the designer or SYMPHONY to make a decision about what I/O scenario to use for which port. The I/O scenario (78) of FIG. 8 shows how an outmaster port psw (79) in software can be mapped to an outmaster port phw (80) in hardware, thereby using the memory port (81) of an ARM-6 RISC processor core. The software process encapsulation P1 (82) represents the software I/O driver and copies data from port psw (79) to a specific memory address 0×08000 via an RPC call (84). The hardware process encapsulation P2 (83) represents the hardware counterpart and checks whether the memory address bus of the ARM (modeled by the

protocol indices of the memory port (81)) equals 0×08000. If this is the case, data that is residing on the **memory data bus of the ARM** is copied to port phw (80) via an RPC call (85).

EX-1004, 19:1-28.

287. This confirms that the disclosed processors would have the communication bus in this claim. The address bus allows for specification of the specific data or instructions to be loaded/stored from/to memory. The data bus transmits the data. Furthermore, “SYMPHONY adds RAM and ROM blocks to store the program code and data”). EX-1004, 20:22-25. The teaching of RAM and ROM to store the software and data teaches, or renders obvious, that the hardware design would including a communication bus to facilitate communications between the processor and such memories. A POSITA would recognize that such memory blocks would be connected by buses to the processor so that the executed software could be communicated to the processor for execution. Moreover, such communication buses are also shown in the various diagrams such as the “data” entries in Figure 16.

288. If the Patent Owner argues that Rompaey does not expressly disclose this element, Rompaey renders obvious this element. As detailed above, Rompaey’s purpose and teachings are devoted to converting the design of an “application” (such as the “pager application”) into the hardware integrated

circuits and software implementing that design. Given the widespread disclosure in Rompaey of the analysis of the system design and the subsequent output of the hardware integrated circuit design in various well-known formats (such as VHDL, HDL, netlist, VERILOG, and RTL), at a minimum, Rompaey renders this element obvious. Moreover, the use of a “communication bus” in a device based on processors and other data processing components (such as memories) is, at a minimum, obvious – particularly in light of the express references to the bus as noted above.

289. Thus, Rompaey discloses this element or renders it obvious.

- iv. 1[d] “outputting by the processor software programming commands to be .executed by the hardware integrated circuit for controlling the electronic device in accordance with the selection of categorized input and output characteristics of the electronic device through the communication bus of the hardware integrated circuit using a communication protocol compatible with the electronic device and the hardware integrated circuit.”

290. Rompaey discloses that CoWare outputs software programming commands to be executed by the hardware circuit.

291. The discussion in element 1[b] above regarding “SYMPHONY” highlights this output of software specific to the designed software. For

example, as noted above, Rompaey's output of both the hardware integrated circuit design and software are shown throughout the specification and the figures.

FIG. 2 shows the architecture of the CoWare system. The CoWare system supports four major design activities: co-specification 8, co-simulation 9, co-synthesis 10 and interface synthesis 11. The input is a heterogeneous specification of an electronic system, the output 12 may include a net list for prior-art commercial tools for the generation of the implementation layout. This output preferably comprises structural VHDL or Verilog and machine code for the programmable processors

EX-1004, 11:12-20.

292. This passage expressly describes the output of "machine code for the programmable processors" which is a form of the claimed software.

FUNCTION: implement communication between two connected ports. The communication implementation can be inserting a FIFO channel, or implementing protocol conversion, or so forth.

INPUT: CoWare encapsulation with a channel between two ports.

OUTPUT: CoWare encapsulation with a hierarchical channel that implements the communication.

dump<language> where <language>={VERILOG, VHDL, C, DFL, C++}

FUNCTION: extract host language code from host language encapsulation.

INPUT: host language encapsulation.

OUTPUT: host language code.

At decision state 188 the system design determines if the system (or a selected portion of the system) is completely refined or is refined as far as required. If not, a loop comprising states 186, 188 and 190 is repeated until the system is completely refined, as determined at decision state 188, and the system designer continues at state 192. At state 192, the system (or system portion) is extracted from the data base by a host language "dumper". The result of the host language dumper is a source file of the system (as shown in state 194) in the corresponding host language, e.g., VERILOG, VHDL, C, DFL, or C++. Proceeding to state 196, the system designer can compile the source file using a traditional host language compiler to generate a compiled description of the system (or system portion) as shown in state 198. This compiled description may be, for example, an executable for a computer (for a simulation), a gate level net list (for a VHDL simulation), or a net list describing the final implementation of the system.

293. EX-1004, 27:66-28:30 (showing the output of C and C++ software).

294. As part of the output, Rompaey's system can store the output software into the memory of the device so that the software is available for execution by the device. *See also* EX-1004, 20:22-25 ("In addition SYMPHONY, adds RAM and ROM blocks to store the program code and data..."). ROM ("read-only memory") is frequently used to store critical software

such as the software that allows the device to initially boot up. As part of this boot up, the ROM frequently stores software such as device drivers and operating systems that permit the device to communicate between processors and the peripherals such as keyboards, buttons and displays.

295. The output software is intended to execute on the designed hardware circuits and control the functionality and characteristics of the processor including the input and output characteristics. For example, as noted above for element 1[a], Fig. 7 shows the output of hardware and the associated “Applic. software” and “IO device drivers” to interface with the display, keyboard, and program controls. *See* EX-1004, 6:45-53 (“the implementation is a heterogeneous implementation comprising hardware subsystems and software subsystems, the software subsystems being executed on one or more of the hardware subsystems. The hardware subsystems comprise any one or more of processor cores, off-the-shelf components, custom components, ASICs, processors, and boards. The software subsystems comprise machine instructions for the hardware subsystems.”). *See also* EX-1004, at Figs. 2, 6-7, 9, 18, 25:29-26:49 (“Implementation of a Process in Software...the Software I/O device driver

is generated automatically with SYMPHONY...”); 8:4-38; 26:14-30;
claims 6-8, 27-37; TITLE; Abstract.

296. Rompaey describes the output of “I/O device driver” software which is the software that controls the display, inputs and outputs such as to keyboards, buttons, and displays. EX-1004, 12:35-43, 17:59-64, 18:31-46, 19:10-28; 26:38-49; 27:61-64. “The I/O device driver processes are merged with the original software application code” controlling the device (*e.g.*, pager). EX-1004, 26:42-43.
297. As detailed in 1[a], 1[b], and 1[c], Rompaey’s CoWare system would design and output application software and I/O device driver software that used a communication protocol compatible with the designed device through the communication bus of the designed circuit based on the selections of characteristics shown in element 1[a]. As noted in 1[c], the software may be stored in RAM or ROM that is connected to the processor by one or more communications buses. The successful execution of the instructions would depend on the instructions being communicated across the bus using a communication protocol compatible with the device (and the processor / memory). Rompaey’s disclosure of executing the software to control the device discloses or renders obvious the use of such a bus and protocols.

298. As detailed above, part of SYMPHONY's function was to determine the protocols for communications between different blocks and ensure compatibility of those protocols. For example, the software stored in RAM or ROM necessarily uses the compatible communication protocols for the bus connecting the memories and processor execution units in order to execute properly.

299. If the Patent Owner argues that Rompaey does not expressly disclose this element, Rompaey renders obvious this element. As detailed above, Rompaey's purpose and teachings are devoted to converting the design of an "application" (such as the "pager application") into the hardware integrated circuits and software executing on the hardware to implement that design.

300. Thus, Rompaey discloses this element.

b) Claim 5: "The method of claim 1, wherein the hardware integrated circuit design is based on at least one predetermined hardware integrated circuit component which is able to control one of the selected values for the control type, display type, input type and output type parameters."

301. Rompaey discloses the use of predetermined hardware integrated circuit components throughout its disclosure of the libraries, databases, and models. As detailed in 1[b] above, Rompaey emphasizes re-use with “off-the-shelf” components.

The concept of hierarchical protocols is also useful to model off-the-shelf components (“re-use of designs”).... The input to the implementation refinement process is a functional specification

302. EX-1004, 16:66-17:18. This allows for “off-the-shelf” components performing specific input / output functions for the designer to select. EX-1004, 5:42-67, 9:49-51.

303. These “off-the-shelf” designs are predetermined hardware integrated circuit components because they have already been generated, are stored in libraries or repositories, and are selected for their known and existing (predetermined) functionality.

304. The off-the-shelf components include many categories of types of functional components that may be selected for the device functionality including “programmable processors, application specific processors with hardwired controller, application specific processors with specialized instruction set, hardware accelerators, micro controllers, communication

blocks and memory, peripherals (DMA, UART, and so forth).” EX-1004, 4:38-49.

305. “Reuse” is a key component for Rompaey:

One of the keys to mastering the complexity of digital system design is the reuse of components. The design process for a digital system must allow the modeling of reusable components and support a design for reuse methodology which allows to design components that are easily reusable.

EX-1004, 4:64-5:2.

306. Furthermore, Rompaey discloses predetermined components such as processors, including the “ARM6” and “CATHEDRAL” processors. EX-1004, Figs. 6-8, 15, 17; 23:12-45, 20:32-38; 25:14-15. See also EX-1004, 26:50-27:3 (“... This initial specification is parsed, in conjunction with libraries of **predefined routines 182**, to result in a database of objects which is stored in the computer memory as shown at states 184, 186. The CoWare libraries contain processor models, I/O scenario models, protocol definitions, process definitions for use as hierarchical port or hierarchical channel, and so forth.”). These “models” each are predetermined hardware integrated circuit designs corresponding to this claim.

307. The “I/O scenario models” (see Figs. 4, 8) are also predetermined hardware integrated circuit components controlling one of the control,

display input or output type parameters as detailed above. They are stored in the repositories / libraries and selected to implement specific functionality.

c) **Claim 6: “The method of claim 5, wherein the hardware integrated circuit design is provided as a software schematic, layout, or both.”**

308. As detailed for element 1[c], CoWare generates hardware integrated circuits in formats including HDL, VHDL, VERILOG, netlist, and RTL which discloses this element. To the extent that multiple different forms of concurrent output are required, Rompaey discloses or renders obvious the multiple forms. Rompaey describes many different options for outputting the hardware integrated circuit design and the selection of the specific format or formats for outputs is a design choice that a POSITA would find obvious. The designer would be selecting from a known and confined small set of different options identified expressly in Rompaey.

309. Rompaey expressly recites the generation of a netlist. E.g., EX-1004, 9:19-21 (“generating a net list comprising the layout information of the implementation”); 9:44-51; 11:15-20; 12:48-50; claims 38, 46. To the extent that a “software schematic” is needed, the disclosure of HDL,

VHDL, RTL, and VERILOG teach, or render obvious such a software schematic because the purpose / design of those formats is intended to provide the requisite information for a schematic of the layout of the circuits on hardware (such as silicon wafers).

310. [This paragraph left intentionally blank].

**d) Claim 8: “The method of claim 1, wherein the
hierarchal classification database is a component of
a fabless development kit.”**

311. Element 1[a] details CoWare and the database(s) used by it (including the libraries and repositories). CoWare is part of a “fabless development kit” based on the specific language used in the ‘657 to describe “fabless.”

Fabless design of ICs and MCUs is well known in the art. In general, the term fabless design is applied to process where the design of the IC is accomplished separately from the manufacture of the device itself.

312. EX-1001, 10:9-13.

313. As an initial matter, this claim recites an element (fabless development kit) that the ‘657 patent admits is “well known in the art.” Given that it is well-known, this “fabless” aspect does not impart any novelty or non-obviousness.

314. Rompaey does not link the usage of CoWare to any particular manufacture of the device. Rather, the output of CoWare are materials that could be sent to a manufacture, but there is no linkage. Thus, Rompaey describes a design of the IC that is accomplished separately from the manufacture of the device itself.

315. Rompaey does not mention the words foundry, manufacturer, or fab in its disclosure. There is nothing within Rompaey that would indicate that it is part of a manufacturing process.

316. Rather, as noted in Figure 18, the final output of CoWare is a “compiled description of the system” element 198. “This compiled description may be, for example, an executable for a computer (for a simulation), a gate level net list (for a VHDL simulation), or a net list describing the final implementation of the system.” EX-1004, 28:27-29. This shows that the design is accomplished separately from the manufacture given the wholesale lack of any reference to a specific manufacturer.

317. Given that “fabless design ... is well known in the art,” the decision to use CoWare either separate or integrated with a specific foundry is, at most, an obvious design choice. This is particularly true because different

foundries may be selected. Thus, at a minimum, Rompaey renders this element obvious.

e) **Claim 9: Preamble “A system comprising:”**

318. Rompaey discloses this system. See 1[Preamble]. As I have noted below, Rompaey discloses, including rendering obvious, each of the elements recited for this system.

- i. 9[a] “one or more processors; a design user interface provided by programming commands executed by the one or more processors, the design user interface allowing a user to characterize an electronic device by selecting a value for a control type parameter, a value for a display type parameter, a value for an input type parameter and a value for an output type parameter, all of which are associated with the operation and control of the electronic device;”

319. Rompaey’s CoWare design environment is implemented on a computer with executing software, thus disclosing, including rendering obvious, a processor.

The CoWare model is implemented on a computer or on a plurality of computers and a set of application programmer’s interface (API) functions is available. ... the tools are executed on a Hewlett Packard 715/50 workstation, a SUN, a DEC or other commercially available workstation, or a personal computer or another computer...

320. EX-1004, 15:50-62.

321. The disclosure that CoWare is software that executes on computers like workstations teaches a POSITA that the computers and workstations would have a processor that executes the software. Computers and workstations are controlled by processors executing software. The purpose of software is to be executed ultimately on a processor.

322. See EX-1004, 7:32-8:3, 26:50-27:26; Fig. 18; claims 1-33 (all claiming a tool in a “computer environment”). A computing environment would teach, or render obvious, that the software tools would be executed on one or more processors existing in that environment.

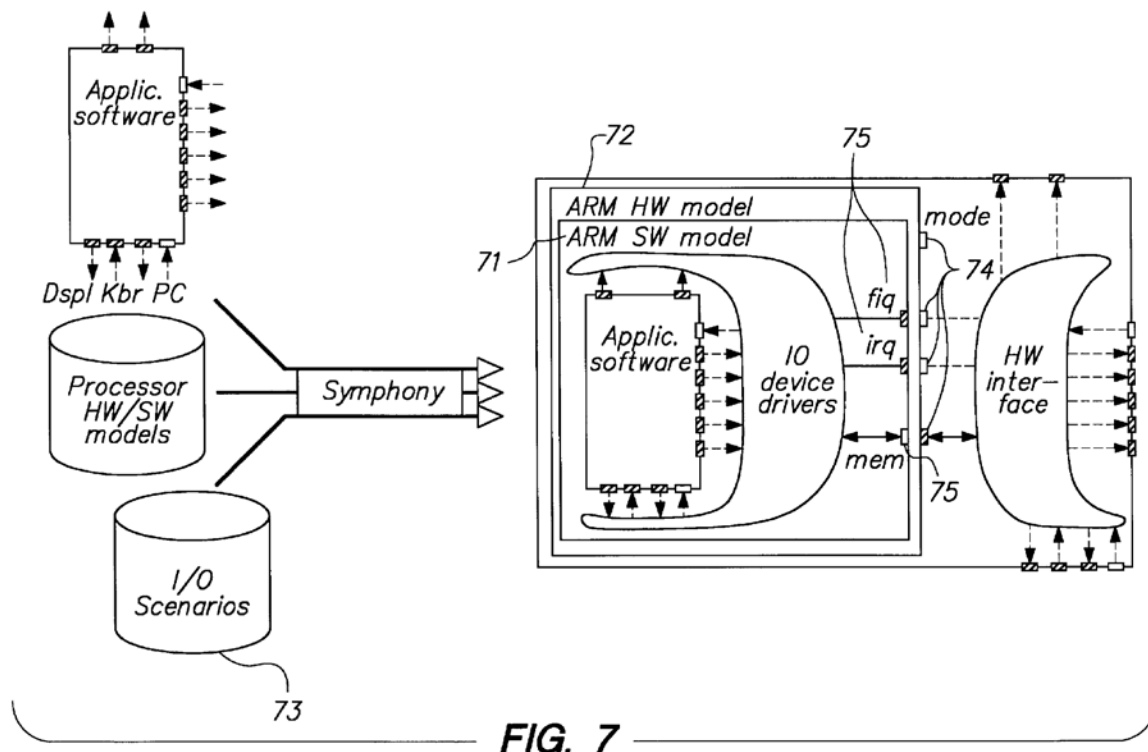
323. Element 1[a] above discusses the selection of the various parameter types – all of which characterize the electronic device and are associated with the operation and control of the device as discussed in element 1[a].

324. I note that PO’s infringement contentions did not identify any additional requirement for the language of “characterize an electronic device” and “are associated with the operation and control of the device” beyond the same content cited for element 1[a] in those contentions. EX-1008, 000036-37.

- ii. 9[b] “logic encoded in one or more non-transitory computer-readable media that, when executed by the one or more processors, is operable to.”

325. As detailed above in 1[preamble], 1[a], 8[a], Rompaey's CoWare design tool was software that executed on processors preferably in workstations or other computers.

326. Rompaey uses the well-known symbol for disk drives associated with its workstation executing the software which is shown below as containing the Processor HW/SW models and the I/O Scenarios.



327. EX-1004, Fig. 7. See also EX-1004, Figs. 2, 6, 18.

328. A POSITA would have understood that the software for the workstations would have been stored on non-transitory media such that the software could be used between instances of the workstation being powered

off. The presence of disk drives on the workstation provides the mechanism for storing such software.

329. Moreover, Rompaey’s claims recite a “database” in a “computing environment.” *E.g.*, claim 1. A POSITA would understand that a “computing environment” would disclose the necessary non-transitory memory (such as disk drives) to store the software to be executed on that computing environment. In particular, in 1996 when Rompaey was filed, such computing environments typically relied on such local storage (disk drives) for the software and data, including databases, used on the workstation or computer.

330. This discloses, or renders obvious, that Rompaey’s CoWare software design tool would have been encoded in one or more non-transitory computer-readable media that is executed by processors.

iii. 9[c]: “receive the selected control type, display type, input type and output type parameter values selected by the user”

331. As I have detailed in 1[a], Rompaey’s CoWare software system receives selections of the parameters as claimed in this element.

332. I incorporate my analysis above for this element.

iv. 9[d] “into a memory associated with the one or more processors, the logic executed by the one or

more processors operable to design a hardware integrated circuit design based on the selection of input and output characteristics of the device, the hardware integrated circuit design including a communication bus, and”

333. As I have detailed in 1[a], 1[b], and 1[c] above, Rompaey’s CoWare designs and outputs hardware integrated circuits based on the selections of characteristics shown in element 1[a]. *See also* 1[b], 1[c], 5 above describing the design and creation of the hardware IC design.

334. I have also detailed above in elements 1[a], 1[b] the reasons that those selections are received “into a memory” as claimed. I incorporate by reference that analysis.

335. This element adds that the hardware integrated circuit design must include a “communication bus.” I discussed this requirement in element 1[a] above.

- v. 9[e] “output software programming commands to be executed by the hardware integrated circuit for controlling the device in accordance with the selected control type, display type, input type and output type parameter values selected by the user using a communication protocol compatible with the device and the hardware integrated circuit through the communication bus of the hardware integrated circuit.”

336. As detailed in 1[a], 1[b], 1[c], 1[d] and 9[d] above, Rompaey's CoWare system would design and output application software and I/O device driver software that used a communication protocol compatible with the designed device through the communication bus of the designed circuit based on the selections of characteristics shown in element 1[a]. As noted above and in 1[d], the software may be stored in RAM or ROM that is connected to the processor by one or more communications buses.

337. Thus, Rompaey discloses this element.

f) Claim 15: “The system of claim 9, wherein the one or more non-transitory computer-readable media further comprises logic that, when executed by the one or more processors, is further operable to simulate the hardware integrated circuit's functionality based on the hardware integrated circuit design”

338. As detailed in 1[b] above, Rompaey discloses simulating the hardware design.

339. Rompaey's use of simulation of the circuitry is shown throughout Rompaey's disclosure. E.g., Abstract (“A hardware and software co-design

environment and design methodology based on a data-model that allows one to specify, **simulate**, and synthesize heterogeneous hardware and software architectures...”).

340. Figure 9 depicts simulation at phase (4) which includes VHDL (hardware integrated circuit design) simulation:

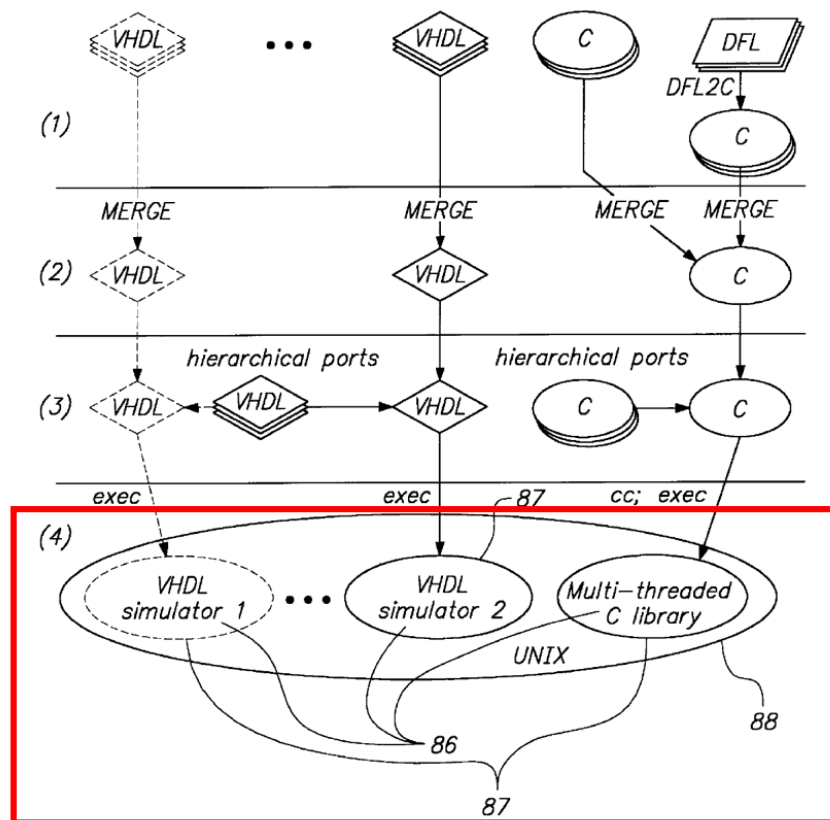


FIG. 9

341. EX-1004, Fig. 9;

342. Rompaey’s disclosure contains pervasive description of the simulation of the design. In one aspect, the simulation is part of the iterative process in

which a design is analyzed (including simulated) and the designer has the option to refine the design further. The design is then further analyzed (including hardware simulation) and this process repeats until the designer is satisfied with the design.

343. See EX-1004, 10:13-15 (“FIG. 9 is a flowchart of a specific embodiment of the construction of a hardware/software co-simulation according to the method of FIG. 2.”); 6:36-34; 7:32-8:3; 9:14-15; 11:12-20; 12:6-11; 18:63-67; 20:39-21:28; 22:38-40; 23:51-56; 22:65-67; 28:13-30; claims 2-5, 27, 35.

344. [This paragraph is left intentionally blank]

g) Claim 16: “The system of claim 9, wherein the design user interface is in communication with a hierarchal classification database, and the user selects the selected control type, display type, input type and output type parameter values from the hierarchal classification database”

345. I have described above for element 1[a] the reasons that the selections made by CoWare designer are from a hierarchal classification database

(including that it would be obvious to use such a database to hold the
selection options).

346. I incorporate that analysis in element 1[a] regarding the selections and
the claimed database herein.

B. Ground 1: The Challenged Claims Are Obvious Over Banerjee.

347. Banerjee’s description of the functional development of the designed system mirrors that of the ‘657 patent – which was filed many years after Banerjee published.

<u>‘657 Patent</u>	<u>Banerjee</u>
“the invention includes a system and method for designing integrated circuits (IC) to integrate into an electronic device for control using IC components that are developed for generic characteristics of electronic devices. A developer can select the functionality of their device and an IC can be designed off the standard components.” EX-1001, 4:46-56.	“the designer compil[e]s a set of system specifications based on the desired functionality of the SoC.... These specifications can be used to define a SoC in terms of the desired inputs and outputs, as well as desired functionality ...” EX-1005, 0004.

348. In the ‘657, the device’s “function” is “categorized again into control types, display types, input types and output types.” EX-1001, 9:1-4. The ‘657 patent uses “functions,” “parameters,” and “characteristics” interchangeably in referencing these function types. EX-1001, 9:1-4 (“The function sub-category is finally categorized again into control types, display types, input types and output types”), 9:11-15 (“Diving further into the classification library provides a category describing the functions of the

blender, which functions may be further categorized by control type, display type, input type, and output type.”); 8:31-33 (“One of the key aspects of the various embodiments of the present invention is the observation by the inventor that the control systems applicable to the majority, if not all, of devices is that an individual device may be defined by four parameters: 1) control type; 2) display type; 3) input type; and 4) output type.”), 14:47-50 (“Once the user selects a device, the user may be provided with a selectable set of characteristics of the user's device. By way of example, the type of characteristics may include control, display, input and output, communications.”). Moreover, Fig. 3 of the ‘657 patent also characterizes these as “functions”:

4.3.1.2.2 Functions

4.3.1.2.2.1 Control Types

4.3.1.2.2.2 Display Types

4.3.1.2.2.3 Input Types

4.3.1.2.2.4 Output Types

349. Thus, the ‘657 does not appear to differentiate between “function” or “parameter” or “characteristic” with respect to the claimed control, display, input, and output types.

350. [This paragraph is left intentionally blank]

1. Banerjee Discloses, And/ Or Renders Obvious, The Challenged Claims.

a) Claim 1: Preamble: “A method for designing an integrated circuit which controls a device when the integrated circuit is integrated into the device comprising”

351. As I have discussed above in the overview of Banerjee, it discloses a software tool for designing hardware and software. The use of the tool as described below teaches the claimed method (and system in claim 9).

352. Banerjee’s claim 17 teaches, or renders obvious many of the claim elements by itself:

17. A computer readable storage medium, comprising **executable instructions for:** entering into a contractual agreement with a host **to access a suite of components** provided by at least one vendor of intellectual property, **selecting components from the suite of components to design an optimized representation of an integrated circuit;** testing the representation of the integrated circuit; paying for the selection and testing of the selected components; and **generating at least one hardware definition language (HDL) description file describing the integrated circuit.**

353. EX-1005, claim 17.

354. Banerjee’s software tool (including elements 15 and 16 in Fig. 1) allows a designer to select the functionality of a desired device design ASICs and SoC for specific applications (devices) and then output the

integrated circuit design and the associated software. EX-1005, ¶¶0027-0028.

- i. Element 1[a] “selecting a value for a control type parameter associated with an electronic device from a hierarchal classification database;

selecting a value for a display type parameter associated with an electronic device from a hierarchal classification database

selecting a value for an input type parameter for the electronic device from the hierarchal classification database;

selecting a value for an output type parameter for the electronic device from the hierarchal classification database;

receiving the selected values for the control type, display type, input type and output type parameters into a memory of a processor programmed by software commands to design a hardware integrated circuit and associated operating software for controlling the hardware integrated circuit to control the electronic device”

355. Banerjee’s software tool performs “selecting” and “receiving” as claimed, in multiple ways.

356. I have discussed herein for 1[a] for Rompaey how the ‘657 patent identifies that the parameter types (control, display, input and output) correspond to the claims. I incorporate by reference that discussion.

357. Banerjee's tool allows the user to select various "parameters" from a hierarchal database using almost verbatim language to the Challenged Claims:

Module 21 further includes a rules engine for determining configurable parameters that the designer or computer program may select. In one embodiment, the rules engine uses a database of descriptive elements that can be selectively incorporated into an RTL description of the component. In other embodiments, the rules engine may access a flat description file that includes one or more parameters that can be selectively customized by either inputting data or by selecting one of a plurality of options for each component. In other embodiments, the description of customizable parameters are described in a relational database with each description of the component incorporating various combinations of the configurable options.

358. EX-1005, ¶0050. As detailed in the paragraph above, Banerjee's designer may select the functionality for the device (based on different applications as described herein). That functionality is related to components implementing the functionality that are selected. The designer can then further select various "parameters," or "characteristics," or

“attributes” or “configurable options” for those components. Each of those corresponds to selecting the claimed parameters.

359. Banerjee teaches many ways for a designer to select the claimed parameters associated with the device.

360. First, Banerjee’s tool receives from a designer’s selections the system design, the related components implementing the design, and the “parameters” and “characteristics” thereof. These selections are received via multiple different user interfaces.

[0033] FIG. 2 illustrates one embodiment of computer program **15** that a designer can use to develop the description file. Computer program **15** includes a functional description module **21** that guides the designer to develop a high-level or macro description of the device. In one embodiment, the description includes a functional description of the device that is generated through a series of menus presented as a graphical user interface (GUI) displaying a series of high level functional block components. Computer program **15** also includes the capability to upload design scripts that describe customized elements created by the designer. This feature promotes inclusion of specifications from multiple IP vendors into the SoC design flow.

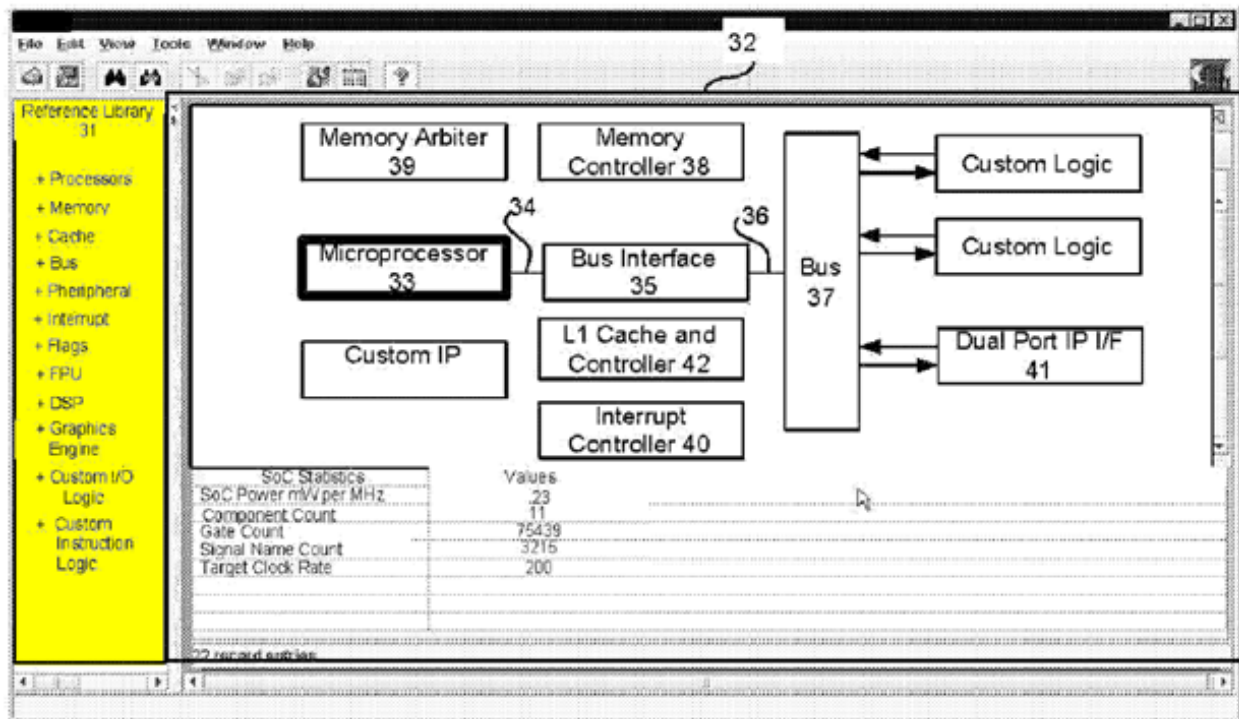
[0034] A second module **22**, the parameter description module, guides the developer in defining the performance requirements for the functional blocks. Module **22** provides a GUI for selecting, by way of example, optimal clock operating frequencies, power levels and maximum allowed power dissipation, die size, and I/O requirements. One skilled in the art will appreciate that other operating parameters can be readily included in module 22. Module 21 utilizes these parameters to optimize the selection of functional block components.

361. The GUI, design scripts, interactive software prompts and command line parameters and the configuration file (and the other items) provide the designer with the ability to select parameters of the device as defined by the design of the different functionalities implemented in the device. Such selections are received into the memory of the workstation / computer executing Banerjee's tools.

362. Banerjee's tool includes a GUI, "design scripts," or "in the form of an interactive software prompt, or ... using command line parameters, by accessing a pre-built configuration file, or a concurrently running application or other computer program 15 resident on the designer's system

or another system linked thereto.” EX-1005, ¶¶0033-0034(quoted above in full).

363. Banerjee gives an example of its GUI in Figure 3. As shown in Figure 3 below, the GUI allows for selecting from different classes of components in a “reference library 31” database (an example of the claimed hierarchal classification database):



364. Banerjee describes Figure 3 and its selection abilities:

[0047] FIG. 3 illustrates a graphical user interface (GUI) associated with module 21 of the computer program 15. The GUI of module 21 provides a reference area 31 and a work palette 32. When executed, module 21 ultimately provides an HDL description of the selected

components and deposits it in the description file. Components are identified in the reference area 31 and when selected are instantiated in the work palette 32.

[0048] Components in reference area 31 may include processors, multiple processor cores, memory sub-systems, bus sub-systems, I/O interfaces, and custom logic by way of example. Indeed it is not unusual for a SoC design to include a processor, a co-processor, video processor, digital signal processing processor or a graphics processor coupled to other components such as memory, memory controllers, caches, a DSP, bus structures (such as a crossbar switch, hierarchical bus structures, networks on chip (NoCs) or direct point-to-point communications), bus controllers and I/O bridges all of which are obtained from the reference area 31 of the GUI. Advantageously, reference area 31 may provide the designer of a SoC with a plurality of configurable or configured processor cores so that the designer can implement a wide range of functions ranging from conventional DSP functions to media-processing functions while employing a consistent set of development tools.

EX-1005, ¶¶0047-0048.

365. Banerjee gives further examples of how the tool might receive selections as claimed:

366.

[0058] It is to be understood that each module 22-25 comprises a GUI, however, such GUIs are not individually illustrated herein

because the design of such are typically application dependent. Note, however, an embodiment of the present invention may include an additional top level GUI (not illustrated) that provides a menu for a designer to select one of the modules 21-25 or to enter into one of a plurality of electronic agreements for the design, simulation, verification and, optionally, the manufacture of an integrated circuit. These agreements may differ in terms of contractual obligations, access to confidential information or access to modules 21-15.

[0059] In one embodiment, the GUI is implemented as an Asynchronous JavaScript and XML, or AJAX, based web page. AJAX is a web development technique for creating interactive web applications that makes web pages more responsive by exchanging small amounts of data with the server such that the entire web page does not have to be reloaded each time the designer requests a change. This increases the web page's interactivity, speed, and usability by avoiding page re-loads when presentation elements (drop-downs, labels, etc.) are changed in response to some other element changing or being selected. It will be appreciated that other remote scripting techniques may be employed to develop an interactive web page that can be hosted by a data center or web sever.

[0060] It will be appreciated that a typical design will require the display of voluminous amounts of data on the designer's browser. Further, it will be appreciated that web pages, unlike native applications, are loosely coupled (meaning that the displayed data is not tightly bound to data sources), the data must be first collected into an HTML page format before presentation on the designer's browser.

Rather than reload the web page for each data modification made by the designer, the present invention uses the remote scripting language to by-pass the requirement to make loosely coupled web page behave much like a tightly coupled application, but with a more variable lag time for the data to pass through a longer “wire” to the remote web browser.

[0061] In one embodiment, component specific information is not downloaded to the designer's browser until required by a local action at client computer 12, in which case the GUI would fetch the necessary additional data to populate the web page without requiring the entire web page to be re-loaded. Advantageously, HTML is generated locally within the browser JavaScript calls and the actual data are downloaded only when needed. This feature means that Ajax web pages on the designer's browser can appear to load relatively quickly since the payload is relatively small. In addition to “load on demand” of components, the present application may, in some embodiments, load stubs of event handlers and then load the functions on the fly to reduce [sic] bandwidth consumption for complex functionality.

EX-1005, ¶¶0058-0061.

367. The description above teaches a POSITA an exemplary implementation of the GUI (and other input forms). As noted above, the GUI may also include a top-level tool that allows for selection of the subtools described in Figure 2.

368. Paragraph 0088 describes the ability of the program to receive into memory the selection of many different types of characteristics including the claimed characteristics:

[0088] As indicated at 63 of the method shown in FIG. 6, a repository location for each component is defined. The technology repository files in the present invention store all of the information related to cells necessary for the synthesis process, **including for example logical function, input/output timing, and any associated constraints.** In the present invention, each designer can define his/her own repository name and location(s), thereby adding further flexibility. A makefile is then generated using the **description file information, repository information previously described, as well as additional designer input information relating to the desired configuration and attributes of the device.** For example, in the computer program 15, the **designer is asked to input the type of “build” (e.g., overall device or system configuration), width of the external memory system data bus, different types of extensions, cache type/size, etc., as described in additional detail below with respect to FIG. 7. It will be recognized that such input may be in the form of an interactive software prompt, or alternatively may occur using command line parameters, by accessing a pre-built configuration file, or a concurrently running application or other computer program 15 resident on the designer's system or another system linked thereto. Many other methods of input are**

also possible, all of which are considered to be within the scope of the invention.

EX-1005, ¶0088.

369. Effectively, Banerjee teaches many different forms of the claimed “selection” and, in each, the selection is received into a memory of the workstation running Banerjee’s design tool.

370. Figures 2 and 3 (below) also disclose exemplary forms of input.

371. Second, Banerjee’s tool receives the claimed characteristics by “access[ing] a flat description file that includes one or more parameters that can be selectively customized by either inputting data or by selecting one of a plurality of options for each component.” EX-1005, ¶0050 (quoted in full above). The flat description file is received into the memory with its selections when the file is read by the program.

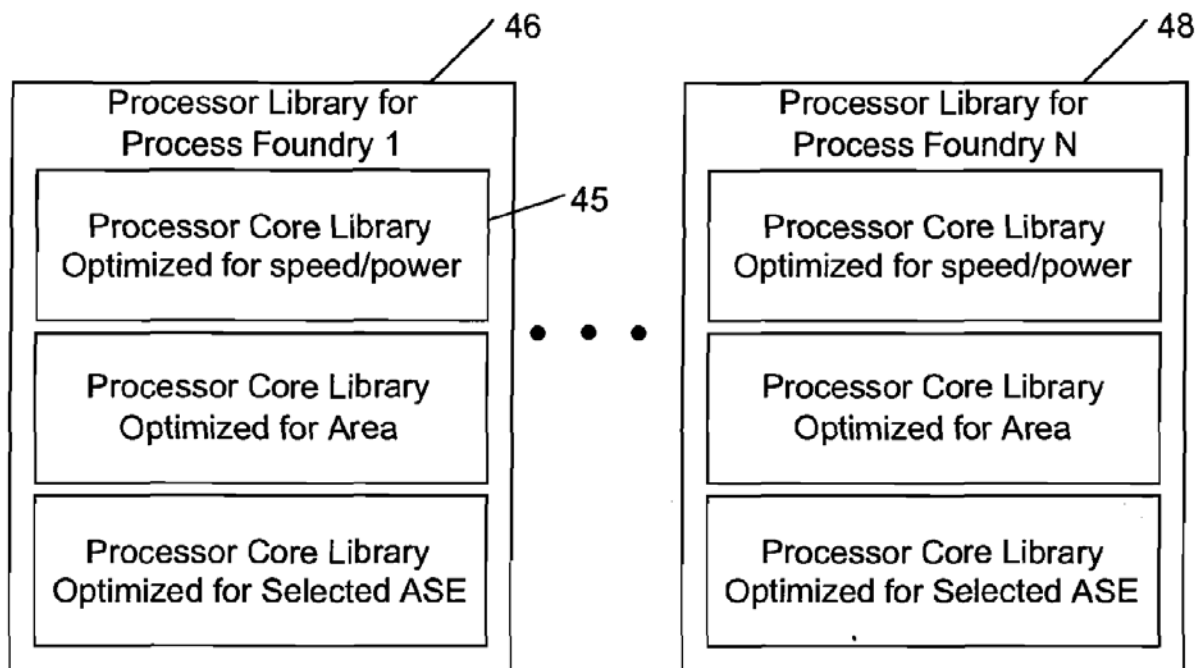
372. For example, the description file can be modified by the user as part of an iterative process after the tool has performed analysis of the design. EX-1005, ¶0009, 0028, 0030, 0040, 0045, 0047, 0067, 0069, 0070 (“as the designer makes additional performance selections, the description file is updated with the RTL code to reflect the designer's designation.”).

373. Third, Banerjee’s will “select” additional parameters from the libraries and databases to perform, for example, optimization. EX-1005,

0064-0067 (describing optimization process); 0078-0086 (describing selections available to system for optimization); 0096-0097; 0030, 0034-0038, 0045; 0055-0056. See also EX-1005, Fig. 1 (“Repositories 18”); Fig. 3 (“Reference library 31”), Fig. 4 (“Processor library”), Fig. 6, Fig. 11 (“Read library files”).

374. These forms of selections are from “libraries” and “repositories” and “databases” – which are all examples of the claimed hierarchal classification database. The libraries and repositories would be understood to be databases by a POSITA. Banerjee uses the terms interchangeably as shown immediately below. Figure 4 is a “database structure” that includes hierarchical “processor library” with various sub “libraries” for different subclasses of processors. A POSITA would understand this type of hierarchical classification structure to be used for each of Banerjee’s classes of devices that are shown in the left hand of Figure 3. Moreover, libraries and repositories are typically stored in an indexed fashion in some hierarchy with different classifications so as to be easily searchable to locate the desire library elements (as depicted in Fig. 4 below).

375. “FIG. 4 illustrates a graphical user interface and database structure.”
EX-1005, 0016.



As illustrated in FIG. 4, selected performance characteristics, provided from module 22, are compared to a matrix of performance specifications associated with each component in a repository. In one embodiment, the description of the performance specifications for each component comprises a plurality of entries in a relational database that includes a basic RTL description of the component linked to other files that describe selectively customizable parameters

....

376. EX-1005, 0063.

377. As noted above, “each component comprises a plurality of entries in a relational database.” The relational database would be understood by a POSITA as one that provides a type of hierarchy and classes within the database to establish the relationships between related elements.

378. The selectable components are stored in a “database” as “class[es]”

that have selectable “subclass[es]”:

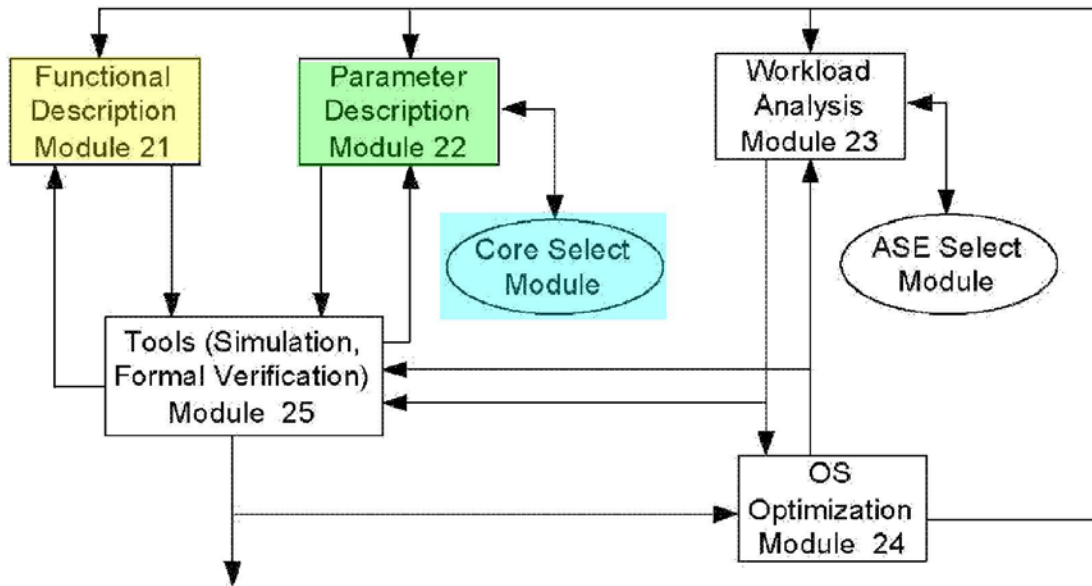
[T]ables 46-48 preferably comprise an object-oriented programming database where each component is defined as an object that is a member of a component class (for example, a processor class) and each version or variation of a particular component is an individual instance that is defined by a subclass that is a more specialized description of the processor class.

379. EX-1006, 0065. *See also*, EX-1005, 0050 (quoted above describing the “database of descriptive elements that can be selectively incorporated” and the “customizable parameters” in “relational database...): 0065 (the “selected characteristics” of the “processor class” “utilize a relational or multidimensional database...”); 0126 (“associated with each component listed in the reference library is a vendor source database file....”); EX-1005, Fig. 1 (“Repositories”); Fig. 3 (“Reference library”), Fig. 4 (“Processor library”), Fig. 6, Fig. 11 (“library files”).¹

¹ The ‘657 patent similarly describes using a hierarchal classification database.

EX-1001, 9:45-48 (“designer will may use a series of cascading menus

380. Banerjee's Fig. 2 depicts an iterative, interactive design process in which the designer selects the functional parameters for the designed device:



381. EX-1005, Fig. 2. First, a designer specifies a system design:

Computer program 15 includes a **functional description module 21** that guides the designer to develop a high-level or macro description of the device. In one embodiment, the description includes a functional description of the device that is generated through a series of menus presented as a graphical user interface (GUI) displaying a series of high level functional block components.

corresponding to the classification scheme of the hierarchal classification scheme....”)

Computer program 15 also includes the capability to upload design scripts that describe customized elements created by the designer.
EX-1005, 0033.

382. Second, Banerjee’s selects further “parameters,” “attributes,” and “characteristics” of the system design and its components in selected classes and subclasses (each being examples of claimed “parameters”).
EX-1005, ¶¶0067-0071, 0074-0092:

A second **module 22, the parameter description module, guides the developer in defining the performance requirements for the functional blocks**. Module 22 provides a GUI for selecting, by way of example, optimal clock operating frequencies, power levels and maximum allowed power dissipation, die size, and I/O requirements. Module 21 utilizes these parameters to optimize the selection of functional block components.....
Module 22 enables the designer to select performance characteristics that are not otherwise determinable from the functional description...
EX-1005, 0034-0035.

383. The “**core select module**” works with “**parameter description module 22**” to allow designers to select from classes and subclasses of processors and select their parameters, features, and attributes tailored to the specific desired application / device. EX-1005, ¶¶0064-0068, Figs. 4, 8.

384. In Banerjee, a designer tailors components and parameters thereof for specific “applications” (*e.g.*, functions for the devices controlled by the designed hardware / software). For example, Banerjee teaches selecting specific parameters for “applications” such as:

- “application[s] [that] demands real-time processing of video or audio packets” with a “DSP application” (EX-1005, ¶0069);
- a “media player application” in which “the processor is configured to include media processing or DSP extension instructions for certain media applications,” (EX-1005, ¶0081);
- “signal processing application” (EX-1005, ¶0086);
- “embedded applications” (EX-1005, ¶0067);
- an “application [that] may perform significant number of floating point multiplication operations” as opposed to “embedded application where the application software seldom performs the multiplication operation” (EX-1005, ¶¶0039, 0072, 0080);
- an “application where a dual core SoC is provided” (EX-1005, ¶0082);
- “applications that do not require address mapping or protection” (EX-1005, ¶0054);

- “applications that are essentially logical control oriented” (EX-1005, ¶0086).

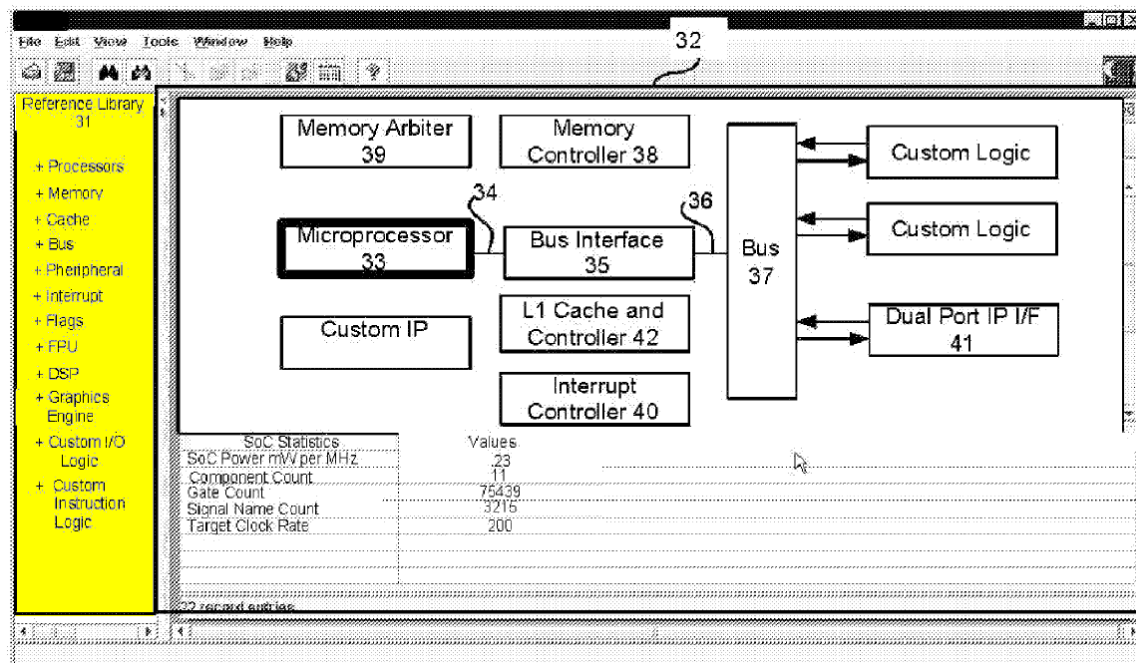
385. For each of Banerjee’s various categories of components, a designer may select application specific “parameters,” “attributes” and “characteristics” – which all disclose the claimed “parameters.” EX-1005, ¶¶0067-0071, 0074-0092. Banerjee describes a “cell phone” device that might have specific requirements. EX-1005, ¶0077. Thus, Banerjee’s system allows the system designer to select the components and characteristics of those components that are specific to the devices for which the hardware and software is being developed.

386. The selections (from Banerjee’s hierarchal databases of “class” and “subclass”) allow Banerjee’s system to select the functions / parameters for a particular application device. Because there is a “large number of possible applications for a SoC, the specific instruction set modifications can be numerous and are typically dependent on the specific application.” EX-1005, ¶0070. Using the “libraries,” “the designer can easily mix and match “best of breed” IP for a particular application.” EX-1005, ¶0124. “[A] designer may select the [component] that is best suited to the particular

application...” EX-1005, ¶¶0125. These are all forms of the claimed selection.

387. Banerjee uses different terminology to describe the selection (and receipt by the design program) of the claimed “characteristics,” Such as “characteristics” (EX-1005, ¶¶0035, 0038, 0040, 0063-0065, 0069); “parameters” (EX-1005, ¶¶0034, 0041-0042, 0045, 0050, 0063-0066, 0078, 0097, 0116); and “attributes” (EX-1005, ¶¶0043, 0055-0057, 0066, 0072, 0088-0089).

388. One invalidating example is that a designer may use the Fig. 3 GUI to specify an “application” device that includes: (a) a processor to control the device; (b) a “graphics engine” providing a graphic display; (c) a multimedia component providing “real-time processing of video or audio packets” and (d) “custom I/O logic” and/or “peripherals” that provide the inputs and outputs to the device.



389. EX-1005, Fig. 3. Fig. 3 shows categories of different component types along the left hand column.

Components in reference area 31 may include processors, multiple processor cores, memory sub-systems, bus sub-systems, I/O interfaces, and custom logic by way of example. Indeed it is not unusual for a SoC design to include a processor, a co-processor, video processor, digital signal processing processor or a graphics processor coupled to other components such as memory, memory controllers, caches, a DSP, bus structures ..., bus controllers and I/O bridges all of which are obtained from the reference area 31 of the GUI. ..., reference area 31 may provide the designer of a SoC with a plurality of configurable or configured processor cores so that the designer can implement a wide range of functions ranging from conventional DSP functions to media-processing functions

EX-1005, ¶0048.

390. As explained below, these various classes of components correspond to one or more of the control type, display type, input type or output type. The designer would select those classes from the classes listed in Figure 3 and then further select the specific subclasses and parameters for the ultimate selection which are all received into the workstations memories through the selection.

391. These classes are “hierarchical” such that, when “selected,” each of the classes displays further refined options for the components that can be selected showing a hierarchy:

[R]eference area 31 comprises a hierarchical index of various components. ... Component may be selected in reference area 31 and dragged into the work palette 32...

392. EX-1005, ¶0049. The designer then selects “configurable parameters” (a further example of the claimed parameters):

Module 21 further includes a rules engine for determining configurable parameters that the designer or computer program may select. In one embodiment, the rules engine uses a database of descriptive elements that can be selectively incorporated into an RTL description of the component. In other embodiments, the rules engine may access a flat description file that includes one or more parameters that can be selectively customized by either inputting data or by

selecting one of a plurality of options for each component. In other embodiments, the description of customizable parameters are described in a relational database with each description of the component incorporating various combinations of the configurable options.

393. EX-1005, ¶0051.

394. This customization by the designer reflects the designer's selection of parameters for the functionality of the device and the controlling ASIC. The "plurality of options" provides further classified selections for each of the types of parameters selected.

395. The designer may then proceed to specify additional parameters and "attributes" (corresponding to the claimed "parameters") of the selected components. EX-1005, ¶¶0052-0054. Further categorized and hierarchical selections can be made:

Further, once a component is selected, the designer may, if desired, also input related attributes when a selected component also requires the selection of hardware attributes. In one embodiment, the designer is prompted to select an appropriate value for the attribute from a popup menu. ... Instead of selecting a value, the designer can select an "optimize value" essentially reserving the attribute value to be determined at a later point in the design process or determined automatically via analysis of the workload by selecting the optimization function.

EX-1005, ¶0055.

396. The series of hierarchical menus allows for refinement and selection of the categorized inputs and outputs defining the functionality of the device. “[A] second menu, if necessary, enables selection” of additional characteristics and “attributes.” EX-1005, ¶0056. These additional selectable menus are examples of selecting parameters from a hierarchical classification database. Each of them relates to an additional selection based on the prior selection from a set of known parameters corresponding to the prior selection.

397. [This paragraph is left intentionally blank]

398. Banerjee allows for selection of the claimed control, display, output and input parameters in selecting the “classes” and “subclasses” of components through the Fig. 3 GUI and subsequent selections described above for the parameters, characteristics, and attributes. Each of those are forms of selection of parameters as claimed.

Components in reference area 31 may include processors, multiple processor cores, memory sub-systems, bus sub-systems, I/O interfaces, and custom logic by way of example. Indeed it is not unusual for a SoC design to include a processor, a co-processor, video processor, digital signal processing processor or a graphics processor coupled to other components such as memory, memory controllers,

aches, a DSP, bus structures (such as a crossbar switch, hierarchical bus structures, networks on chip (NoCs) or direct point-to-point communications), bus controllers and I/O bridges all of which are obtained from the reference area 31 of the GUI. Advantageously, reference area 31 may provide the designer of a SoC with a plurality of configurable or configured processor cores so that the designer can implement a wide range of functions ranging from conventional DSP functions to media-processing functions while employing a consistent set of development tools.

399. EX-1005, 0048.

400. Banerjee describes multiple different parameters corresponding to the claimed classes.

401. **“Control Type Parameter”**

402. Banerjee describes selection of many “control type parameters.”

403. This includes the selection of any of the following “classes” (and then the “subclasses”) of components shown in Fig. 3 and described in 0048 (quoted above): processors, multiple processor cores, memory controllers, DSP (digital signal processor), and bus controllers.² *See also* EX-1005, 0027 (selecting “processor core, specialized processors, buses and

² The ‘657 uses the term “controller” to describe a device that is separate and apart from the claimed “device” that is being designed. *E.g.*, EX-1001, Figs. 6-7.

controllers, and software components”), 0116 (listing types of selectable control parameters), 0053-0054, 0125. Similarly, the “applications that are essentially logical control oriented” show selection of “control type parameters.” EX-1005, ¶0086.

404. For example, the ‘657 patent notes that “At step 2003 The developer may install the IC into their device and integrate it with the control aspects of the device. For example, if the device were a microwave, the IC may integrate with the microwave controls such that the IC could completely or partially manipulate the microwave. The IC may be directly connected to the device's internal bus such that it directly manipulates the device....”

405. In addition, selection of Banerjee’s software is also a “control type parameter.” The ‘657 patent states that the designed device “contains Control Management Software.” EX-1001, 12:65-13:8, Fig. 7. The “Management software” may include “drivers or firmware necessary to operate the IC.” EX-1001, 15:64-67. This is exactly what Banerjee discloses in the ability to select the software that controls Banerjee’s designed device. EX-1005, 0075 (“module 24 configures the operating system to support application specific functions by incorporating only selected drivers”); 0072-0074, 0081, 0085, Fig. 5. Banerjee notes that “the

design of a new device typically requires control and application software.”

EX-1005, 0030. Such control software (including the drivers, firmware and operating system that may be selected) is a control type parameter as claimed. The “control type parameter” is the specific software (and, as needed, any custom instructions) that effectuate the control by controlling the operation of the processor.

406. For each selected “class,” Banerjee’s tool allows for further selections of “parameters,” “characteristics,” and “attributes” for that class and the subsequently selected subclasses as detailed above. These further selections also constitute selection of the claimed control type parameters.

407. **“Display Type Parameter.”**

408. Banerjee discloses the selection of several different types of display type parameters. The “display type” includes visual communication such as “word or graphic.” EX-1001, 9:16, 14:53-57.

409. Banerjee discloses selecting the class of “processor” which then includes selecting subclasses such as “a graphics processor” or a “video processor” and components that can “implement... DSP [and] media-processing functions.” EX-1005, 0048; Fig. 3 (including “Graphics Engine” class). The designer can further select from associated other

parameters / components such as memories and buses. EX-1005, 0048; Fig. 3. This class / subclass selection is then further refined by selecting further parameters, attributes, and characteristics (the claimed “parameter”) as described herein. Each selection corresponds to the claimed selection. For example, in addition to the selection of the parameter that the display would include graphics or word (as selected by selecting the graphics engine), the designer would then select the “other operating parameters” by module 22 such as the type of graphics, resolutions supported, protocols supported.

410. The “graphics engine” (or “graphics processor”), “video processor” and “digital signal processor (‘DSP’)” are device output and display parameters (*e.g.*, a device that outputs/displays graphics and/or video) that the designer selects. EX-1005, ¶0048.

411. The designer may “select” a specific subclass of graphics or video processor and attributes thereof as a parameter. EX-1005, ¶0066. Within that selection, the designer may select additional parameters. *E.g.*, EX-1005, ¶0068 (“...provide additional instructions targeted to a specific task. For example, one logic block may provide a set of instructions directed to handling graphics oriented tasks”). The designer can select “attributes” of

such processors such as these additional instructions or “other operating parameters” as noted in the operation of Module 22.

412. For the video/DSP, “if the application demands real-time processing of video or audio packets, the processor core 33 selected by the designer may include specific instructions directed towards the digital signal processing (DSP) application.” EX-1005, ¶0069. These result in the selections of various selected additional “IP blocks” and “application specific instruction set modifications.” *Id.* The selections of these additional blocks of logic is a form of parameter because it provides information about the functionality of the display as well as implementation details of that functionality.

413. Similarly, the designer may select / configure “media-processing functions” (a type of display, input, and/or output parameter for the device / application) and then the characteristics of such a selection. EX-1005, ¶¶0048, 0081. “[I]f the processor is configured to include media processing or DSP extension instructions for certain media applications,” then the system allows for further configuration between hardware and software. EX-1005, ¶0081.

414. The selection of additional instructions for graphics, video, or media-processing is a form of the claimed selection because it is a selection of parameters that define the display and the types of display (such as more or less sophisticated graphics) as well as frame and resolution abilities based on these additional instructions.

415. **“Output Type Parameters.”**

416. Banerjee discloses the selection of several different types of output type parameters. The ‘657 patent teaches that “input and output of the device may be audio, video, etc. ... The blender's display references an ‘output’” EX-1001, 14:59-61; EX-1001, 9:22-24. Thus, each “display type parameter” immediately above also corresponds to an “output type parameter.” For example, to meet the claim (which recites separate display / output parameters), Banerjee’s teaching would include the “graphics” as a “display” parameter and the “real-time video” as an “output” parameter.

417. Similarly, Banerjee teaches selecting components for “real-time processing of ... audio packets.” EX-1005, 0069. This exactly corresponds to the ‘657 statement that “output of the device may be audio.” EX-1001, 14:59. Thus, this further teaches the selection of the output parameters in Banerjee as detailed for the “media” processing in the display parameters

above. These selections include the selections of classes, subclasses, parameters, characteristics and attributes in Banerjee as described herein. For example, the user “selects” parameters related to additional audio processing logic and instructions. EX-1005, 0069-0071. Furthermore, the “output type parameters” selection would include the “peripherals” [sic], “Custom Instruction Logic,” and the “I/O custom logic” discussed below for the “input type parameters.”

418. **“Input type parameter”**

419. Banerjee discloses the selection of several different types of input type parameters. Banerjee notes that its specifications for the designed device including “desired inputs and outputs.” EX-1005, 0004.

420. Banerjee’s Fig. 3 specifically identifies at least the following selectable “classes” that correspond to the input and output type parameters: “peripheral” (sic), “DSP,” “custom I/O logic,” “Custom Instruction Logic,” and “Interrupt.” Each class includes its own set of selectable “subclasses” along with parameters, attributes and characteristics as described above. Each of these selections corresponds to the claimed selection of input type parameters.

421. For example, the “custom I/O logic” allows for selection of categories of such input and outputs specific (“custom”) for the designed application (device). For example, ¶0048 (and Fig. 3) specifically identify the ability to select:

Components in reference area **31** may include processors, multiple processor cores, memory sub-systems, bus sub-systems, **I/O interfaces, and custom logic** by way of example. Indeed it is not unusual for a SoC design to include a processor, a co-processor, video processor, digital signal processing processor or a graphics processor coupled to other components such as memory, memory controllers, caches, a DSP, bus structures (such as a crossbar switch, hierarchical bus structures, networks on chip (NoCs) or direct point-to-point communications), bus controllers and **I/O bridges** all of which are obtained from the reference area **31** of the GUI. Advantageously, reference area **31** may provide the designer of a SoC with a plurality of configurable or configured processor cores so that the designer can implement a wide range of functions ranging from conventional DSP functions to media-processing functions while employing a consistent set of development tools.

422. EX-1005, ¶0034. This “custom I/O logic” would require specific parameters (selected as discussed herein) to provide the “custom” aspect. Those parameters are “custom” to the device – providing device-specific logic.

423. One example of “custom I/O logic” that would be recognized by, or obvious to, a POSITA would be the “custom I/O logic” that would allow a cell phone (referenced by Banerjee) (or other wirelessly equipped device) to receive and process wireless signals. This would include the various RF and processing components to receive, demodulate, decode and process input RF signals (and the corresponding logic to transmit output signals). Rompaey gives examples of these types of components. A POSITA would recognize that “parameters” for such “custom I/O logic” would include the frequencies and protocols of the wireless signals (such as CDMA, FDMA, GSM, ...).

424. Banerjee describes how the user may select specific parameters for this custom I/O logic and the I/O requirements:

A second module 22, the parameter description module, guides the developer in defining the performance requirements for the functional blocks. Module 22 provides a GUI for selecting, by way of example, optimal clock operating frequencies, power levels and maximum allowed power dissipation, die size, and I/O requirements. One skilled in the art will appreciate that other operating parameters can be readily included in module 22. Module 21 utilizes these parameters to optimize the selection of functional block components

425. ¶0034. Such I/O requirements disclose (or render obvious) the selection of input and output parameters as noted in this paragraph. The “other operating parameters” would include, for example, the wireless parameters discussed above for a cell phone example such as the RF operating frequencies and the parameters needed for the RF components to demodulate, decode and process the RF signals. Each of these parameters (and the ones above) would correspond to the input and output type parameters in the ‘657 claims.

426. In addition, Banerjee specifically discloses a “cell phone” as a device. EX-1005, 0077. A POSITA would understand “cell phone” to include devices with input / output functionality such as touch screens or buttons – which matches the ‘657 statement that “the input device may be push buttons or touch sensitive elements...” EX-1001, 9:7-8. These are also forms of the “custom I/O logic.” The designer would select parameters including by selecting the specific type of input / output (such as push button or touch screen for a phone) and then selecting the “other operating parameters” using module 22 as described above. These are all forms of the claimed parameters. By 2007 (when Banerjee was filed) and certainly by 2013 (the priority date on which the prior art is assessed), a POSITA

would understand that cell phones had buttons and displays. The buttons would include buttons such as for volume control or to turn the device on / off. The displays would include touch displays and LCD displays. The keyboards would include either “hard” keyboards (such as a numeric touchpad) or “soft” keyboards implemented through touchscreen displays. Reference to a cell phone, in the context of Banerjee with its overall specified designs, would at least render obvious that the designed ASIC would include the peripherals well-known to be present on a cellular phone (e.g., buttons, screens, numeric keyboards...).

427. Banerjee’s selection of interrupts and “peripherals” also disclose (or render obvious) forms of both input and output parameters to be selected. EX-1005, Fig. 3, ¶¶0053, 0076, 0097. In Banerjee, the ability to select the category of peripherals from Figure 3 and then select the various sub-menus, including with the characteristics, attributes and parameters (as discussed above) discloses the selection of inputs and outputs characteristics for the device that will include the designed SoC and the associated peripherals specified by the designer’s selections and the parameters thereof (all corresponding to the claimed selections).

428. For example, Fig. 3 specifically shows “pheripherals” [sic] as a selectable component. A POSITA would have understood “peripherals” to include devices such as keyboards, buttons, and displays which are all well-known and commonly used peripherals for devices. After selecting the type of peripheral (which would be selection of an input and/or output type parameter), the user would then select the “other operating parameters” needed to implement that type of peripheral.

429. For example, paragraph 76 states:

As indicated at 55, module 24 selects operating functionality based on the workload analysis performed by module 23. This feature enables **the operating system to support any identified peripherals or services while building the operating system kernel**. Clearly, there are aspects of the operating system things that can be configured, however, module 24 links the operating system configuration to the design of the integrated circuit and the application software.

430. ¶0076. The highlighted passage above is an example of the tool performing “selection” of parameters (such as selecting I/O drivers or devices drivers and their operating parameters) to include in the operating system. For example, the tool would select the I/O device drivers for keyboard, touch screens, displays or buttons to include in the operating system and such selection would include the claimed parameters both in the

selection of the peripheral subclass (the specific type of peripheral) and in the “other operating parameters” reference above selected by module 22 such as, for example, the protocols and signaling used to communicate with the input and output peripheral. These are all the claimed input and output parameter types.

431. These input and output types for the peripheral also include “buses” (Fig. 3) that provided for communications to device components such as the peripherals. Many buses provide specific forms of input and output type parameters such as buses leading to either entirely separate peripherals (for example, a PCI bus or a bus for an RS-232 port or a “USB” which is a “universal serial bus” leading to the pervasive USB connectors for various devices). The selection of the sub-component of bus and the parameters associated with it (e.g., the specific protocols) corresponds to the claimed selection. Banerjee provides extensive details on the buses and selection thereof and parameters thereof. EX-1005, 0032, 0048, 0051-0054. For example, as noted in paragraph 54:

Note that reference area **31** includes a plurality of repositories each of which comprises a plurality of related components. For example, processor **33** is selected from the processor repository while the bus interface unit **35** and bus **37** are selected from the bus logic

repository. More specifically, each repository includes a plurality of components that are each represented as an icon but which are treated by computer program **15** as a unique IP block. Within each repository, **a number of different variations of a component type are visually presented to the designer** in the form of either text or an icon

432. EX-1005, 0054. The selection of the “variations” is a form of selection as claimed and would also lead to the sub-menus in which “other operating parameters” are selected.

433. Banerjee’s inputs / outputs include the ability to send / receive data in Banerjee’s described cell phone through “custom I/O logic” such as network interface controllers.³ This would be part of the “desired inputs and outputs” selected by the user for the device. EX-1005, 0004.

434. Banerjee’s “digital signal processor” (DSP) and “Custom Instruction Logic” selection also is a form of an input type parameter. The “signal” being “processed” would be understood to be an “input” – for example being “input” to the device from sensors, communication components, peripherals, or offboard storage. By defining the type of DSP and its associated customized specific instructions, the designer is selecting an input type parameter. EX-1005, 0069, 0081, 0048. For example, one form

³ E.g., EX-1001, 17:56-61 (describing “network interface controller” as a form of input / output for the device).

of “DSP and “Custom Instruction Logic” might be a particularized processor and instructions to be able to process the particular wireless signals for which the device is designed. An example would be to be able to specify a DSP and Custom Instruction Logic to process, for example, CDMA wireless signals. This would be a selection of the sub-class of processor and then further parameter selections related to that subclass. Each selection corresponds to the claimed selections.

435. Similarly, the designer may select / configure “media-processing functions” (a type of input / output characteristic for the device / application) and then the characteristics of such a selection. EX-1005, ¶0048. “[I]f the processor is configured to include media processing or DSP extension instructions for certain media applications,” then the system allows for further configuration between hardware and software. EX-1005, ¶0081. These instructions would be specific to the device into which they are incorporated because they would reflect additional selected hardware and software for that device’s functionality including its inputs and outputs.

436. Each of the selections above are “received” as claimed because Banerjee’s tool receives these selections into the memory of the software tool being used by the designer to specific, tailor, and optimize the design

and output the hardware and software for the design. The tool itself is receiving the selections so that it can operate thereon to develop the hardware and software implementing the design.

437. If PO argues that Banerjee does not expressly disclose this element, Banerjee renders obvious every aspect of this element for all of the reasons above and, in addition for the following reasons. For example, according to PO, the claimed parameters are met by the following:

control type parameter (e.g., system control signals), a display type parameter (e.g., multimedia or graphics aspects), an input type parameter (e.g., memory and other system inputs), and an output type parameter (e.g., system outputs) for an integrated circuit design.

438. EX-1008, 000025. It is not clear what “system control signals” or “system outputs” references as these as seemingly just vague open ended statements.

439. However, to the extent PO’s assertions are understandable, Banerjee provides extensive details on selecting each of these types of parameters cited by PO. EX-1005, Fig. 3; ¶¶0038, 0048, 0053-0054, 0073-0074, 0077-0078, 0088, 0096-0098, 0125.

440. According to PO, the “memory” represents an “input.”⁴ Banerjee provides extensive details on the selection of memory and its characteristics. EX-1005, Fig. 3; ¶¶0038, 0048, 0053-0054, 0073-0074, 0077-0078, 0088, 0096-0098, 0125. These are shown below in paragraphs 0073-0078.

Refer now to FIG. 5, which illustrates module 24 of computer program 15 in more detail. Once the software and hardware attributes of the design are established, it is possible to determine the optimal operating system footprint. This module is used to minimize the footprint of the operating system and middleware if any, that will typically be stored in nonvolatile memory on the device or in an off-chip persistent storage device. Module 24 interfaces with modules 23 and 22 to determine whether it is preferable to implement certain functions in software rather than hardware. For example, in one embedded application, the application software may perform significant number of floating point multiplication operations so a hardware multiplication is the preferred implementation thereby minimizing the need for the operating system to include a floating point multiplication algorithm. In another embedded

⁴ EX-1008, 000025 (“an input type parameter (e.g., memory and other system inputs)”).

application where the application software seldom performs the multiplication operation, it is area and power efficient to implement the multiplication operation in software rather than in hardware even if it will take significantly longer to perform the multiplication when the need arises.

[0073]

Module 24 enables the configuration of the operating system to minimize its footprint in memory in the finished product. Specifically as indicated at 52, a minimum feature operating system is initially provided. The operating system may be, for example, Windows CE available from Microsoft Corporation, Linux or a real time operating system (RTOS). In one embodiment, the entire operating system may be ported to the device but importing a full blown operating system will result in inefficient use of memory resources because unused features will be unnecessarily included. However, with module 24, it is possible to configure the operating system based on a prototype description. In a preferred embodiment, a prototype or baseline operating system is provided together with options that the designer can configure to support the operating of the hardware components.

[0074]

As indicated at 53, module 24 configures the operating system to match the compiled features of the operating system with the instantiated hardware components. For example, if the design comprises a simple processor that does not require paging to manage virtual memory such that the design does not include a translation look aside buffer (TLB) based memory management unit, then the operating system module will not include a TLB management module. If the system does not include a cache, then there is no need to bring in a cache management module. Specifically, as the designer selects each hardware feature in module 21, module 24 updates the operating system configuration to incorporate corresponding operating system modules that support the selected feature. During the operating system build, these support features will be retained and linked to pertinent components while other operating system options are not selected.

[0075]

As indicated at 54, module 24 will also incorporate corresponding operating system modules that support the functional features of the added IP blocks in the operating system build. Thus, depending on the selected

hardware functionality, module 24 configures the operating system to support application specific functions by incorporating only selected drivers.

[0076]

As indicated at 55, module 24 selects operating functionality based on the workload analysis performed by module 23. This feature enables the operating system to support any identified peripherals or services while building the operating system kernel. Clearly, there are aspects of the operating system things that can be configured, however, module 24 links the operating system configuration to the design of the integrated circuit and the application software.

[0077]

By way of illustration, consider a design process for building a cell phone where a minimum operating system footprint is desirable due to the limited memory that is available on this SoC design. As will be appreciated by one of skill in the art, cell phones do not have the luxury of having megabytes of memory available to store never used operating system services. Accordingly, module 24 configures the operating system at build time so that only the services that need to be present are built into the operating system.

[0078]

In one embodiment, module 24 outputs C or C++ code that configures the Windows, Linux or RTOS operating system by outputting header files, switches, and a modified makefile. Advantageously, the hardware and the operating system do not need to be configured at the same time, but they may be configured in parallel or in a sequential manner. Advantageously, module 24 may make certain tradeoffs in order to optimize one or more functions in conjunction with modules 21 and 22. By way of illustration, the designer can provide module 24 with a range of possibilities such as 1 megabyte of memory, restrict the processor to be no bigger than 3 mm² while meeting certain performance metrics. Module 24 then determines the optimal configuration for the design by selecting the processor that most closely matches the specified parameters and providing any additional functionality in software either at the application, middleware or operating system level.

441. Thus, at a minimum, Banerjee renders these elements obvious.

442. Banerjee's "ASICs" are specifically tailored toward the "application" (the device) that the IC device will control. As detailed above, one focus of Banerjee was the ability to specifically tailor the controlling IC for its

“application” such as multimedia, embedded, real-time audio and video.

Such “applications” either are, define, or render obvious, the device and its parameters because, for example the hardware and software is “specific” to that device / application. Thus, at a minimum, Banerjee renders obvious receiving selected parameters for, or associated with, the device.

443. Moreover, PO asserts that the claimed parameter selection corresponds to “options and parameters for the integrated circuit that correspond to those of the electronic devices into which the integrated circuits will be integrated.” EX-1008, 000025. Similarly, PO asserts “[e]xamples of controlled devices ...devices (and/or their subsystems, chips, or boards).” EX-1008, 000024-25. As detailed above, Banerjee discloses such configuration and attributes of both the designed integrated circuits and the associated device (with the tailored “applications”) that the circuits (and software) control including their “subsystems” and “chips” of those devices.

444. Regarding “hierarchal classification database” Banerjee discloses a GUI of components and subcomponents identifies specifically as “classes” and “subclasses” and their “parameters” and “characteristics” and “attributes” –which can be progressively refined with further selections

using the sub-menus – all from forms of databases such as a “relational database” as detailed above. At a minimum, this renders obvious the receipt of selections from a hierarchal classification database.

445. Regarding “hierarchal classification database” Banerjee discloses a GUI of components and subcomponents identified specifically as “classes” and “subclasses” and their “parameters” and “characteristics” and “attributes” –which can be progressively refined with further selections using the sub-menus – all from forms of databases such as a “relational database” as detailed above. At a minimum, this renders obvious the receipt of selections from a hierarchal classification database. I have discussed in detail above the reasons that Banerjee’s disclosure for the selectable parameters would be understood by a POSITA to include selections from a hierarchal classification database. All of that analysis would be used to support that such disclosure at a minimum renders the usage of such a hierarchal classification database obvious.

446. Thus, Banerjee discloses, including renders obvious, this element.

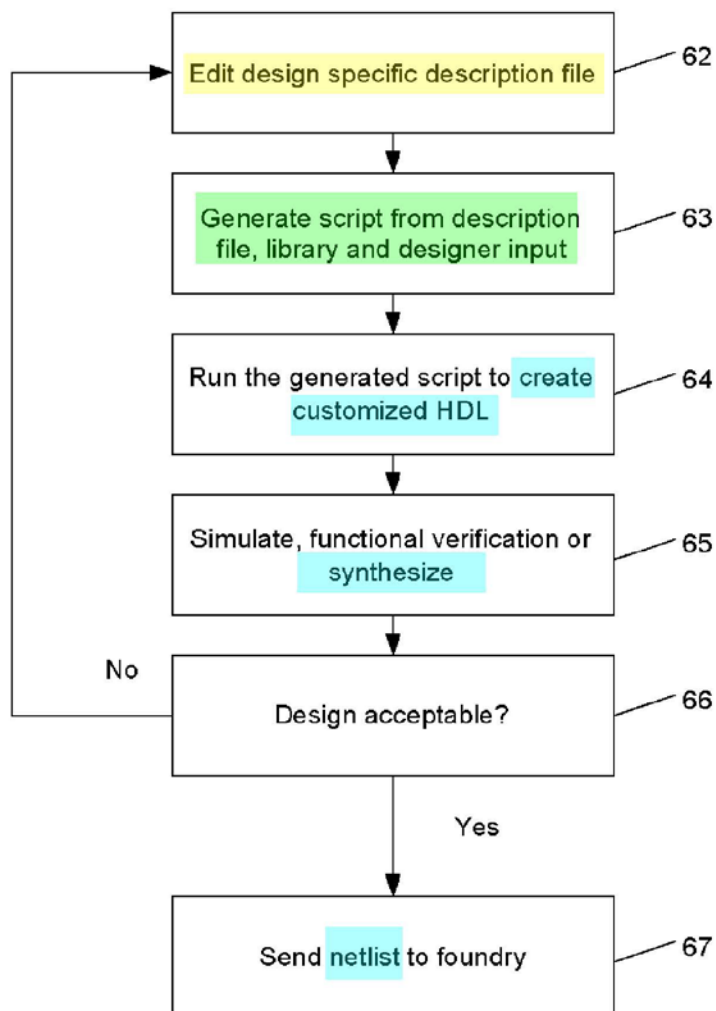
447. [This paragraph is left intentionally blank]

- ii. Element 1[b] “analyzing by the processor the selected values for the control type, display type, input type and output type parameters of the electronic device;”

448. Banerjee's tool performs different forms of the claimed "analyzing."

449. As an initial matter, as noted above, Banerjee has an iterative process by which the designer selects and specifies the components, characteristics, parameters and attributes of the design. The tool then "analyzes" the design and the designer can then select / change additional parameters to iteratively work toward a final system design. In each of the analyses below, the software tool is using the selected values for the different parameters discussed in element 1[a] above in the analysis to be able to ultimately generate the hardware and software.

450. Banerjee's tool analyzes the selected characteristics to create the hardware integrated circuit design for the device in formats such as netlists, HDL, VHDL, and RTL. For example, Fig. 6 shows this process in which the **selections** are received from **edited description files, the libraries/repositories and designer inputs** (which are inputs and selections) with the system ultimately "**create the customized HDL**" in the form of a "**netlist**" that could be sent to a foundry:



451. EX-1005, Fig. 6.

452. Figure 6 shows many different forms of the claimed analysis which ultimately results in the general hardware integrated circuit design (“netlist”). In the Fig. 6 example, “a description file specific to a particular design is edited at 62 where the designer selects desired modules or functions for the design. In one embodiment, analysis of the workload is

used to automatically generate a description file. Instructions relating to the design are added to or subtracted from the description file, or generated as necessary.” EX-1005, ¶0086. *See also* EX-1005, ¶0089 (“create a customized VHDL or Verilog model”) which is another form of the claimed analysis.

453. Banerjee repeatedly teaches that the received data (as identified in 1[a] above) is used to generate the integrated circuit. “[C]omputer program 15 comprises an interactive architecture module that gathers information regarding the design parameters of the SoC and generates a merged HDL file descriptive of the design that complies with those design parameters.” EX-1005, ¶0045.

454. Such analysis includes Banerjee’s processes of “merging,” “compiling,” and “optimizing.” The “compiling” is also referenced as “logic synthesis” which is a form of analyzing the received selections. “Module 25 also performs logic synthesis to compile and optimize the high-level circuit description and generate a gate-level netlist representation of the device.” EX-1005, ¶0044.

455. “Merging” analyzes the system to see what components may be linked together. “Compiling” is the process of converting the system design to the

hardware integrated circuit designs and associated software. EX-1005, 0004, *See also* EX-1005, ¶¶0089-0102 (describing the compilation, merge, and optimization analysis occurring before the generation of final Verilog, VHDL or other HDL files); Figs. 7, 8 (teaching “optimizing” the design); Fig. 9 (“configure the ASE [application specific instruction extensions]”); Figure 10 (“generate structural HDL files...”); ¶0038 (“optimization module...”). Similarly, Banerjee’s tools analyze the design for incompatibilities within the design. EX-1005, ¶0041, ¶¶0082, 0125. Each of these necessarily includes analysis of the selected parameters described in 1[a] because such actions cannot be performed without consideration of the design parameters identified and selected in paragraph 1[a].

456. As detailed in 1[a], Banerjee’s “modules 21, 22, and 23” allow the designer to select the device components and characteristics:

Using the descriptions generated by modules 21, 22 and 23, module 25 of computer program 15 assembles a description file that comprises a series of predefined VHDL, Verilog, System Verilog or other synthesizable constructs of an HDL. ... The description file specifies the functional requirements of the device in terms of the components and the overall operational characteristics in RTL or other programmable form that is readily converted to RTL.

EX-1005, ¶0040;

457. These selections are then analyzed to ultimately generate the hardware integrated circuit design and software and output those. *See also* EX-1005, ¶0043 (“computer program 15 initiates the creation and testing of HDL-based synthesizable SoCs”), ¶0009 (“an interactive architecture module gathers workload information about the SoC and generates a merged HDL file descriptive of the design.”); ¶¶0047, 0057, 0071; *see also* EX-1005, claims 1, 17.

458. Furthermore, after the designer specifies the system design and the intended applications (*e.g.*, functions for the device such as multimedia), Banerjee’s “module 23 performs a dynamic analysis of the workload to identify instructions and to select appropriate application specific instruction extensions ... Based on the analysis, module 23 recommends the addition of certain application specific instruction set modifications or the designer may specify a defined user instruction.” EX-1005, ¶0067. “[M]odule 24 selects operating functionality based on the workload analysis performed by module 23.” EX-1005, ¶0076. Similarly, a designer’s selection of particular memory “is analyzed (based on the designer’s technology choice), and the memory subsystem values are set accordingly.” EX-1005, ¶0097. These additional instructions would include the

additional software instructions and associated hardware that are specific to the application. This is performed based on the parameters selected in 1[a] because this is an analysis of the overall system design and performance specification as well as how the selected components can successfully integrate.

459. Banerjee's tool will analyze the system design (and parameters) to add any "application specific extensions" (referenced as "ASE" in Figs. 2, 4-5, 9-11). EX-1005, ¶0097. The extensions include "application specific instruction extensions" (EX-1005, ¶0067) which include logic for additional instructions that the processor executes to perform tasks specific to the application (for the device). EX-1005, ¶0081; Fig. 3 ("custom instruction logic"); ¶¶0048, 0069 (for "real-time processing of video or audio ... If the DSP IP block is added to the design, RTL code for the application specific instruction set modifications are added to the description file").

460. Thus, Banerjee teaches many different forms of analyzing the received selections to generate and output the hardware circuit design and associated software.

461. If the Patent Owner argues that Banerjee does not expressly disclose this element, Banerjee renders obvious every aspect of this element. The ‘657 patent has sparse disclosure of “analyzing” (and does not mention that term) and, at most, discloses associating parameters with predefined ICs. As detailed above, Banerjee discloses the same association between the functional design (including parameters) and the implementations supplied by libraries and other inputs for those designs. Thus, at a minimum, Banerjee renders this “analyzing” obvious.

462. Moreover, the Patent Owner asserts that the claimed “*analyzing...*” occurs when a designer “uses the [] [d]esign [t]ools to define the desired parameters and configure the selected IP and create ... RTL, an IP microarchitecture, and executable files for the integrated circuit appropriate for the associated electronic devices.” EX-1008, 000029. As detailed above, Banerjee discloses such analyzing to create and output the RTL, netlist, or VHDL defining the IP microarchitecture and the executable files (the OS, application software, and middleware) for the integrated circuit appropriate for the device (that implements the specified “applications”) and thus, at a minimum, renders this element obvious.

463. [This paragraph is left intentionally blank]

- iii. Element 1[c] “outputting by the processor a hardware integrated circuit design based on the selection of input and output characteristics of the electronic device, the hardware integrated circuit including a communication bus for communicating with the electronic device; and”

464. As detailed in 1[b], Banerjee’s tool outputs the hardware integrated circuit designs based on the selections detailed in 1[a] in many formats including Verilog, VHDL, System Verilog, HDL, netlist, GDSII (GDS2), and RTL.

465. I have noted above for Rompaey that Verilog, VHDL, System Verilog, HDL netlists and RTL all teach, or render obvious the claimed format for the hardware integrated circuit designs. GDSII (GDS2) is a format that is used to provide integrated circuit design as well. It is a binary format that can specify the layout of the circuitry for the designed device.

466. Banerjee discloses the outputting of the hardware integrated circuit design based on the selections throughout the specification and claims. *see* 1[b]; EX-1005, ¶0094 (“The output of computer program 15 is an HDL synthesizable model of the design.”); Figs. 5, 6, 7, 10, 11, 12; ¶¶0028, 0040, 0043-0045, 0050, 0066, 0071, 0089-0093, 0095 (“a storage location is identified at 72 for the description file and HDL (in this case Verilog) files to be generated in subsequent steps”), 0096-0098, 0099 (“At 96, the

structural HDL files are generated, along with the synthesis scripts and simulation makefiles.”); 0102 (“...all component instantiations are written to the VHDL file. Finally, output drives are written to the VHDL file at 124....”); 0107, 0116, 0117, 0008-0012, claims 1, 17, 18.

467. Moreover, as I have noted above, the designer may iteratively select various characteristics, run the analysis, output the hardware integrated circuit design, further analyze the design, and then select modified characteristics (starting the process all over again).

468. Banerjee teaches that the designed circuit includes communication buses as claimed. *See* Fig. 3 (“Bus” in element 31; “Bus 37”); ¶¶0116 (“the IC comprises a SoC comprising ... one or more bus structures ...”); 0027 (“...automated description of hardware components, such as ... buses and controllers...”), 0031, 0048 (“Components in reference area 31 may include ... bus sub-systems ... Indeed it is not unusual for a SoC design to include ... bus structures (such as a crossbar switch, hierarchical bus structures, networks on chip (NoCs) or direct point-to-point communications), bus controllers ... all of which are obtained from the reference area 31...”); 0051-0054 (extensive details of buses), 0082, 0088, 0098, 0116, 0125.

469. These bus structures would be used to communicate the instructions to be executed from the storage (such as the ROM that can store the software for the designed ASIC) to the processor for execution.

470. Thus, Banerjee discloses (including renders obvious) this claim element.

471. [This paragraph is left intentionally blank]

- iv. Element 1[d] “outputting by the processor software programming commands to be .executed by the hardware integrated circuit for controlling the electronic device in accordance with the selection of categorized input and output characteristics of the electronic device through the communication bus of the hardware integrated circuit using a communication protocol compatible with the electronic device and the hardware integrated circuit.”

472. Banerjee’s tool outputs the software specific to the designed system and application thereof and based on the selections identified in 1[a]. The output software includes: (1) the operating system; (2) application software; (3) application specific instruction extensions; and (4) “middleware” (such as I/O device driver software).

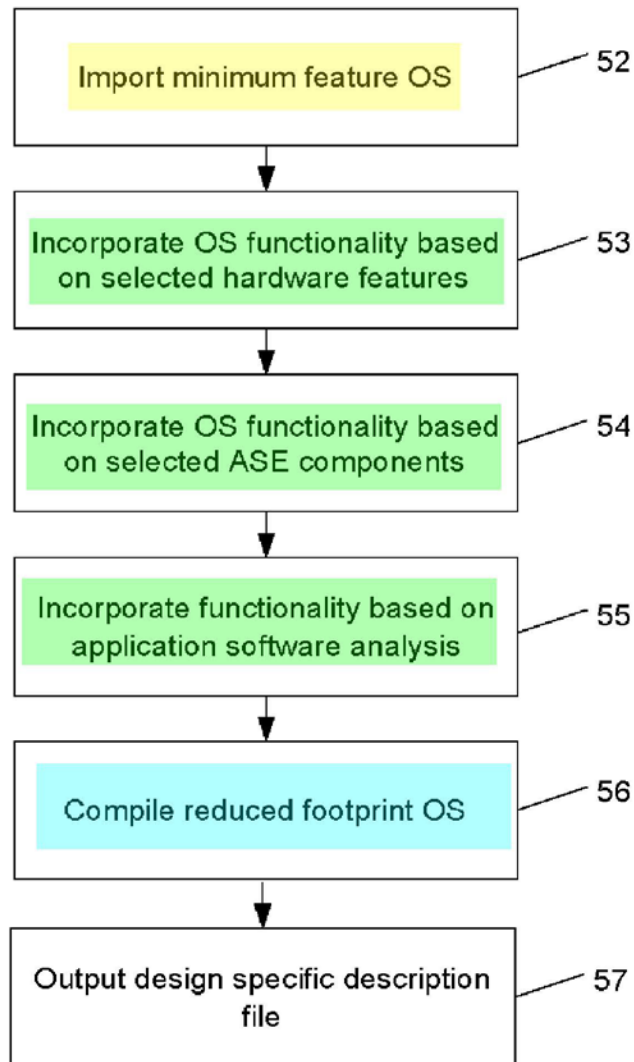
[M]odule 24 **outputs** C or C++ code that configures the Windows, Linux or RTOS operating system by outputting header files, switches, and a modified makefile. Module 24 then determines the optimal

configuration for the design by selecting the processor that most closely matches the specified parameters and providing any additional functionality in software either at the application, middleware or operating system level.

EX-1005, ¶0078; *See also* EX-1005, ¶¶0038-0039, 0045-0046, 0056, 0067, 0084.

473. Each of these types of software are in accordance with the selections and each control the electronic device and the functions thereof. Indeed, for the application specific instructions, the application software, and I/O drivers (and maybe even the operating system), that software is specific to the type of application (device) for which the ASIC / SoC is being designed.

474. Fig. 5 exemplifies how the operating system software is compiled and output:



475. EX-1005, Fig. 5; see also ¶¶0072-0082 (describing Fig. 5 and how OS software is generated and output), 0083 (“the optimized operating system is compiled into a reduced footprint operating system that is instantiated in the device, preferably in a non-volatile computer readable storage medium.”). For example, the OS and device drivers are specific to the applications and components selected. EX-1005, ¶¶0075, 0081.

476. Banerjee's tool compiles and stores (outputs) the software into nonvolatile memory of the device (such as a "ROM"). EX-1005, ¶¶0072, 0083.

477. Similarly, Banerjee's tool generates and outputs software instructions related to the application specific instructions which as discussed above are based on the specific selections for the device (application) being designed. EX-1005, ¶¶0086, 0038.

478. The software is purposefully designed to use communications protocols compatible with the device and to operate through one or more of the communications buses on the device so that the software can control the device. For example, the software has to load from the persistent storage into the processor(s) controlling the device to be able to execute properly and the software must be able to communicate across the buses to access data in memory and control the device.

479. There are multiple types of buses to/from the memory on the designed devices that would be selected and used to convey the instructions from the "nonvolatile memory on the device" to the processor for execution. EX-1005, ¶0072, ¶0082 ("module 24 would compile a symmetric multiprocessing operating system that would support, by way of example,

PCI Express, DDR2, bus width, description of on-chip buses, whether it is compatible with the Open Core Protocol, cache sizes and many features that can be configured in a typical SoC design.”)

480. Moreover, Banerjee teaches that compatibility among the components is tested and resolved during the development phase. EX-1005, ¶¶0041, 0125. Such compatibility would require that the communication buses use compatible protocols or the instructions would not load or execute properly.

481. [This paragraph is left intentionally blank]

b) Claim 5: “The method of claim 1, wherein the hardware integrated circuit design is based on at least one predetermined hardware integrated circuit component which is able to control one of the selected values for the control type, display type, input type and output type parameters.”

482. Banerjee uses predetermined hardware integrated circuit designs that have been stored in, for example, the repositories and libraries and these correspond to the “predetermined” hardware in this claim because they have been previously designed and then categorized and stored for easy access for further devices.

483. Banerjee describes instances of the predefined hardware models throughout the specification. E.g., EX-1005, ¶¶0040 (“module 25 of computer program 15 assembles a description file that comprises a series of **predefined** VHDL, Verilog, System Verilog or other synthesizable constructs of an HDL.”); 0057 (“Once a component is at least partially instantiated in the work palette 32, a corresponding HDL description script is obtained from repository 18 and added to the description file.”), 0107 (“generate a description of a SoC design using libraries of IP blocks (components) in a hardware description language”); 0028 (repositories); 0054 (“repositories can be defined and components can be readily added to or subtracted from each of the repositories”); 0063, 0065-0066, 0071; 0079-0080; 0088; 0101-0102, 0106 (“component libraries”), 0117; 0119, 0124.

c) **Claim 6: “The method of claim 5, wherein the hardware integrated circuit design is provided as a software schematic, layout, or both.”**

484. As detailed in 1[b], Banerjee’s tool outputs the hardware integrated circuit designs in many formats including Verilog, VHDL, System Verilog, HDL, netlist, GDSII (GDS2), and RTL. As I have noted for Banerjee, these

formats are all well-known formats for hardware integrated circuit designs.

I incorporate by reference my discussions of Banerjee for these format.

485. To the extent that multiple different forms of concurrent output are required, Banerjee discloses or renders obvious the multiple forms. Banerjee describes many different options for outputting the hardware integrated circuit design and the selection of the specific format or formats for outputs is a design choice that a POSITA would find obvious. The designer would be selecting from a known and confined small set of different options identified expressly in Banerjee.

486. The formats include specifically layouts. EX-1005, ¶0091 (“a synthesis engine may create a specific physical layout of the design”). These disclose, or render obvious, this claim.

487. To the extent that a “software schematic” is needed, the disclosure of HDL, VHDL, RTL, and VERILOG teach, or render obvious such a software schematic because the purpose / design of those formats is intended to provide the requisite information for a schematic of the layout of the circuits on hardware (such as silicon wafers).

**d) Claim 8: “The method of claim 1, wherein the
hierarchical classification database is a component of
a fabless development kit.”**

488. Element 1[a] details Banerjee’s tool and the database(s) used by it (including the libraries and repositories). Banerjee’s tool is part of a “fabless development kit” based on the specific language used in the ‘657 to describe “fabless.”

Fabless design of ICs and MCUs is well known in the art. In general, the term fabless design is applied to process where the design of the IC is accomplished separately from the manufacture of the device itself.

489. EX-1001, 10:9-13.

490. As an initial matter, this claim recites an element (fabless development kit) that the ‘657 patent admits is “well known in the art.” Given that it is well-known, this “fabless” aspect does not impart any novelty or non-obviousness.

491. Banerjee repeatedly describes its software tool(s) as being separate from the manufacturing of the device by, for example, a foundry. For example, after the entire design is done, then it is sent to a foundry. EX-1006, Fig. 6.

492. EX-1005, 0050 (after the design is completed, “the circuit description is sent to a selected foundry”);

In this manner, the design repositories and synthesizable files are maintained in a secure environment. Preferably, only a hardened design is exported to a foundry at the conclusion of the design process as indicated at **67**.

493. 0091 (exported to a foundry at conclusion of design process);

494. Moreover, a user has the option to “select” from different foundries – which indicates that the tool is separate from any particular manufacturer.

EX-1005, 0095 (“As shown in FIG. 7, process **70** comprises a series of steps, the first of which is initializing the program and setting default values at step **71**. Next, a storage location is identified at **72** for the description file and HDL (in this case Verilog) files to be generated in subsequent steps. In step **73**, the technology/feature size or selected foundry is obtained from the designer (or another source, as previously described). For example, the designer may desire a 0.130 micron process or a selected foundry manufacturer for their design.”)

Once synthesized, the GDSII data is sent from data center **150** to a selected semiconductor foundry, as indicated at **154**, that will fabricate the design for the designer **157**. Advantageously, the designer does not need to explicitly contract with the foundry because

the entire design process is automated by computer program **15**. Further, it should be noted, that the designer does not require access to any of the “soft” synthesizable IP block because such information is securely maintained within data center **150**. This feature is particularly important should the designer be located in a geographical region where export laws would not otherwise allow access to the RTL.

495. 0108,

496. *See also* EX-1005, 0116-0128 (can select and send design to different foundries). Thus, Banerjee discloses that its tool performs the design separately from the manufacture and discloses this claim. EX-1005, 0120 (“The pricing information may specify the cost to instantiate the component in a design, if any, and the cost for each device that is subsequently manufactured by a contract foundry.”).

497. Given that “fabless design ... is well known in the art,” the decision to use Banerjee’s tool either separate or integrated with a specific foundry is, at most, an obvious design choice. This is particularly true because different foundries may be selected. Thus, at a minimum, Banerjee renders this element obvious.

498. This paragraph left intentionally blank.

e) **Claim 9: Preamble “A system comprising:”**

499. Banerjee teaches this “system.” *See* claim 1 and its preamble above and my discussion of the operation and usage of Banerjee’s tool for designing systems. Each of the components below are disclosed in Banerjee.

- i. Element 9[a] “one or more processors; a design user interface provided by programming commands executed by the one or more processors, the design user interface allowing a user to characterize an electronic device by selecting a value for a control type parameter, a value for a display type parameter, a value for an input type parameter and a value for an output type parameter, all of which are associated with the operation and control of the electronic device;”

500. Banerjee discloses this claim element.

501. Banerjee’s “**computer program**” tool executes on a “**server**” computer which discloses, or renders obvious, the claimed processor[s] as part of a “system” because a processor executes software.

502. As I noted for Rompaey above, the disclosure of the computer (such as a server) teaches, including rendering obvious, the claimed processor because the software tool would be understood to execute on a processor. I incorporate by reference that disclosure for Rompaey above for this element.

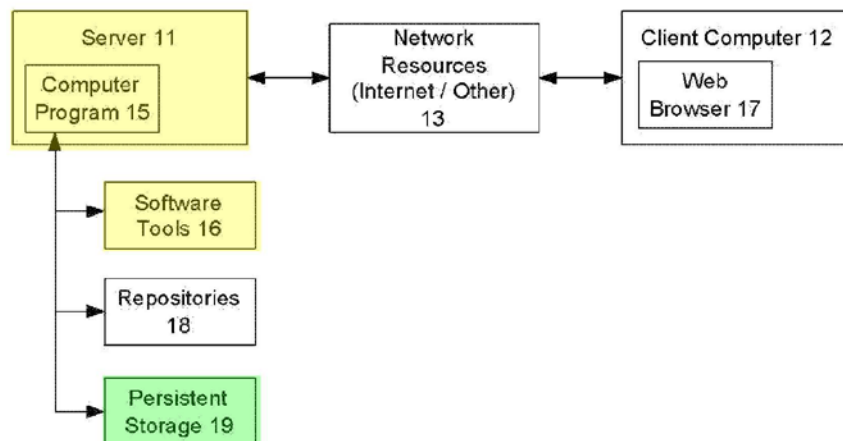


Figure 1

503. EX-1005, Fig. 1;

504. Banerjee repeatedly teaches that its disclosure is based on a software tool executing on a server computer – which renders obvious a processor and explicitly discloses such a processor. *See e.g.*, EX-1005, Fig. 13; ¶¶0027-0028 (“As illustrated in FIG. 1, the present invention for managing the design and configuration of a system on a chip (SoC) or other silicon design takes ... the form of a server 11.... A silicon-on-chip (SoC) designer accesses **server-resident computer program 15 and software tools 16** using a web browser 17 installed on client computer 12. ... server 11 may be one server in a data center environment or ... a plurality of servers.... Computer program 15, software tools 16 and repositories 18 are all retained on a **persistent storage device 19** that is capable of storing computer

readable information. **Persistent storage device 19 may be either coupled to or is a component of server 12. Software tools 16 may further include synthesis and formal verification tools.**”); 0059, 0062, 0085, 0091, 0103-0105, 0112-0116.

505. For example, Banerjee also claims “17. A computer readable storage medium, comprising executable instructions” which renders obvious the claimed processor.

506. Element 1[a] details the “design user interface” provide by Banerjee for designing a system including characterizing an electronic device by selecting the claimed parameters. As detailed therein, each of those selected parameters is associated with the operation and control of the electronic device in which the selected design is incorporated.

- ii. Element 9[b] “logic encoded in one or more non-transitory computer-readable media that, when executed by the one or more processors, is operable to.”

507. Banerjee’s “[c]omputer program 15, software tools 16 and repositories 18 are all retained on a persistent storage device 19 that is capable of storing computer readable information” which is part of the server. EX-1005, ¶0028, Figs. 1, 13. This discloses the claimed logic

encoded in one or more non-transitory computer readable media executed
by the processor in 9[a].

508. The “persistent storage device 19” teaches, including rendering
obvious, the non-transitory computer-readable media.

- iii. Element 9[c] “receive the selected control type,
display type, input type and output type parameter
values selected by the user”

509. Banerjee’s program receives a selection of categorized input and
output characteristics as detailed in 1[a]. I incorporate by reference my
discussion of element 1[a] above..

- iv. Element 9[d] “into a memory associated with the
one or more processors, the logic executed by the
one or more processors operable to design a
hardware integrated circuit design based on the
selection of input and output characteristics of the
device, the hardware integrated circuit design
including a communication bus, and”

510. As detailed in elements 1[b] and 1[c] above, Banerjee’s program tools
design a hardware integrated circuit design and output based on the
selections in 9[c] / 1[a]. That hardware design is output in many different
formats as I have described for element 1[c] above and for claim 5 above.
See 1[d].

- v. Element 9[e] “output software programming commands to be executed by the hardware integrated circuit for controlling the device in accordance with the selected control type, display type, input type and output type parameter values selected by the user using a communication protocol compatible with the device and the hardware integrated circuit through the communication bus of the hardware integrated circuit.”

511. As detailed in 1[d], Banerjee discloses the outputting of the software as claimed herein.

- f) **Claim 15: “The system of claim 9, wherein the one or more non-transitory computer-readable media further comprises logic that, when executed by the one or more processors, is further operable to simulate the hardware integrated circuit's functionality based on the hardware integrated circuit design”.**

512. Banerjee’s tool simulates the hardware integrated circuit functionality.

See EX-1005, Figs. 2-3, 6-8, 10-13 (all reciting simulation); ¶¶0010 (“an apparatus for generating, simulating, and/or synthesizing a SoC design using the aforementioned method is disclosed that allows a SoC designer to

remotely interact with the program during the design, simulation, and/or synthesis processes.”), 0031, 0038, 0041-0042 (“During the simulation and verification, ... Timing simulation is preferably done after device layout to determine if the design works at the parameters (e.g. clock rate and power) specified by the designer....Full-system simulation of the design ensure that it will support the workload as analyzed and parameterized by modules 23 and 24....”), 0057 (“... module 21 builds the HDL file for simulation and synthesis....”); 0058, 0090-0093 (describing simulation process); 0099, 0103, 0115, 0127.

513. As I have noted above, Banerjee teaches an iterative process in which the user makes selections for the system design, the design is analyzed including simulation of the hardware integrated circuit functionality, the user may assess the simulation and decide to select more features or changes to the system design and the process starts all over again.

g) **Claim 16: “The system of claim 9, wherein the design user interface is in communication with a hierarchal classification database, and the user selects the selected control type, display type, input**

**type and output type parameter values from the
hierarchal classification database”**

514. I have described above for element 1[a] the reasons that the selections made by Banerjee’s designer are from a hierarchal classification database (including that it would be obvious to use such a database to hold the selection options).

515. I incorporate that analysis in element 1[a] regarding the selections and the claimed database herein.

**XI. SECONDARY CONSIDERATIONS DO NOT REBUT THE PRIMA
FACIE CASE OF OBVIOUSNESS**

516. In my opinion, secondary considerations do not rebut the prima facie case of obviousness for at least two reasons. *First*, I have not seen any evidence of any secondary consideration to date.

517. Indeed, Patent Owner never proffered such evidence during prosecution of the ‘657 patent, the parent ‘899 patent or its related applications including the ‘466 Application which was rejected (and abandoned) on an obviousness basis. To the extent Patent Owner seeks to proffer secondary considerations evidence in this proceeding, I reserve the right to comment on such evidence.

518. *Second*, I do not believe that Patent Owner will be able to establish a nexus between any secondary consideration and the challenged claims. I have been advised that a nexus does not exist where the alleged secondary consideration results from something other than what is both claimed and novel in the claims.

519. As I explained above, there is nothing novel in the claims. Indeed, all claim elements are disclosed in the prior art. Thus, I do not see a basis for Patent Owner to assert that there is a nexus between any secondary consideration and the claims.

520. Moreover, as I have discussed before, the prosecution of the parent ‘899 would indicate that the purported basis for non-obvious and/or novelty was generating hardware integrated circuits based upon user specifications and the ‘657 claims were identified as being not patentably distinct to the parent ‘899 claims for which these statements were made. However, as detailed above in Grounds 1 and 2 as well as the State of the Art, such tools had existed since the 1980’s to perform the actions cited by the Applicant to the Examiner in support of patentability as discussed in the Prosecution History section above. Thus, there would not be any nexus between any alleged secondary considerations and any patentable aspect of the claims.

XII. CONCLUSION

521. I may utilize the documents cited and/or listed herein, or portions of those documents, as exhibits at any hearing or trial in this proceeding. I may further prepare and use exhibits that summarize portions of my testimony or key terms or concepts presented therein, or other demonstrative exhibits, at any hearing or trial in this proceeding.

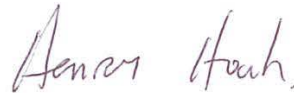
522. I reserve the right to supplement my testimony and this report in response to any judicial determinations, in response to the opinions expressed by the Patent Owner's experts in this proceeding, and/or in light of additional evidence or testimony brought forth at trial or otherwise brought to my attention after the date of my signature below.

523. I hereby declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct, and that all statements made of my own knowledge are true and that all statements made on information and belief are believed to be true. I understand that willful false statements are punishable by fine or imprisonment or both. *See* 18 U.S.C. § 1004.

IPR2024-00566

Declaration of Henry Houh, Ph.D. Under 37 C.F.R. § 1.68 in Support of
Petition for *Inter Partes* Review of U.S. Patent Nos. 9,367,657

Date: February 21, 2024



Henry Houh

Henry H. Houh

15 Ballard Ter.
Lexington, MA 02420
henry.houh@gmail.com
cell: 781-325-8273

Education

Massachusetts Institute of Technology, Cambridge, MA

- PhD in Electrical Engineering and Computer Science, February 1998. “Designing Networks for Tomorrow's Traffic,” thesis supervised by Professor David Tennenhouse and Professor John Guttag. GPA 4.7/5.0
- Master of Science in Electrical Engineering and Computer Science, February 1991. “Demonstration of a laser repetition rate multiplier,” thesis. GPA 4.5/5.0
- Bachelor of Science in Electrical Engineering and Computer Science, June 1989. “Boundary element analysis of arbitrarily shaped dielectric structures,” thesis. GPA 4.7/5.0
- Bachelor of Science in Physics, February 1990. GPA 4.7/5.0

MITx, Cambridge, MA

- September 2020 – present: Enrolled in Statistics and Data Science MicroMasters program, a sequence of five MIT-offered graduate-level courses in statistics and machine learning, culminating in a MicroMasters certificate. Have completed four of five required courses.

Experience

Houh Consulting Inc. / Independent Consultant

- 2009 - present: Technical consultant specializing in Social Networking, Web 2.0, Web Site Development, Data Networking, Optical Networking, Telecommunications, Media Streaming and Voice Over IP. Clients include: Akin Gump Strauss Hauer & Feld LLP, Arnold & Porter Kaye Scholer LLP, BBN, Baker Botts LLP, Barnes & Thornburg LLP, BBN Technologies, Beus Gilbert McGroder PLLC, Bracewell LLP, Christensen O'Connor Johnson Kindness PLLC, Cooley LLP, Covington & Burling LLP, Cozen O'Connor P.C., DLA Piper, Davis Polk & Wardwell LLP, Desmarais LLP, Duane Morris LLP, Erise IP, P.A., Faegre Baker Daniels LLP, Finnegan, Henderson, Farabow, Garrett & Dunner, LLP, Fish & Richardson P.C., Goldman Ismail Tomaselli Brennan & Baum LLP, Haynes and Boone, LLP, Kaye Scholer, Kellogg Huber Hanson Todd Evans & Figel PLLC, Kirkland & Ellis LLP, Klarquist Sparkman, LLP, McCarthy Tetrault LLP,

Declaration of Henry Houh, Ph.D. Under 37 C.F.R. § 1.68 in Support of
Petition for *Inter Partes* Review of U.S. Patent Nos. 9,367,657

McGuireWoods LLP, Morgan, Lewis & Bockius LLP, Munger Tolles & Olson LLP, Orrick, Herrington & Sutcliffe LLP, O'Melveny & Myers LLP, Orrick, Herrington & Sutcliffe LLP, Paul Hastings LLP, Paul, Weiss, Rifkind, Wharton & Garrison LLP, Perkins Coie LLP, Quinn Emanuel Urquhart & Sullivan, LLP, Sidley Austin LLP, Smith Baluch LLP, Sterne, Kessler, Goldstein & Fox PLLC, Troutman Sanders LLP, Venable LLP, Weil, Gotshal & Manges LLP, Wiley Rein LLP, Wilmer Cutler Pickering Hale and Dorr LLP, and Winston & Strawn LLP.

BlocksCAD Inc. (H3XL spin-out)

- 2014 - present: Founder and member of Board of Directors. BlocksCAD is a visual Computer-Aided Design (CAD) program, designed to teach children mathematics and geometric visualization. BlocksCAD is being used in U.S. schools and worldwide in maker spaces, with 350,000+ users. BlocksCAD was selected to be part of the LearnLaunch 2017 Boost Accelerator, the MIT Play Labs 2018 Summer Accelerator, and the MassChallenge 2018 Boston Cohort. Recipient of DARPA DSO grant in 2014, and SBIR grant from the USDA in 2019.

H3XL Inc. d/b/a Einstein's Workshop (formerly Lexington Robotics)

- 2009 - 2022: Founder and President. Started a science, technology, engineering and math enrichment program and creative/maker space, in 7,000 square feet of space. Serve 2,000+ kids and families annually. As of end of 2022, have delivered an estimated 250,000 student-instruction-hours of STEM courses. Principle Investigator for 2-year DARPA grant to improve 3D Computer Aided Design tool developed in-house for the purposes of teaching 8 year olds and up 3D CAD.

Eons

- 2008 - 2009: Chief Technology Officer. Created product that Eons acquired from BBN Technologies. Integrated BBN product with Eons social networking platform and significantly increased the Eons group creation rate. Helped evaluate advertisement platform offerings and rolled out the "Boom Network" advertising network. Eons raised \$32 million from General Catalyst Partners, Charles River Ventures, Sequoia Capital, and Intel Capital.

BBN Technologies

- 2007 - 2008: Delta Division, Vice President of Technology. Grew "Boomerang" counter-sniper project engineering team and significantly de-risked \$10 million worth of product deliveries. Identified sales team leading to \$10 million dollars of new sales within 6 months and \$100+ million in additional orders in the following two years. Boomerang was a significant asset leading to the acquisition of BBN by Raytheon in 2009. Created new business plan and grew team; launched new fully-featured social networking web site in 5 months. Served as lead expert witness in patent infringement lawsuit, resulting in

Declaration of Henry Houh, Ph.D. Under 37 C.F.R. § 1.68 in Support of
Petition for *Inter Partes* Review of U.S. Patent Nos. 9,367,657

\$58 million jury award to client; verdict for patents I testified on were upheld on appeal which resulted in a \$120 million settlement.

- 2004 - 2007: Delta Division, Director of Technology, responsible for commercializing IP and creating new businesses. Hired and grew division's initial engineering team. Wrote three business plans, two of which were funded. For call center business plan, acted as general manager, hiring and managing engineering team, inside sales team, and identifying and recruiting a new general manager. Identified and recruited other key employees to Delta Division, including senior members of team leading to successful internal sales growth and spin-outs of projects. Contracted by BBN to BBN spin-out PodZinger as VP of Operations and Technology. Additional areas studied include mesh networking, network security, satellite communications, and semantic web.

PodZinger Inc. (BBN spin-out, also known as EveryZing and now RAMP)

- 2006: Vice President of Operations and Technology. Significantly upgraded capability of consumer-facing search site and redeployed web site from company to co-location facility. Identified key portions of infrastructure for upgrading and cost reduction. Helped write business plan, evaluating advertisement models of revenue. Hired in operations replacement and phased back to BBN.

Commonwealth Capital

- 2004: Entrepreneur-in-residence (informally), performed technical due diligence on business plans, brainstormed ideas for new businesses with venture partner. With venture partner, left for portfolio company BBN to form core of commercialization team.

Empirix, Inc./Teradyne, Inc.

- 2001 - 2004: Chief Technologist, Engineering Manager, Web Application Test Group. Researched potential new product areas; developed product plan and prototype. Responsible for three new and existing products. Managed off-shore development team. Chief architect for all web testing products. Re-architected core testing product, helped write javascript interpreter. Provided technical vision for core product.
- 2000 - 2001: Chief Technologist, Communication Infrastructure Test Group. Responsible for incorporating new technology internally, tracking new technologies, technical evaluation of partnerships and potential acquisitions. Helped develop division strategy. Developed plans which formed core capabilities for successful new products introduced in 2004-5.
- 2000 - 2001: Engineering Manager, Communications Infrastructure Test Group. Execution of new product plan developed in prior role. Grew team from four existing engineers to team of over 30 on immediate team and over 40 on project. Delivered new platform in one year. Platform and derivatives accounted for large portion of booked products for the division within 2 years and is currently (2008) a key portion of new product offerings.

Declaration of Henry Houh, Ph.D. Under 37 C.F.R. § 1.68 in Support of
Petition for *Inter Partes* Review of U.S. Patent Nos. 9,367,657

- 2000: Empirix was formed as a spin-out of Teradyne in January 2000. Reported to CEO in carve-out of Empirix from Teradyne.
- 1999: (Teradyne) Director of Business Development, Software Test Units. Reported directly to Chairman of the Board/Founder and then to general manager of software test unit (6 divisions of Teradyne). Evaluated and researched acquisition and partnership candidates. Internally assessed technology position in market and gaps in product lines. Worked with senior division staff to develop new product strategies. Attended internal Teradyne divisional board meetings. Chairman served as my mentor.

3Com Corporation/NBX Corporation

- 1999: Software Engineer 5. Continued work after 3Com acquisition of NBX. Built cross-division relationships for new products and research directions. NBX was acquired by 3Com in March 1999.
- 1997 - 1999: (NBX) Senior Scientist and Engineer. Work in IP Telephony. Architected next-generation product. Protocol design and validation for core protocol now used tens of millions of times daily. Led team in integration of IP protocols into current product. Designed audio reconstruction algorithms. Developed applications for bug analysis and diagnosis of system problems. Implementation of network simulator. Work on collaborative projects with external partners. Worked to identify gaps in product. Representative at numerous trade shows. Innovated novel methods of using product.

MIT Department of Electrical Engineering and Computer Science, Cambridge, MA

- 1991 - 1998: Research assistant, Telemedia Network Systems Group. Design, development, and implementation of Gigabit ATM network for distributed multimedia system. Studied host interface design issues. ATM network simulation.
- Spring 1989, Fall 1990, Spring 1995: Teaching assistant, Computation Structures digital systems course. (Spring 1995 Head TA)
- 1988 - 1989: Head laboratory teaching assistant for Computation Structures. Responsibilities included writing and revising lab assignments, and maintaining the lab.
- 1987: Laboratory teaching assistant for Computation Structures.
- 1987: Design, construction, and programming of 16-bit computer.

Agora Technology Group, Incorporated

- 1994 - 1996: Founder and CEO. Conceived and oversaw development of targeted advertising-supported Web sites. Responsible for company's vision and direction. Attended the first two WWW Conferences, presenting a workshop and paper at the first, and appearing on the "Commercialization and Economics of the Web" panel and chairing the "Where Commercial Services and the Web Are Headed" panel at the second. Sold company intact; is currently an operating stand-alone company.

AT&T Bell Laboratories, Holmdel, NJ

Declaration of Henry Houh, Ph.D. Under 37 C.F.R. § 1.68 in Support of
Petition for *Inter Partes* Review of U.S. Patent Nos. 9,367,657

- 1989 - 1990: Implementation of cascadable all-optical fiber logic gate. Modelocking of all-fiber erbium laser. Construction of modelocked laser repetition rate booster. Strong optics laboratory and fiber optic experience.
- Summer 1988, 1987: Research in integrated optics. Analysis of rectangular waveguides using microwave modeling. Fabrication of integrated optical components.

Honors

- MIT Alumni Association Great Dome Award, 2010, Baker 60th Anniversary Reunion Co-Chair (highest group award given by MIT Alumni Association)
- MIT Alumni Association Presidential Citation Award (now known as Great Dome), 2008, Member of MIT Chairman's Salon committee
- MIT Alumni Association Bronze Beaver Award, 2007 (highest individual award given by MIT Alumni Association)
- MIT Alumni Association Volunteer Honor Roll, February 2004
- MIT Alumni Association Lobdell Award, 1999
- Boston Museum of Science Gold Pin for 1000 hours of Volunteer Service, April 1999
- MIT Alumni Association Presidential Citation Award (now known as Great Dome), 1997, Member of Alumni Online Communications Committee

Patents and Patent Publications

- U.S. Patent #9,697,231, H. Houh, J. Stern, R. Spina, M. Meteer, "Methods and apparatus for providing virtual media channels based on media search," July 4, 2017.
- U.S. Patent #9,697,230, H. Houh, J. Stern, "Methods and apparatus for dynamic presentation of advertising, factual, and informational content using enhanced metadata in search-driven media applications," July 4, 2017. See also WO2007056485.
- U.S. Patent #7,975,296, L. Apfelbaum, H. Houh, T. Mayberry and G. Friedman, "Automated security threat testing of web pages," July 5, 2011. See also US20030159063, WO2003067405.
- U.S. Patent #7,877,736, H. Houh and J. N. Stern, "Computer language interpretation and optimization for server testing," January 25, 2011. See also US20050138104, WO2005043300.
- U.S. Patent #7,801,910, H. Houh and J. N. Stern, "Method and apparatus for timed tagging of media content," September 21, 2010. See also US20070112837, US20090222442, WO2007056535.
- U.S. Patent #7,590,542, D. C. Williams, W. C. Hand, H. Houh, A. R. Seeley, "Method of Generating Test Scripts Using a Voice-Capable Markup Language," September 15, 2009. See also EP1530869, US20030212561, WO2003096663.
- U.S. Patent #6,967,963, H. H. Houh, P. Anderson, C. Gadda, "Telecommunication method for ensuring on-time delivery of packets containing time-sensitive data," November 22, 2005. See also EP1060400, WO2000033092, CA2318774.

Declaration of Henry Houh, Ph.D. Under 37 C.F.R. § 1.68 in Support of
Petition for *Inter Partes* Review of U.S. Patent Nos. 9,367,657

- U.S. Patent #5,144,375, M. C. Gabriel, H. H. Houh, N. A. Whitaker, “Sagnac Optical Logic Gate,” September 1, 1992. Also issued as European Patent # EP0456422, July 23, 1997, German Patent #DE69126913, August 28, 1997
- U.S. Patent Application Publication No. 20020015387, “Voice Traffic Packet Capture and Analysis Tool for a Data Network” (Abandoned in 2007)
- U.S. Patent Application Publication No. 20020016708, “Method and Apparatus for Utilizing a Network Processor as Part of a Test System” (Abandoned in 2007)
- U.S. Patent Application Publication No. 20020016937, “Method and Apparatus for Utilizing a Network Processor as Part of a Test System.” (Abandoned in 2007) See also WO2002011413.
- U.S. Patent Application Publication No. 20070106646, “User-directed navigation of multimedia search results” (Abandoned in 2009)
- U.S. Patent Application Publication No. 20070106660, “Method and apparatus for using confidence scores of enhanced metadata in search-driven media applications” (Abandoned in 2009)
- U.S. Patent Application Publication No. 20070106685, “Method and apparatus for updating speech recognition databases and reindexing audio and video content using the same” (Abandoned in 2015)
- U.S. Patent Application Publication No. 20070118873, “Methods and apparatus for merging media content” (Abandoned in 2015)
- U.S. Patent Application Publication No. 20090222442, “User-directed navigation of multimedia search results” (Abandoned in 2016)
- U.S. Patent Application 11/395,732, “Search snippet creation for audio and video data” (Abandoned in 2009)
- U.S. Patent Application 11/774,931, “Methods and apparatus for managing a social networking web site”
- U.S. Patent Application 11/774,947, “Methods and apparatus for organizing media files”
- U.S. Patent Application 11/774,956, “Methods and apparatus for managing an online event”
- U.S. Provisional Patent Application 61/086,909, “Measuring and ranking relationship activity”
- U.S. Provisional Patent Application 61/086,914, “Detecting media object commonality”
- U.S. Provisional Patent Application 61/086,904, “Message categorization based on message characteristics”
- U.S. Provisional Patent Application 61/086,905, “Photo tagging to request action”

Trials, Testimony and Depositions

- Case No. 1:06CV682 (CMH/BRP), Verizon vs. Vonage, U.S. District Court for the Eastern District of Virginia, filed expert report, was deposed and testified at trial.
- Case No. 1:08CV157 (CMH/TRJ), Verizon vs. Cox, U.S. District Court for the Eastern District of Virginia, was deposed and testified at trial.
- Case No. 5:09-cv-476, Two-Way Media vs. AT&T, U.S. District Court for the Western District of Texas, filed expert report, testified at trial.

Declaration of Henry Houh, Ph.D. Under 37 C.F.R. § 1.68 in Support of
Petition for *Inter Partes* Review of U.S. Patent Nos. 9,367,657

- Case No. 2:10-cv-248 (RAJ/FBS), ActiveVideo Networks vs. Verizon, U.S. District Court for the Eastern District of Virginia, filed expert report and was deposed as an expert witness.
- Case No. 1:11-cv-00880-TSE-JFA, Bear Creek Technologies, Inc. vs. Verizon Services Corp., et al, U.S. District Court for the Eastern District of Virginia, was deposed as an expert witness.
- Case No. 3:10-CV-298-BBC, AlmondNet, Inc. vs. Microsoft Corp., U.S. District Court for the Western District of Wisconsin, filed expert report.
- Case No. 6:10-cv-00597, Guardian Media Technologies, Ltd. Vs. AT&T Operations, Inc. et al., U.S. District Court for the Eastern District of Texas, Tyler Division, filed expert report.
- Case No. ESCV2010-02282C, The Octopus Solution LLC v. Gary Brown et al., Essex, MA Superior Court, testified at trial.
- Investigation No. 337-TA-882, In the matter of Certain digital media devices, including televisions, Blu-ray disc players, home theater systems, tablets and mobile phones, components thereof and associated software, U.S. International Trade Commission, filed expert reports, was deposed and testified at hearing.
- Investigation No. 337-TA-995, In the matter of Certain communications or computing devices, and components thereof, U.S. International Trade Commission, filed expert reports, and was deposed.
- Case No. 8:12-cv-122-LES-TDT, Prism Technologies LLC v. AT&T Mobility LLC, U.S. District Court for the District of Nebraska, filed expert report, was deposed and testified at trial.
- Case No. 11-2684 (JWL), Sprint Communications Co., L.P., v. Comcast Cable Communications, LLC, et al., U.S. District Court for the District of Kansas, filed expert report and was deposed.
- Case No. 11-2686 (JWL), Sprint Communications Co., L.P., v. Time Warner Cable, Inc., et. al., LLC, et al., U.S. District Court for the District of Kansas, filed expert report, was deposed, and testified at trial.
- Case No. 12-487 (SLR), Cox Communications, Inc., et al., v. Sprint Communications Company L.P., et al., U.S. District Court for the District of Delaware, filed expert report and was deposed.
- Case IPR 2014-00039, Microsoft Corporation v. B.E. Technology LLC, U.S. Patent No. 6,628,314. Submitted declaration and cross-examined in deposition.
- Case IPR 2014-00086, Apple, Inc. v. Evolutionary Intelligence, LLC, U.S. Patent No. 7,010,536. Submitted declaration and cross-examined in deposition.
- Case No. 6:11-CV-421, Stragent, LLC v. Intel Corporation, U.S. District Court for the Eastern District of Texas, Tyler Division, deposed and testified at trial as a fact witness.
- IPR2014-01366, IPR2014-01367, IPR2014-01368, Samsung et al. v. Straight Path, submitted IPR declarations and was deposed.
- IPR2015-1006, IPR2015-1007, Cisco v. Straight Path, submitted IPR declarations.
- IPR2014-01457, IPR2014-01459, Microsoft v. Biscotti, submitted IPR declaration and was deposed.
- IPR2016-00302, Apple v. Nonend, submitted IPR declaration.

Declaration of Henry Houh, Ph.D. Under 37 C.F.R. § 1.68 in Support of
Petition for *Inter Partes* Review of U.S. Patent Nos. 9,367,657

- IPR2014-00812, Yelp and Twitter v. Evolutionary Intelligence, submitted IPR declaration.
- IPR2015-00307, Cisco v. AIP, submitted IPR declaration.
- Microsoft v. Acacia, submitted IPR declaration.
- IPR2015-00342, AT&T Mobility LLC and Cellco Partnership d/b/a Verizon Wireless v. Solocron, submitted IPR declaration.
- Mediatube v. Bell Canada, Canadian Federal Court, submitted expert report and testified at trial.
- CBM2016-00036, Google v. At Home Bondholder's Liquidating Trust, submitted CBM declaration.
- IPR2016-01198, IPR2016-01201, Apple, Inc. v. VoIP-Pal.com, Inc., submitted IPR declaration and was deposed.
- Case No. 3:14-cv-2824-VC, Trend Micro Incorporated v. Rpost Holdings, Inc., et al, submitted expert report and was deposed
- Akamai v. Limelight Networks, IPR2016-01631, IPR2016-01711, IPR2017-00249, IPR2017-00349, submitted IPR declarations and was deposed
- Hyperlync Technologies v. Verizon Sourcing, No. 650151/2015 New York Supreme Court, submitted declaration and was deposed
- Cascades Streaming Technologies v. Big Ten Networks, submitted expert reports and was deposed
- Klaustech, Inc. v. Google LLC., Case No. 4:10-CV-05899-JSW (N.D. Cal.), submitted expert report and was deposed
- Cisco v. Uniloc, IPR2017-00198, IPR2017-00058, submitted IPR declarations and was deposed.
- Cisco v. Uniloc, IPR2017-00597, submitted IPR declaration and was deposed
- Twitter v. Youtoo Technologies, LLC, IPR2017-01133, IPR2017-01131, submitted IPR declarations and was deposed
- Taser International Inc. (now Axon Enterprise, Inc.) v. Digital Ally Inc., IPR2017-00375, submitted IPR declaration and was deposed
- Taser International Inc. (now Axon Enterprise, Inc.) v. Digital Ally Inc., IPR2017-00376, IPR2017-00515, IPR2017-00775, submitted IPR declarations
- HTC Corporation v. IPA Technologies, IPR2018-00306, IPR2018-00307, submitted IPR declaration
- Hulu v. Soundview, IPR2018-00864, IPR2018-01023, submitted IPR declarations and was deposed
- XpertUniverse v. Cisco, 3:17-cv-3848, U.S. District Court for Northern District of California, submitted declaration in support of claim construction and was deposed
- Improved Search v. Microsoft, C.A. No. 16-cv-650-JFB-SRF (Del.), submitted declaration in support of claim construction
- Apple v. Uniloc, IPR2018-00361, IPR2018-00394, IPR2018-00395, IPR2019-00700, IPR2019-00701, IPR2019-01667, submitted IPR declarations
- LG Electronics v. Uniloc, submitted IPR declaration
- SRC Labs and Saint Regis Mohawk Tribe v. Microsoft, 2:18-cv-00321-JLR, U.S. District Court for Western District of Washington at Seattle, submitted declaration in support of claim construction and was deposed

Declaration of Henry Houh, Ph.D. Under 37 C.F.R. § 1.68 in Support of
Petition for *Inter Partes* Review of U.S. Patent Nos. 9,367,657

- Verizon Wireless v. Barkan Wireless IP Holdings LP, submitted IPR declarations
- Verizon Wireless and Samsung Electronics America Inc. v. Barkan Wireless IP Holdings LP, IPR2019-199, IPR2019-200, IPR2019-234, IPR2019-631, IPR2019-632, submitted IPR declarations
- Samsung Electronics America Inc. v. Barkan Wireless IP Holdings LP, IPR2019-100, submitted IPR declaration
- Cisco Systems Inc. v. Meetrix IP LLC, IPR2019-539, IPR2019-540, IPR2019-541, IPR2019-542, IPR2019-543, IPR2019-544, submitted IPR declarations
- GMG Products LLC v. Traeger Pellet Grills LLC, PGR2019-24, PGR2019-34, PGR2019-35, PGR2019-36, submitted PGR declaration and was deposed
- Uniloc 2017 LLC v. Google LLC, 2:18-cv-00502-JRG-RSP, U.S. District Court for the Eastern District of Texas, Marshall Division, submitted declaration in support of claim construction
- Microsoft Corporation v. Uniloc 2017 LLC, IPR2020-00102, IPR2020-00103, submitted IPR declarations
- Microsoft Corporation and HP Inc. v. Syncloud Technologies, LLC, IPR2020-00316, IPR2020-01031, IPR2020-01032, IPR2020-01269, IPR2020-01270, IPR2020-01271, submitted IPR declarations
- Seven Networks, LLC v. Apple Inc, 2:19-cv-115-JRG, U.S. District Court for the Eastern District of Texas, Marshall Division, submitted declaration in support of claim construction
- Unified Patents Inc. v. Ortiz & Associates Consulting, LLC, IPR2019-00743, submitted IPR declaration
- Uniloc 2017 LLC v. Apple Inc., 3:19-cv-01905-JD, U.S. District Court for the Northern District of California, San Francisco Division, submitted declaration in support of claim construction and was deposed
- ClearPlay Inc. v. Dish Network LLC et al.; 2:14-cv-00191, U.S. District Court for the District of Utah, Central Division, submitted expert report and re-examination declarations, was deposed, and testified at trial
- AMC Networks, Inc. et al v. Sound View Innovations, LLC, IPR2020-00482, submitted IPR declaration
- Case No. 1:17-cv-01734-RGA, Sprint Communications Co., L.P., v. Charter Communications, Inc., et al., U.S. District Court for the District of Delaware, filed expert report and was deposed.
- CommScope Technologies LLC et al. v. Barkan Wireless IP Holdings LP et al., IPR2020-00827, IPR2020-00829, IPR2020-00831, IPR2020-00833, IPR2020-00835, IPR2020-00838, submitted IPR declarations
- Cisco Systems Inc. v. Monarch Networking Solutions LLC, IPR2020-01226, submitted IPR declaration
- CoolTV Network.com v. Blackboard, Inc., Facebook, Inc., International Business Machines Corporation, Kaltura, Inc., Limelight Networks, Inc., Microsoft Corporation, Ooyala, Inc., Snap, Inc., and Trapelo Corp., submitted declaration in support of claim construction and was deposed
- Facebook, Inc. v. Onstream Media Corporation, IPR2020-01507, IPR2020-01508, IPR2020-01525, IPR2020-01527, IPR2020-01528, submitted IPR declarations

Declaration of Henry Houh, Ph.D. Under 37 C.F.R. § 1.68 in Support of
Petition for *Inter Partes* Review of U.S. Patent Nos. 9,367,657

- Hisense Co LTD. v. Maxell, LTD, IPR2020-01598, IPR2020-01600, submitted IPR declarations
- Microsoft Corp. v. O'Brien, IPR2021-00015, submitted IPR declaration and was deposed
- Sable Networks Inc. v. Cisco Systems, Inc., WA:20-CV-00288-ADA, U.S. District Court for the Western District of Texas, Waco Division, submitted declaration in support of claim construction
- Peloton Interactive, Inc. v. Icon Health and Fitness Inc et al., IPR2021-00342, submitted IPR declaration
- Hulu, LLC v. SITO Mobile R&D IP, LLC et al., CBM2020-00028, IPR2020-00158, IPR2020-00206 , IPR2020-00219 , IPR2020-00265 , IPR2020-00298 , IPR2020-00304 , IPR2020-00308 , submitted CMB and IPR declarations and was deposed
- Twitter, Inc. and Google LLC v. B.E. Technology, LLC, IPR2021-00482, IPR2021-00483, IPR2021-00484, IPR2021-00485, submitted IPR declarations
- MG Freesites LTD v. ScorpCast, LLC, IPR2021-00511, submitted IPR declaration
- Avionte, LLC v. Sandeep Acharya, Samar Basnet, and Aqore LLC, Arbitration, Hon. Jeffrey Keyes (Ret.)
- Microsoft Corporation v. WSOU Investments, LLC d/b/a Brazos Licensing and Development, IPR2021-00751, submitted IPR declaration
- SAP America, Inc. v. Express Mobile, Inc., IPR2021-01144, IPR2021-01145, IPR2021-01146, submitted IPR declarations
- StratosAudio Inc. v. Hyundai Motor America, 6:20-cv-01125-ADA, U.S. District Court for the Western District of Texas, Waco Division, submitted claim construction declaration and was deposed Peloton Interactive, Inc. v. Ifit Health and Fitness Inc, IPR2022-00029, IPR2022-00030, submitted IPR declarations and was deposed
- Investigation No. 337-TA-1263, In the matter of Certain televisions, remote controls, and components thereof, U.S. International Trade Commission, filed expert reports, was deposed, and testified at hearing
- Alexander Stross v. Netease, Inc., et al., U.S. District Court (Central District of California), Case 2:20-cv-00861-AB(JC), filed expert reports, and was deposed
- Ralph Ledergerber v. Netease, Inc., et al., U.S. District Court (Central District of California), Case 2:20-cv-00862-AB(JC), filed expert reports and was deposed
- John Walmsley v. Netease, Inc., et al., U.S. District Court (Central District of California), Case 2:20-cv-00863-AB(JC), filed expert reports and was deposed
- Bernhard Kuhmstedt v. Netease, Inc., et al., U.S. District Court (Central District of California), Case 2:20-cv-02044-AB(JC), filed expert reports and was deposed
- Netflix, Inc. v. CA, Inc., IPR2021-01319, submitted IPR declaration and was deposed
- F45 Training Pty Ltd. v. Body Fit Training USA Inc., C.A. No. 20-1194 (LPS) (D. Del.), filed expert report
- Apple v. Fintiv, IPR2022-00976, IPR2022-01149, IPR2022-01150, IPR2023-00399, submitted IPR declarations and was deposed
- Estech Systems, Inc. v. Howard Midstream Energy Partners d/b/a Howard Energy Partners, C.A. No. 6:20-cv-777-ADA (Western District of Texas, Waco Division), filed expert reports and was deposed
- Echelon Fitness Multimedia, LLC. v. Peloton Interactive, Inc., IPR2020-01187, submitted IPR declaration and was deposed

Declaration of Henry Houh, Ph.D. Under 37 C.F.R. § 1.68 in Support of
Petition for *Inter Partes* Review of U.S. Patent Nos. 9,367,657

- VIZIO. V. Maxell Ltd., IPR2022-01458, submitted IPR declaration
- Maxell, Ltd. et. al. v. VIZIO, Inc., Case No. 2-21-cv-06758 (C.D. Cal.). filed declaration and was deposed
- Microsoft Corporation v. SurfCast, Inc., submitted IPR declarations
- The Walt Disney Company et. al. v. WAG Acquisition LLC, IPR2022-01227, IPR2022-01228, IPR2022-01346, submitted IPR declarations and was deposed
- Rakuten Rewards et. al. v. IBM, submitted IPR declarations and was deposed
- Fubotv Media Inc. v. Sito Mobile R&D IP, LLC et. al., submitted IPR declarations
- Unified Patents, LLC v. Noblewood IP LLC, IPR2022-01111, submitted IPR declaration
- Unified Patents, LLC. v. Pedersen, Peter Henrik, IPR2023-00029, submitted IPR declaration
- Ericsson Inc v. Dali Wireless Inc., submitted IPR declarations
- Universal Electronics, Inc. et. al. v. Roku, Inc., IPR2022-00818, submitted declarations and was deposed
- Google LLC. v. Wildseed Mobile LLC et. al., IPR2023-00244, IPR2023-00245, IPR2023-00246, IPR2023-00247, IPR2023-00248, submitted declarations
- Lululemon Athletica Canada Inc. et. al. v. Nike, Inc., IPR2023-00180, IPR2023-00424, submitted declarations
- Zoho Corporation, et. al. v. Meetrix IP, LLC, IPR2023-00371, IPR2023-00377, IPR2023-00379, IPR2023-00380, IPR2023-00382, submitted declarations
- WSOU Investments, LLC d/b/a/ Brazos Licensing and Development v. Google LLC, 6:20-cv-572, U.S. District Court for the Western District of Texas, Waco Division, submitted reports and was deposed
- Google LLC v. Safecast Limited LLC, IPR2023-00652, submitted declaration
- Investigation No. 337-TA-1338, In the matter of Certain smart televisions, U.S. International Trade Commission, filed reports and was deposed
- Cisco Systems, Inc. v. Orckit IP LLC, IPR2023-00714, submitted declaration
- SupplyPro, Inc. v. Sandvik Machining Solutions AB, et. al., AAA Case No. 01-22-000-7897. submitted declaration
- TrackThings LLC v. NETGEAR, Inc., Case No. 22-00981-RGA-JLH, U.S. District Court for the District of Delaware, submitted declaration
- Walt Disney Parks & Resorts US Inc. v. Agile Journeys LLC; IPR2024-00029, submitted declaration
- Aylo Freesites Ltd v. Dish Technologies LLC; IPR2024-00043, IPR2024-00044, IPR2024-00045, IPR2024-00046, IPR2024-00047, IPR2024-00048, IPR2024-00146, IPR2024-00147; submitted declarations

Publications

- “IP switching: server driven flow classification,” H. H. Houh, *Proceedings of the Washington University Workshop on Integration of IP and ATM*, November 1996.
- “Aurora at MIT,” D. D. Clark, H. H. Houh, and D. L. Tennenhouse, Editors, *MIT Laboratory for Computer Science Technical Report 673*, December 1995.

Declaration of Henry Houh, Ph.D. Under 37 C.F.R. § 1.68 in Support of
Petition for *Inter Partes* Review of U.S. Patent Nos. 9,367,657

- “ViewStation Applications: Implications for Network Traffic,” C. J. Lindblad, D. Wetherall, W. Stasior, J. F. Adam, H. H. Houh, M. Ismert, D. Bacher, B. Phillips, and D. L. Tennenhouse, *IEEE Journal of Selected Areas in Communications*, 1995.
- “The VuNet Desk Area Network: Architecture, Implementation, and Experience,” H. H. Houh, J. F. Adam, M. Ismert, C. J. Lindblad, and D. L. Tennenhouse, *IEEE Journal of Selected Areas in Communications*, 13 (4), May, 1995.
- “Reducing the Complexity of ATM Host Interfaces,” H. H. Houh and D. L. Tennenhouse, Hot Interconnects II Symposium Proceedings, Stanford, August 11-12, 1994.
- “Media-intensive data communications in a ‘desk-area’ network,” J. F. Adam, H. H. Houh, M. Ismert, and D. L. Tennenhouse, *IEEE Communications*, August 1994.
- “ViewStation Applications: Intelligent Video Processing Over A Broadband Local Area Network,” C. J. Lindblad, D. J. Wetherall, W. Stasior, B. Phillips, D. Bacher, J. Adam, H. Houh, M. Ismert, and D. L. Tennenhouse, Proceedings of the 1994 USENIX Symposium on High-Speed Networking, Oakland, CA, August 1994.
- “The Media Gateway: Live Video on the World Wide Web,” H. H. Houh, C. J. Lindblad, J. Soo, D. L. Tennenhouse, and D. J. Wetherall, Workshop at the 1994 World Wide Web Conference, Geneva, Switzerland, May 1994.
- “Active Pages: Intelligent Nodes on the World Wide Web,” H. H. Houh, C. J. Lindblad, and D. J. Wetherall, Proceedings of the 1994 World Wide Web Conference, Geneva, Switzerland, May 1994.
- “Wavelength Division vs. Code Division Access Methods for Optical Networks,” H. H. Houh, Area Exam Paper, May 1993.
- “Experience with the VuNet: A Network Architecture for a Distributed Multimedia System,” J. F. Adam, H. H. Houh, D. L. Tennenhouse, Proceedings of the 18th Conference on Local Computer Networks, pp. 70-76, September 1993
- “The VudBoard: A Simple DMA Interface,” H. H. Houh, Proceedings of the 4th Gigabit Minijam, January 1994.
- “A Software-Oriented Approach to the Design of Media Processing Environments,” D. L. Tennenhouse, J. Adam, D. Carver, H. Houh, M. Ismert, C. Lindblad, W. Stasior, D. Weatherall, D. Bacher, and T. Chang., submitted to the International Conference on Multimedia Computing and Systems, May 1994.
- “A Network Architecture for Distributed Multimedia Systems,” J. F. Adam, H. H. Houh, M. Ismert, and D. L. Tennenhouse, submitted to the International Conference on Multimedia Computing and Systems, May 1994.
- “The Viewstation Collected Papers,” D. L. Tennenhouse, J. Adam, C. Compton, A. Duda, D. Gifford, H. Houh, M. Ismert, C. Lindblad, W. Stasior, R. Weiss, D. Wetherall, D. Bacher, D. Carver, and T. Chang, MIT Laboratory for Computer Science Technical Report, MIT/LCS/TR-590, November 1993.
- “The Viewstation Collected Papers II,” W. F. Stasior, D. L. Tennenhouse, J. F. Adam, D. R. Bacher, H. H. Houh, M. Ismert, C. J. Lindblad, B. M. Phillips, D. J. Wetherall, MIT Laboratory for Computer Science Technical Report, MIT/LCS/TR-696, April 1996.
- “A System's Perspective of the Sagnac Fiber Logic Gates and Their Possible Applications,” A. Huang, N. Whitaker, C. Gabriel, H. Avramopoulos, P. M. W. French, H. H. Houh, and I. Chuang, *Applied Optics*, September 10, 1994

Declaration of Henry Houh, Ph.D. Under 37 C.F.R. § 1.68 in Support of
Petition for *Inter Partes* Review of U.S. Patent Nos. 9,367,657

- “Complete Switching in a Three-Terminal Sagnac Switch,” H. Avramopoulos, P. M. W. French, M. C. Gabriel, H. H. Houh, N. A. Whitaker, T. Morse, IEEE Phot. Tech. Lett. **3** (3), 235
- “Complete Switching in a Three-Terminal Sagnac Switch,” H. Avramopoulos, P. M. W. French, M. C. Gabriel, H. H. Houh, N. A. Whitaker, IEEE/LEOS Annual Meeting, Paper PDP-13, November 1990
- “All-optical phase-locked oscillator,” N. A. Whitaker, Jr., H. H. Houh, H. Avramopoulos, T. F. Morse, IEEE/LEOS Annual Meeting, Paper ELT2.4/MOO3, November 1990
- “Passive modelocking of an all-fiber erbium laser,” H. Avramopoulos, H. H. Houh, N. A. Whitaker, M. C. Gabriel, T. F. Morse, IEEE/LEOS Conference on Optical Amplifiers and their Applications, Paper PDP-8, August 1990
- “Transverse modes, waveguide dispersion, and 30ps recovery in submicron GaAs/AlAs microresonators,” J. L. Jewell, S. L. McCall, A. Scherer, H. H. Houh, N. A. Whitaker, A. C. Gossard, and J. H. English, Appl. Phys. Lett. **55** (1), July 3, 1989

Leadership, Activities and Interests

- Leadership
 - Reading Symphony Orchestra
 - Board of Directors (Treasurer), 2022 - present
 - Discovery Museums (Acton, MA)
 - Board of Directors, 2023 – present
 - Board of Directors, Development Committee, 2023 – present
 - Science and Technology Advisory Council, 2012 - present
 - Tau Beta Pi National Engineering Honor Society
 - Executive Councilor Elect, to serve 2024-2026
 - Advisor, MA B Chapter at MIT, 2003 - present
 - District Director (National Officer), Tau Beta Pi, New England Area, 1991 - 2003
 - President, MA B Chapter at MIT, Fall 1988 - Spring 1989
 - Laureate award, 1989
 - MIT Alumni Association Board of Directors
 - National Selection Committee, 3 year term, 2020 – 2023
 - K-12 STEM Initiatives Co-chair, 2013 – 2017
 - Awards Committee Chair, 2012 – 2014
 - Awards Committee, 3 year term, 2011 – 2014
 - Vice President, 2 year term, 2004 – 2006
 - Alumni Ad-Hoc Committee, Chair, 2005-2006
 - Board Member, 2 year term, 1997 – 1999
 - MIT Club of Boston
 - Board of Directors, 2006 - 2011
 - K-12 Initiatives Chair, 2009 - 2012
 - VP of Communications, MIT Club of Boston, 2003 - 2006
 - Past-President, MIT Club of Boston, 2002 - 2003
 - President, MIT Club of Boston, 2001 - 2002
 - President-Elect, MIT Club of Boston, 2000 - 2001
 - VP of Programs, MIT Club of Boston, 1999 - 2000
 - Activities Super-Chair, MIT Club of Boston, 1998 - 1999
 - MIT Enterprise Forum of Cambridge, Inc.
 - Past Chair, 2009 – 2011
 - In-NOW-vation Co-chair, 2010
 - Chair, 2007 - 2009
 - Vice Chair, 2005 - 2007
 - Executive Board Member, 2002 - 2011
 - Winter Workshop Co-Chair, February 2007 - conceived idea for conference, which sold-out and produced largest attendance numbers in recent memory
 - Spring Workshop Co-Chair, Spring 2004
 - Membership Committee Chair, Fall 2003 - 2006

Declaration of Henry Houh, Ph.D. Under 37 C.F.R. § 1.68 in Support of
Petition for *Inter Partes* Review of U.S. Patent Nos. 9,367,657

- 25th Anniversary Dinner Chair, Fall 2003
- As Membership Chair and Board Member, started Special Interest Groups in 2004; a SIG won the MIT Presidential Citation award, the MIT Alumni Association's highest award for organizations, in 2006
- Estabrook Elementary School PTA
 - Advisory committee to the superintendent on PCB issue, 2010-2011
 - 4th Grade after-school science program co-organizer, 2010-2012
 - 4th and 5th Grade before-school Math Olympiad co-organizer, 2009-2013
 - 5th Grade BBQ and Yearbook Committee, 2011, 2013
 - Family Math Night volunteer, 2008-2012
- MIT Class of 1989
 - Secretary, five consecutive 5 year terms, 1989 - 2014
 - 30-year Reunion Committee, 2019
 - 25-year Reunion Committee, 2014
 - 20-year Reunion Committee and Gift Committee, 2009
 - 15-year Reunion Committee and Gift Committee, 2004
 - 10-year Reunion Committee and Gift Committee, 1999
 - 5-year Reunion Committee, 1994
 - Interim Treasurer, 1993 - 1994
 - Instituted annual senior class career fair, now raising over \$100,000 annually for senior class activities, Fall 1988
- Strong, consistent record of leadership dating to high school
- Acting
 - '21' (Sony Pictures), credited as "Chinatown Dealer," 2007, Kevin Spacey's movie about the MIT Blackjack Team inspired by "Bringing Down the House" by Ben Mezrich, opened nationwide on March 28, 2008. 21 was the number one movie in US for two weeks and number one globally for one week. 21 also topped the DVD sales, Blu-ray sales and DVD rental charts.
 - Spring Lake Theater Company, first New York-area off-broadway production of "A Chorus Line," played role of Mark, Summer 1990
- Former member of the MIT Blackjack team
- Producer for 10,000 Maniacs' 2013 album "Music from the Motion Picture"
- Executive Producer for 10,000 Maniacs' 2015 album "Twice Told Tales"
- Violist, violist, harpist, guitarist, singer, actor: played in many amateur/semi-professional groups including Firebird Pops Orchestra, Reading Symphony Orchestra, MIT Summer Philharmonic Orchestra, Carlisle Chamber Orchestra, Rivers Symphony Orchestra, Opera51 Orchestra, Freisinger Chamber Orchestra, Merrimack Valley Philharmonic, Longwood Symphony, MIT Symphony, and Somerville Community Chorus
- Violist, "Coming Home" album by Karen Phillips, 2014
- Google Scholar citations: 2079, h-index: 21, i10-index: 26 (since 2017: 662/12/14)
- Erdos-Bacon number of 7
- Erdos-Bacon-Sabbath number of 11