

IN THE 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. 8,924,899

Inter Partes Review No.: IPR2024-00476

PETITION FOR *INTER PARTES* REVIEW OF U.S. PATENT NO. 8,924,899

Mail Stop “PATENT BOARD”
Patent Trial and Appeal Board
U.S. Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

TABLE OF CONTENTS

	Page
I. INTRODUCTION	1
II. OVERVIEW	1
III. GROUNDS FOR STANDING (37 C.F.R. § 42.104(A)).....	8
IV. RELIEF REQUESTED (37 C.F.R. § 42.22(A))	9
V. REASONS FOR THE REQUESTED RELIEF	9
A. ‘899 Patent Summary	9
B. Prosecution History And Priority Date.	11
C. Claim Construction	11
D. Person of Ordinary Skill in the Art (“POSITA”).....	11
VI. IDENTIFICATION OF CHALLENGES.....	12
A. Challenged Claims	12
B. Statutory Grounds for Challenges.....	12
VII. IDENTIFICATION OF HOW THE CHALLENGED CLAIMS ARE UNPATENTABLE.....	13
A. The Challenged Claims’ Structure.	13
B. The State of the Art.	13
C. Ground 1: The Challenged Claims Are Obvious Over Rompaey Alone Or In Combination With CoWare I.....	16
1. Overview of Rompaey	16
2. Rompaey Discloses, And/ Or Renders Obvious, The Challenged Claims Alone Or in Combination With CoWare I	18
a. Claim 1.....	18

Preamble: “A computer implemented method for designing an integrated circuit which controls a device when the integrated circuit integrated into the device comprising”	18
i. 1[a] “receiving a selection of categorized input and output characteristics of an electronic device into a memory of a processor programmed by software commands to design a hardware integrated circuit and associated operating software for controlling the integrated circuit to control the electronic device”	18
ii. 1[b]: “analyzing by the processor the selection of categorized input and output characteristics of the electronic device”	40
iii. 1[c] “outputting by the processor of a hardware integrated circuit design based on the selection of input and output characteristics of the device; and”	47
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.”	48
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 input or output characteristics.”	49
c. Claim 6: “The method of claim 5, wherein the hardware integrated circuit design is provided as a software schematic, layout, or both.”	50
d. Claim 8: Preamble “A system comprising:”	51

i.	8[a] “one or more processors;”	51
ii.	8[b] “logic encoded in one or more non-transitory computer-readable media that, when executed by the one or more processors, is operable to.”	51
iii.	8[c]: “receive a selection of categorized input and output characteristics of a device”	52
iv.	8[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”	53
v.	8[e] “output software programming commands to executed [sic] by the hardware integrated circuit for controlling the device in accordance with the selection of categorized input and output characteristics of the device using a communication protocol compatible with the device through the communication bus of the hardware integrated circuit.”	54
e.	Claim 14: “The system of claim 8, 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”	54
D.	Ground 2: The Challenged Claims Are Obvious Over Banerjee.....	57
1.	Overview of Banerjee	57

2.	Banerjee Discloses, And/ Or Renders Obvious, The Challenged Claims.	60
a.	Claim 1: Preamble:	60
i.	Element 1[a]	60
ii.	Element 1[b]	72
iii.	Element 1[c]	77
iv.	Element 1[d]	78
b.	Claim 5.....	80
c.	Claim 6.....	80
d.	Claim 8: Preamble	81
i.	Element 8[a]	81
ii.	Element 8[b]	82
iii.	Element 8[c]	82
iv.	Element 8[d]	82
v.	Element 8[e]	83
e.	Claim 14.....	84
VIII.	THE BOARD SHOULD NOT DENY INSTITUTION UNDER SECTION 314	85
A.	The Board Should Not Deny Institution Under 35 U.S.C. § 325(d)	85
B.	The Board Should Not Deny Institution Under 35 U.S.C. § 314(a).....	85
IX.	MANDATORY NOTICES UNDER 37 C.F.R. §42.8.....	88
A.	Real Party-in-Interest (37 C.F.R. § 42.8(b)(1)).....	88
B.	Related Matters (37 C.F.R. § 42.8(b)(2)).....	89

1.	Judicial Matters	89
2.	Administrative Matters:	89
3.	Related Patents.....	89
C.	Lead/Back-up Counsel (37 C.F.R. § 42.8(b)(3))	89
D.	Notice of Service Information (37 C.F.R. § 42.8(b)(4))	90
E.	Fees Under 37 C.F.R. § 42.103	90
X.	CONCLUSION.....	91
	CERTIFICATION OF SERVICE ON PATENT OWNER.....	1
	CERTIFICATE OF COMPLIANCE.....	3

Petitioner’s Exhibit List

<i>Exhibit #</i>	<i>Description</i>
1001	U.S. Patent No. 8,924,899 (“the ‘899 patent”)
1002	Prosecution History for U.S. Patent No. 8,924,899 (downloaded from PAIR)
1003	Declaration of Dr. Henry Houh.
1004	U.S. Patent No. 5,870,588 (“Rompaey”)
1005	U.S. Pat. Appl. No. 2008/0222581 (“Banerjee”)
1006	<i>CoWare—a design environment for heterogeneous hardware/software systems</i> , K. Van Rompaey, D. Verkest, I. Bolsens and H. De Man, Proceedings EURO-DAC ’96, Geneva, Switzerland, pp. 252-257 (1996).
1007	<i>CoWare—A design environment for heterogeneous hardware/software systems</i> , Verkest, D., Van Rompaey, K., Bolsens, I. et al. Des Autom Embed Syst 1, 357–386 (1996).
1008	ICPillar Preliminary Infringement Contentions, ‘899 Patent.
1009	U.S Pat. No. 5,197,016 (“Sugimoto”)
1010	U.S. Patent No. 4,922,432 (“Kobayashi”)
1011	Fletcher, William I., <i>An Engineering Approach to Digital Design</i> , Prentice-Hall, Inc., pp. 491-505 (1984) (excerpts)
1012	ICPillar Opening Claim Construction Brief
1013	Arm Opening Claim Construction Brief.
1014	Complaint in <i>ICPillar v. ARM Ltd. et al</i> , 1:23-cv-282 (WDTX)
1015	Scheduling Order in <i>ICPillar v. ARM Ltd. et al</i> , 1:23-cv-282 (WDTX)

<i>Exhibit #</i>	<i>Description</i>
1016	Pacer Docket in <i>Sable Networks, Inc. et al v. Cloudflare, Inc.</i> , case 6:21-cv-00261, WDTX.
1017	Pacer Docket in <i>Demaray LLC v. Samsung Electronics Co., Ltd.</i> , case 6:20-cv-00636, WDTX.

I. INTRODUCTION

Pursuant to 35 U.S.C. §§ 311 *et seq.* and 37 C.F.R. §§ 42.1 *et seq.*, Arm Limited (“Petitioner”) hereby petitions for an *inter partes* review (“IPR”) of U.S. Patent No. 8,924,899 (“the ‘899 patent”) owned by ICPillar LLC (“Patent Owner” or “PO”). Petitioner respectfully submits that Claims 1, 5, 6, 8, and 14 (the “Challenged Claims”) of the ‘899 patent are unpatentable under 35 U.S.C. § 103. Accordingly, Petitioner respectfully requests that the Board institute an IPR pursuant to 37 C.F.R. § 42.108.

II. OVERVIEW

The Challenged Claims are invalid over decades of tools for designing integrated circuits. The actual invention of the ‘899 patent relates to a “system and method for universal control of electronic devices.” EX-1001, Title. The Background is devoted to controlling devices remotely, such as a “television and its remote controller.” EX-1001, 1:6-20. The disclosed invention (a user interface for a “controller”) is recited by *some* claims (*e.g.*, claims 2-4), but the Challenged Claims omitted this invention and recite elements that, under PO’s view, relate merely to the design of hardware circuits and associated software.

Challenged Claim 1 recites “[a] computer implemented method for designing an integrated circuit which controls a device when the integrated circuit integrated into the device....” Given that the ‘899 patent was filed in May 2013 and that the

electronics industry had been using hardware / software design tools since at least the 1980's (*infra* at Section VII.B), it is unsurprising that many prior art references invalidate the Challenged Claims – especially under the PO's broad view of the claims.

Challenged Claim 1 recites four steps. First, there is a “receiving a selection of categorized input and output characteristics of an electronic device into a memory” EX-1001, claim 1. In the ‘899 patent, a “developer can select the functionality of their device” to select the characteristics. EX-1001, 4:35-45. The types of device functionality can be selected from a library. “[T]he 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.” EX-1001, 9:1-4. These types of functions correspond to the characteristics selected by the designer.

[T]he 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.

EX-1001, 14:34-37. Output functionality include “displays” and input functionality includes “buttons.”

Characteristics may include multiple types within each characteristic.

... Display type references the form of visual communication, such as

for example, alpha numeric, ... 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.

EX-1001, 14:37-50.¹ Thus the claimed selection of input and output characteristics corresponds to a designer specifying that a particular device has particular functionality such as a display (output characteristic) or buttons (input characteristic) and the components for implementation thereof.

As detailed below, the prior art herein includes a pager with a "display," keyboard, and control buttons that the designer selects as types of functions for the pager device and then designs the components implementing those functions.

Steps 2-4 in Challenged Claim 1 recite "analyzing" the designer's selection and then "outputting ... a hardware integrated circuit design" and "software" based on the selections. EX-1001, claim 1. The prior art herein details these steps as well.

This Petition focuses on two similar references. First, U.S. Pat. No. 5,870,588

¹ Unless otherwise noted, all emphases and annotations herein have been added.

(“Rompaey”) was filed in 1995 (eighteen years before the ‘899 patent) and teaches a software tool for co-developing hardware and software (“CoWare”) which a system designer uses to design devices (referenced as “systems” in Rompaey). EX-1004, 10:53-11:35. Rompaey teaches designing an exemplary device – a “pager” for which a designer selects both input functions / characteristics (from a keyboard and buttons) and output functions / characteristics (to a “display”).

A CoWare designer selects basic functional blocks of the designed device from various libraries and refines those functions into different components. For example, Rompaey’s Fig. 10 depicts a diagram of the pager components that the designer selected by specifying the relevant functionality of the device and its components. EX-1004, 21:33-22:35.

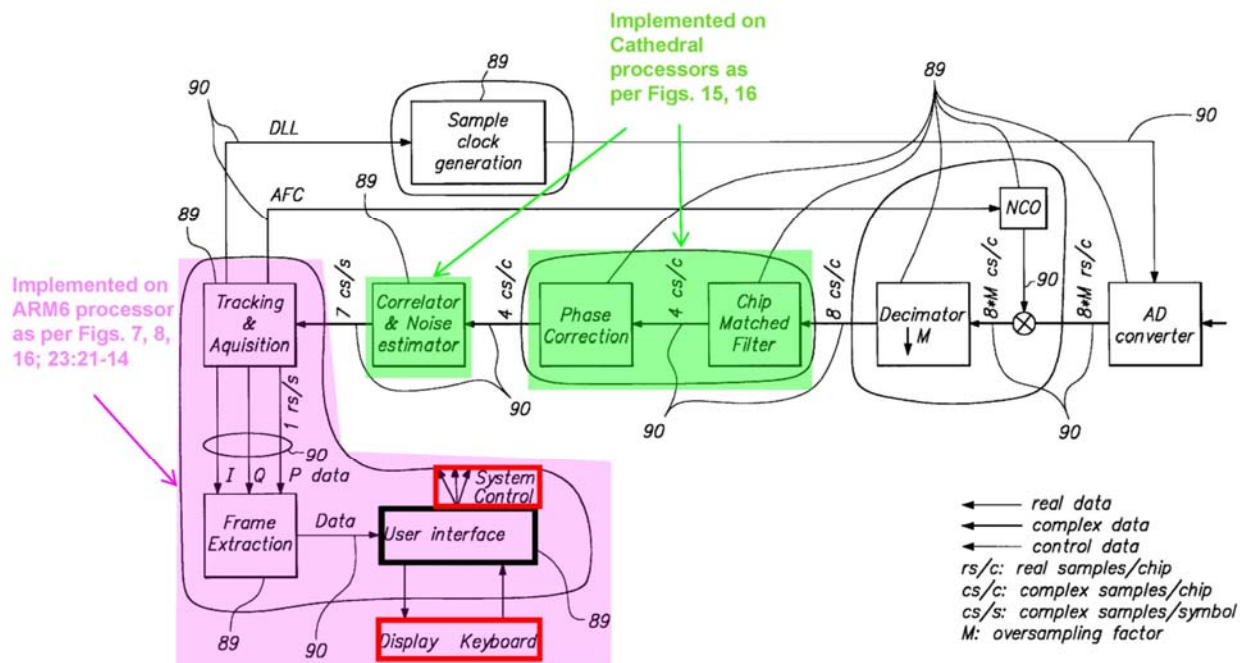
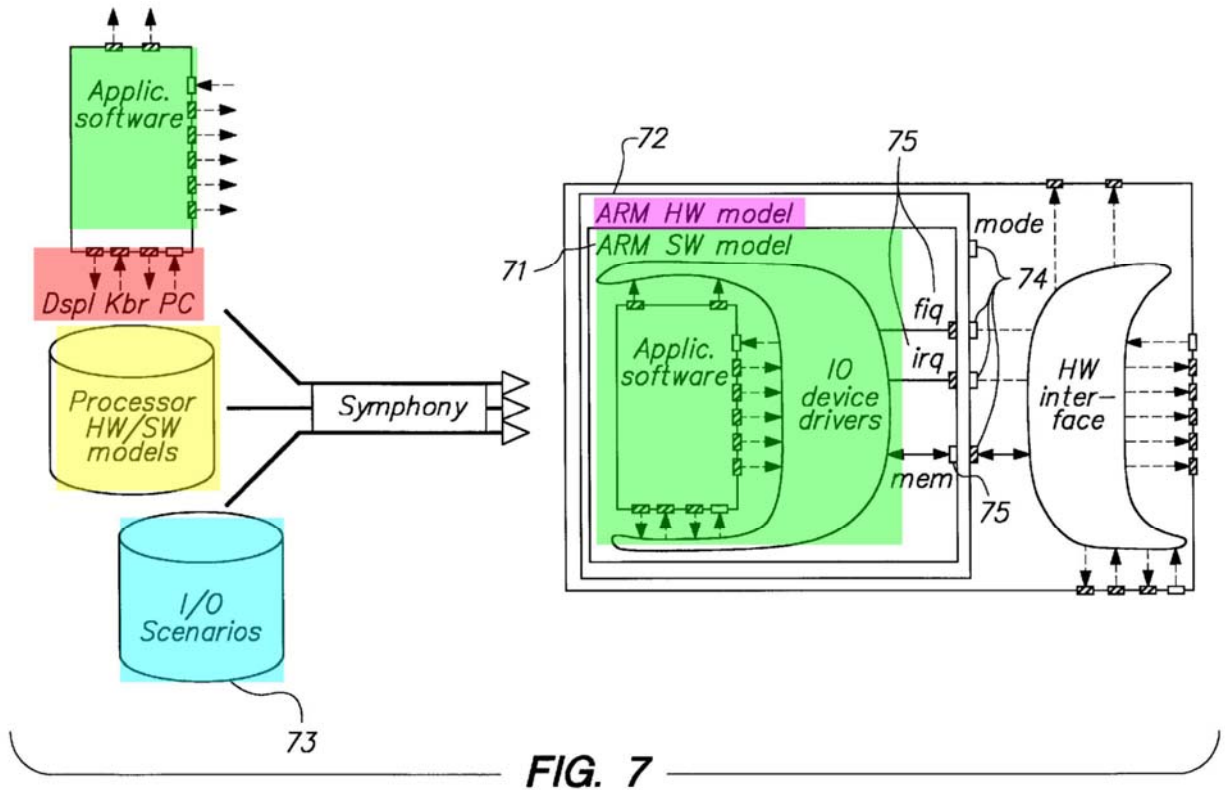


FIG. 10

EX-1004, Fig. 10.

The designer's selection includes selecting that the device will include types of functions such as outputting to a "Display" and inputting from a "Keyboard" (both as shown in red box above) which are shown as interfacing with the selected and designed "User Interface" (as shown in black outlined box above). The designer then selects from various categorized libraries design components that implement the selected functional characteristics (such as implementing a "user interface" block that interacts with the "display" and "keyboard.>"). EX-1004, Figs. 18, 6-8, 10, 15, 17; 21:35-22:67.

The CoWare system designer refines the device design by selecting further from categories of components to implement the device. For example, Fig. 7 below shows an implementation of the ARM6 processor components in **purple** in Figure 10 above. The design can select from categories of pre-defined "**I/O Scenarios**" to implement the inputs and outputs using a sub-portion of CoWare called "SYMPHONY."



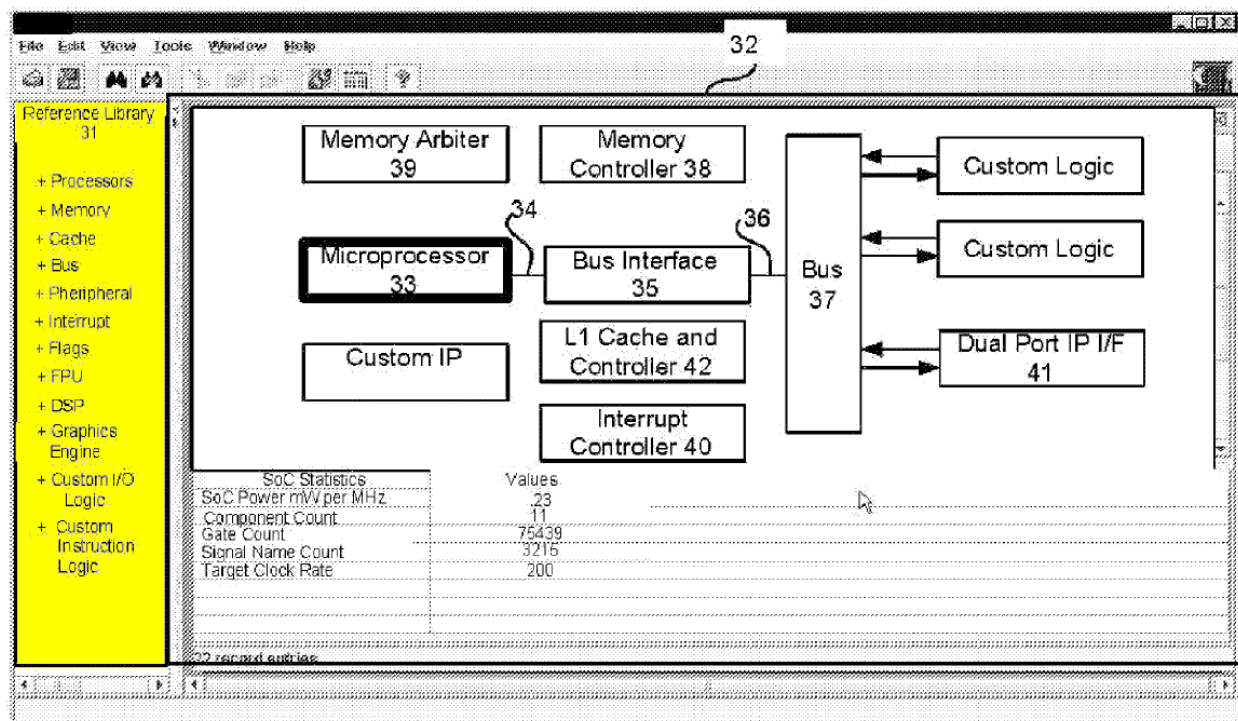
EX-1004, Fig. 7. “FIG. 7 illustrates the tool SYMPHONY. SYMPHONY makes use of ... a library of I/O scenarios (73) for that [selected] processor.” EX-1004, 17:65-18:12.

Ultimately, CoWare analyzes the selected system design and generates a hardware integrated circuit design and the associated software used to implement the designed device (*e.g.*, pager). The “user interface” (Fig. 10) controls device inputs / outputs such as the keyboard and display.

A second invalidating reference is U.S. Pat. Appl. No. 2008/0222581 (“Banerjee”) published in 2008. Banerjee teaches a software tool for designing and

creating both “application specific integrated circuits” (“ASICs”) and associated software to control devices.

Banerjee’s designer creates a “description [that] 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.” EX1005, ¶0033. Banerjee’s Figure 3 exemplifies the selection of functions / characteristics of the designed device.



EX-1005, Fig. 3. The designer selects from many categories of functions implemented by pre-defined components to design the desired system. After selecting from the top-level categories, the designer continues to use “sub-menus”

to select from sub-categories and to select “parameters” and “characteristics” of the desired components. EX-1005, ¶¶0055-0058.

Banerjee designer tailors hardware and software for particular devices and their functionality (referred to by Banerjee as “applications”). For example, Banerjee’s designer may tailor the system for devices that need multimedia applications or a device that “demands real-time processing of video or audio packets.” EX-1005, ¶¶0069, 0081. Banerjee’s selection of “video or audio” exactly matches the ‘899 description of the claimed input/output characteristics. EX-1001 at 14:37-47 (“the type of characteristic may include ...input and output ... input and output of the device may be audio, video....”).

Once Banerjee’s designer selects the functionality / characteristics of a designed application, Banerjee’s software tool then analyzes the designer’s selections and outputs the hardware circuit design and the related software to implement the designed system. Sections VII.C.2.a.ii—VII.C.2.a.iv, *infra*.

The Examiner did not consider the prior art cited herein. Petitioner respectfully requests that the Board institute trial and cancel the Challenged Claims.

III. GROUNDS FOR STANDING (37 C.F.R. § 42.104(A))

Petitioner certifies that the ‘899 patent is available for IPR and that Petitioner is not barred or estopped from requesting IPR of the Challenged Claims. 37 C.F.R. § 42.104(a). This Petition is filed pursuant to 37 C.F.R. § 42.106(a).

IV. RELIEF REQUESTED (37 C.F.R. § 42.22(A))

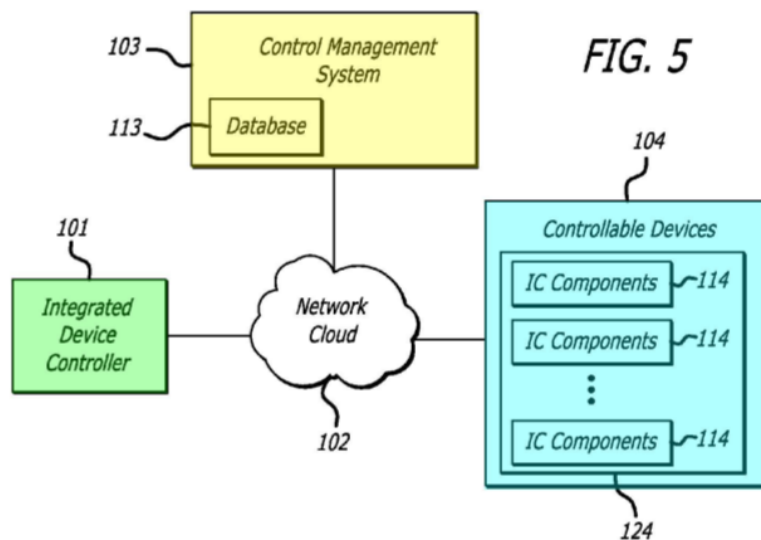
Petitioner respectfully requests institution of IPR pursuant to 37 C.F.R. § 42.108 and cancellation of the Challenged Claims.

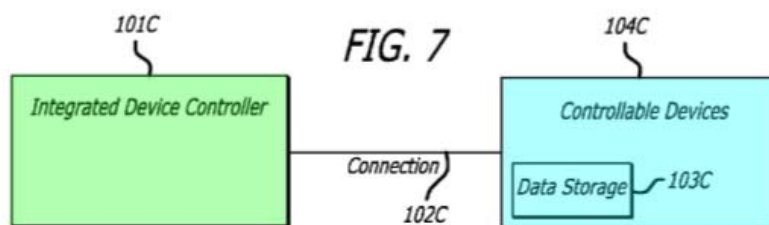
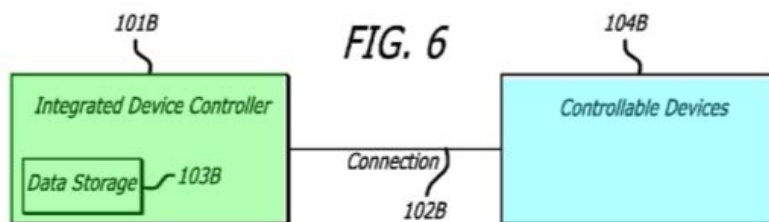
V. REASONS FOR THE REQUESTED RELIEF

As explained below and in the Declaration of Dr. Henry Houh (EX-1003), the Challenged Claims are disclosed by, and obvious over, the prior art as understood by a person of ordinary skill in the art (“POSITA”) as of the invention date.

A. ‘899 Patent Summary

The ‘899 patent sought to solve a problem of different devices requiring different types of remote controls. EX-1001, 1:6-3:30. The ‘899 patent proposes a solution that operates over a “network cloud” connection as shown in Figures 5-7 below.





EX-1001, Figs. 5-7. According to the ‘899 patent, a “**controller**” receives information from a “**control management system**” to be able to “control” different “**controllable devices**” across the internet. EX-1001, 10:40-11:10, 11:11-12:63 (describing Figs. 6, 7). This actual invention of the ‘899 patent (the “controller” and its operation) is recited in claims 2-4, 7 and 9-11 – which are not challenged herein.

Instead, the Challenged Claims recite functionality to “receiving a selection of categorized input and output characteristics of an electronic device” and then output hardware and software corresponding to the selections. Such functionality existed in the prior art for many years before the 2013 priority date of the ‘899 patent.

B. Prosecution History And Priority Date.

The application leading to the '899 patent was filed on May 23, 2013. Therefore, the “first-to-file” provisions of the America Invents Act apply. Thus, the applicable priority date is no earlier than May 23, 2013.

The Examiner rejected the original claims as unpatentable under Section 101 and anticipated under Section 102. EX-1002, 000057-00069. The applicant then amended the independent claims and argued that the cited prior art did not disclose outputting a hardware integrated circuit. EX-1002, 000044-000050. In response, the Examiner allowed the claims. EX-1002, 000016-000020.

C. Claim Construction

Unless otherwise defined, the claims should be given “their ordinary and customary meaning.” *Phillips v. AWH Corp.*, 415 F.3d 1303 (Fed. Cir. 2005). In the Litigation, both ICPillar and Arm have submitted claim construction briefs to the Court with proposals for some terms. EX-1012, EX-1013.

For purposes of this Petition, the parties’ competing claim constructions do not need to be resolved by the Board because the prior art herein invalidates the claims under either parties’ constructions and/or under any reasonable constructions.

D. Person of Ordinary Skill in the Art (“POSITA”)

A POSITA was familiar with automated circuit design. EX-1003, ¶¶46-47. A POSITA possesses this knowledge through training and work experience, including having a Bachelor’s or Master’s Degree in computer science, electrical

engineering, or a related field, with at least three years of practical experience.

Experience could substitute for the advanced degree. *Id.*

VI. IDENTIFICATION OF CHALLENGES

A. Challenged Claims

Claims 1, 5, 6, 8, and 14 of the ‘899 patent are challenged.

B. Statutory Grounds for Challenges

The Challenges are set forth in detail below and summarized as follows:

Ground	Claims	Basis	Reference
1	1, 5, 6, 8, 14	§ 103	Rompaey alone or in combination with CoWare I
2	1, 5, 6, 8, 14	§ 103	Banerjee

U.S. Patent No. 5,870,588 (“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).

U.S. Patent Application No. 2008-0222581 (“Banerjee”) was filed on March 9, 2007 and published on September 11, 2008. Banerjee is prior art under section 102(a) (post-AIA). Banerjee issued as U.S. Pat. No. 8,103,987 on January 14, 2012.

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) (“CoWare I”) (EX-1007).

The following patents and publications present the state of the art:

CoWare-a design environment for heterogeneous hardware/software systems, K. Van Rompaey *et al.*, Proceedings EURO-DAC '96, Geneva, Switzerland, 1996, pp. 252-257 (“CoWare II”) (EX-1006).

CoWare I and CoWare II published in 1996 and were submitted to the PTO during the prosecution of Rompaey (and were publicly available no later than 1999 when Rompaey issued in addition to publication noted above). CoWare I and CoWare II are prior art under 35 U.S.C. § 102(a).

VII. IDENTIFICATION OF HOW THE CHALLENGED CLAIMS ARE UNPATENTABLE

A. The Challenged Claims’ Structure.

The ‘899 patent has two independent claims: claim 1 recites a “method for designing an integrated circuit.” Claim 8 recites a corresponding “system.” EX-1001, 18:38-20:29. The dependent Challenged Claims recite known components and functionalities.

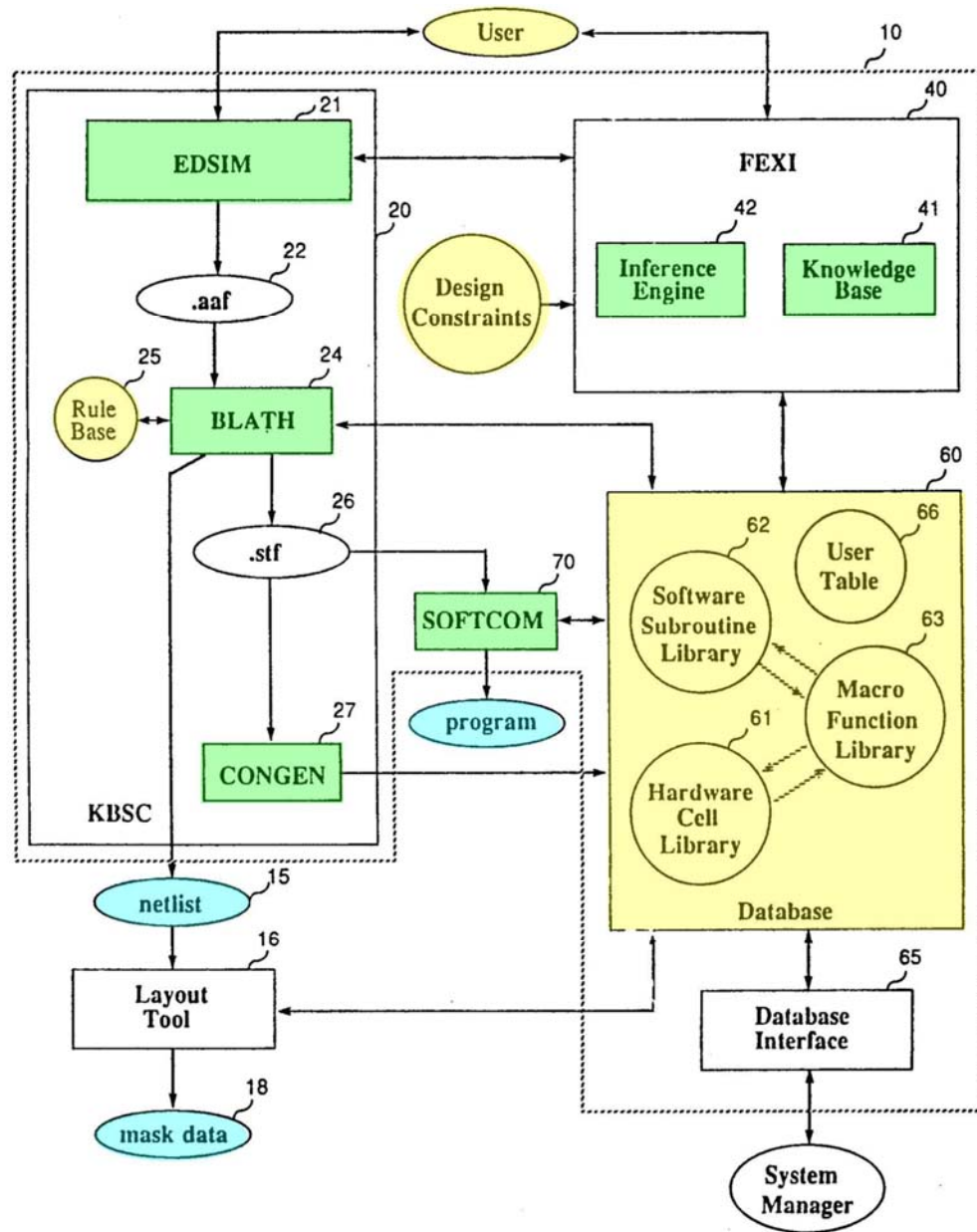
The discussion below focuses on claim 1 and its dependent claims. The analysis for claim 8 cites to the corresponding discussion for claim 1.

B. The State of the Art.

The state of the art for tools to develop hardware circuits and corresponding software was well-developed long prior to the ‘899 patent filing. The literature and commercially available products teach such tools at least back to the 1980’s. EX-1003, ¶¶56-89. The ‘899 patent acknowledges that “[f]abless design of ICs and

MCUs is well-known in the art.” EX-1001, 9:65. The ‘899 patent acknowledges prior art included “computer aided design software developed by Synopsis, Inc.; Cadence Design Systems, Inc.; Magma Design Automation” EX-1001, 13:13-16.

U.S Pat. No. 5,197,016 (“Sugimoto”) was filed in 1989. 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 “ISSC” system for designing an ASIC and its associated software in which 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.” EX-1009, 4:26-29.



EX-1009, Fig. 2. ISSC 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.

Sugimoto is a continuation-in-part of U.S. Patent No. 4,922,432 (“Kobayashi”) filed in 1988. Kobayashi has similar content and also presents its ASIC design in the context of a specific controlled device: “soft drink vending machine controller design example.” EX-1010, 12:37-14:30, Figs. 10-15.

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. The Fletcher passages referenced by Kobayashi show invalidating content from 1984, when Fletcher was published. EX-1011.

Thus, by May 2013, a POSITA would have long known that tools existed for system designers to select from categories of functional components to be used for the device, such as input and output components, to specify those components’ characteristics, and to have the tools generate and output the hardware integrated circuits and software to control the desired electronic device. EX-1003, ¶89.

C. Ground 1: The Challenged Claims Are Obvious Over Rompaey Alone Or In Combination With CoWare I.

1. Overview of Rompaey

Rompaey discloses or renders obvious the Challenged Claims.² Rompaey

² As used herein, “discloses” includes that the reference’s disclosure renders obvious a claim element and the claim as a whole.

describes a system called CoWare which designs hardware and software based on a designer's selected specifications. CoWare "allows one to specify, simulate, and synthesize heterogeneous hardware and software architectures from a heterogeneous specification... to represent a specification of an essentially digital system." EX-1004 at Abstract; *see also* EX-1004, 6:6-18.

Using CoWare, a system designer inputs a device specification, selects and specifies inputs of "models" from libraries with categorized hardware and software pre-existing components. The designer refines the device design interactively by selecting the functions, parameters and characteristics to implement the designed system / device. The system designer then uses CoWare to convert the design into hardware circuit designs and software. CoWare outputs the hardware integrated circuit design (using known formats such as VHDL and net lists) and the associated software that executes on the integrated circuit.

Rompaey's pager device example invalidates the Challenged Claims. The pager example shows a "device" for which the designer selects input and output characteristics (*e.g.*, a "display" and a "keyboard" as shown below). After the user selects the functionality and characteristics of the device, CoWare analyzes the selected system design, and further receives the selection of "Processor HW/SW models" and "I/O scenarios" to generate the hardware / software design. EX-1004, 26:42-43. CoWare then synthesizes and outputs a hardware circuit design and

software implementing the system.

Thus, Rompaey discloses what is claimed in the Challenged Claims.

2. Rompaey Discloses, And/ Or Renders Obvious, The Challenged Claims Alone Or in Combination With CoWare I

a. Claim 1

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

If the preamble is limiting, Rompaey discloses this element. CoWare designs integrated circuits – including the exemplary design of circuits and software controlling a “pager” device. *E.g.*, EX-1004 at 6:6-18; 8:4-48; 10:53-62; Figs. 1-2, 10-13; claim 6; EX-1003, ¶¶95-98.

If PO argues that Rompaey does not disclose any aspect of this claim, the discussion of the elements below explains why any such disputed aspect is obvious.

i. 1[a] “receiving a selection of categorized input and output characteristics of an electronic device into a memory of a processor programmed by software commands to design a hardware integrated circuit and associated operating software for controlling the integrated circuit to control the electronic device”

Rompaey describes three broad examples of this “receiving” element. EX-1003, ¶¶99-208. Rompaey’s designer specifies the functionality of the designed system (device) and the components (hardware and software) implementing that

functionality for that specific device.

The ‘899 patent describes selecting “characteristics” by a designer selecting that a particular device will have a desired functionality of a particular category:

[T]he 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.

EX-1001, 4:35-45. The categorized types of functions are stored in libraries. “[T]he 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.” EX-1001, 9:1-4. The ‘899 patent refers to selecting these **control/display/input/output** functions as selecting “characteristics.” “[T]he user may be provided with a selectable set of characteristics of the user’s device... the type of characteristics may include **control, display, input and output, communications**.” EX-1001, 14:34-37.

The ‘899 specification provides specific examples of input and output characteristics (buttons, displays) and a general statement of the characteristics.

Characteristics may include multiple types within each characteristic.... 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.

EX-1001, 14:37-50.

Rompaey selects functions that specify the characteristics of the device in the same way as in the ‘899 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 ‘899 patent references selecting that the device will have the function of a “display” and the components for the “display”).

EX-1003, ¶109.

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 selects the function for “a command or instruction to modify or initiate new action on the [pager]” as identified in the ‘899 patent for an “input.” EX-1001, 14:47-50. For example, the user pushes on the pager’s keyboard or pager control button – an

“input” characteristic / functionality akin to the ‘899 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, 9:7-8. EX-1003, ¶110.

Rompaey’s CoWare performs the claimed “receiving” of these characteristics using different ways of receiving data into the memory of the workstation that runs CoWare: (1) a designer-specified “description of the system;” (2) library files representing categories of reusable components mapped onto the system design; and (3) designer inputs with “tools” that refine the functions of the components within the designed device. EX-1003, ¶¶111- 192.

Figure 18 exemplifies CoWare’s system design process.

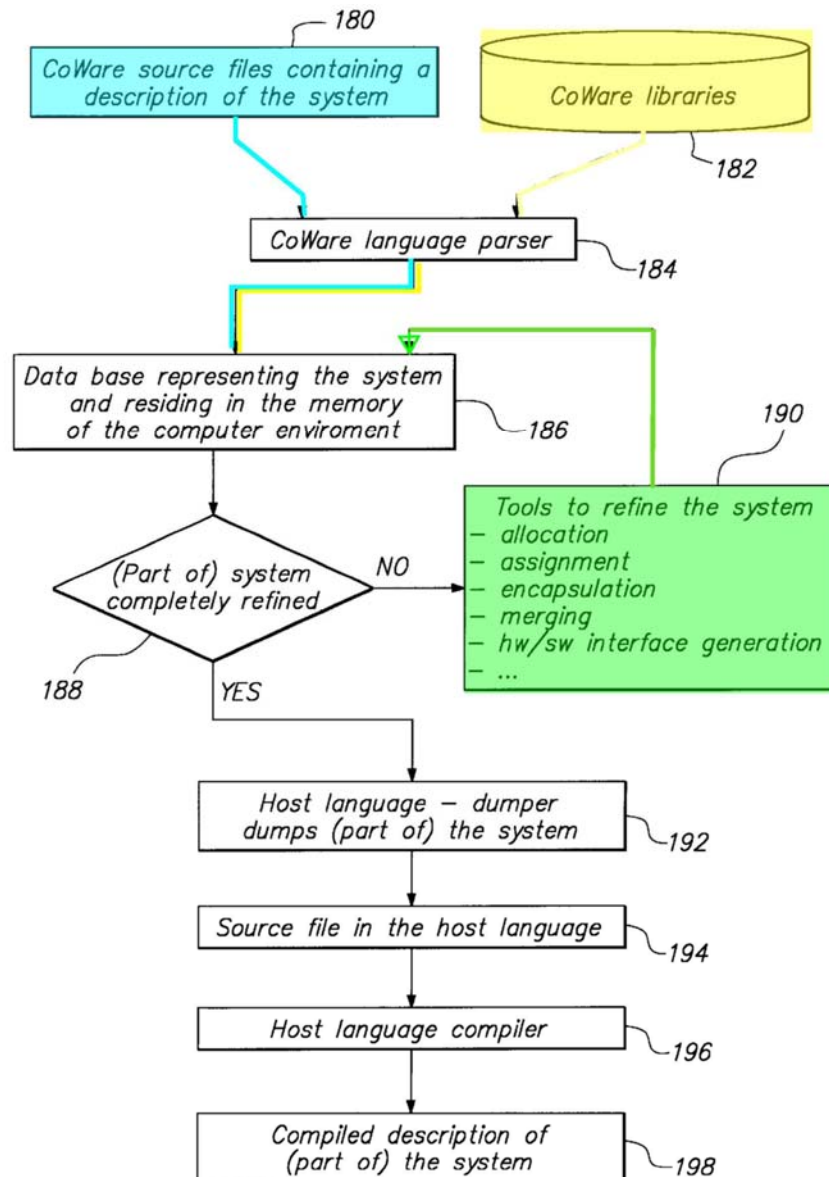


FIG. 18

EX-1004, FIG. 18.

FIG. 18 is an operational flowchart for a presently preferred method of specifying a hardware/software codesign 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. ... 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.

EX-1004, 26:50-27:3. The blue, green, and yellow blocks/passages are examples of distinct actions of the claimed “receiving.” The source files, libraries, database, and “classes for each object” are “hierarchical” and categorized. They have all been “received” into the “database representing the system and residing in the memory of the computer environment.” EX-1004, Fig. 18, element 186. EX-1003, ¶¶111-122.

Fig. 2 also depicts this process in showing the “co-specification” of the system, the “refinement and optimization” of the system, and the “IP Libraries.”

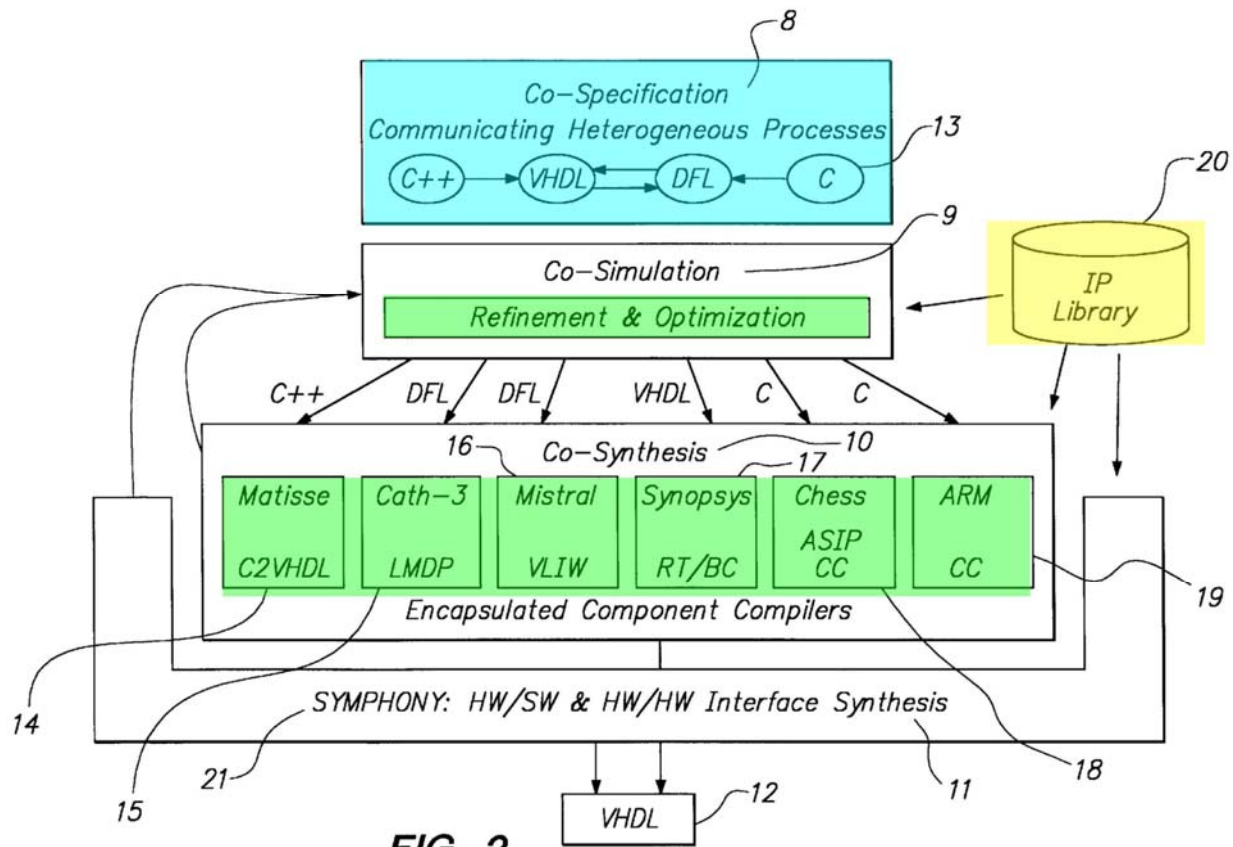


FIG. 2

EX-1004, Fig. 2.

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. EX-1004, 11:21-29. Ports and channels define the inputs and outputs of processes. Rompaey summarizes the designer’s specification options (11:21-12:5) and 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.

The hierarchical categorized objects are used to select and define the characteristics of the device, including input and output characteristics. For

example, as shown in Figure 4 below, the designer can select and define the characteristics of an “RS232” port. EX-1004, 14:58-15:49. RS-232 is a well-known input / output component. EX-1003, ¶¶133-134.

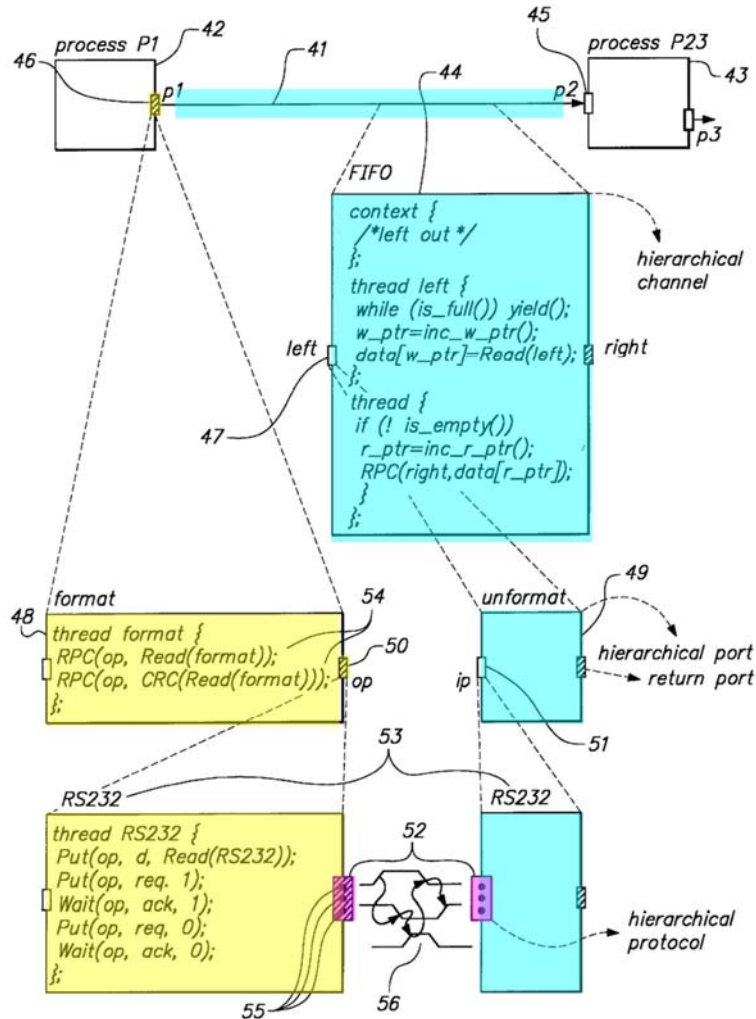


FIG. 4

EX-1004, Fig. 4. 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-characteristics 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).

EX-1004, 15:44-49. Thus, the designer has selected, and the CoWare tool has received, the claimed input and output characteristics by selecting the port (RS-232) expressed through the “terminals” that perform the input and output according to the sub-characteristics for that RS-232. 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 characteristics of the overall device. EX-1003, ¶¶135-148. *See also* EX-1004, Fig. 1 (element 5); 3:19-25.

Rompaey teaches designing a pager device in which the designer selects the input and output characteristics by selecting that the device will include input / output functions of a display and keyboard and the details of the implementation of those functions.

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.

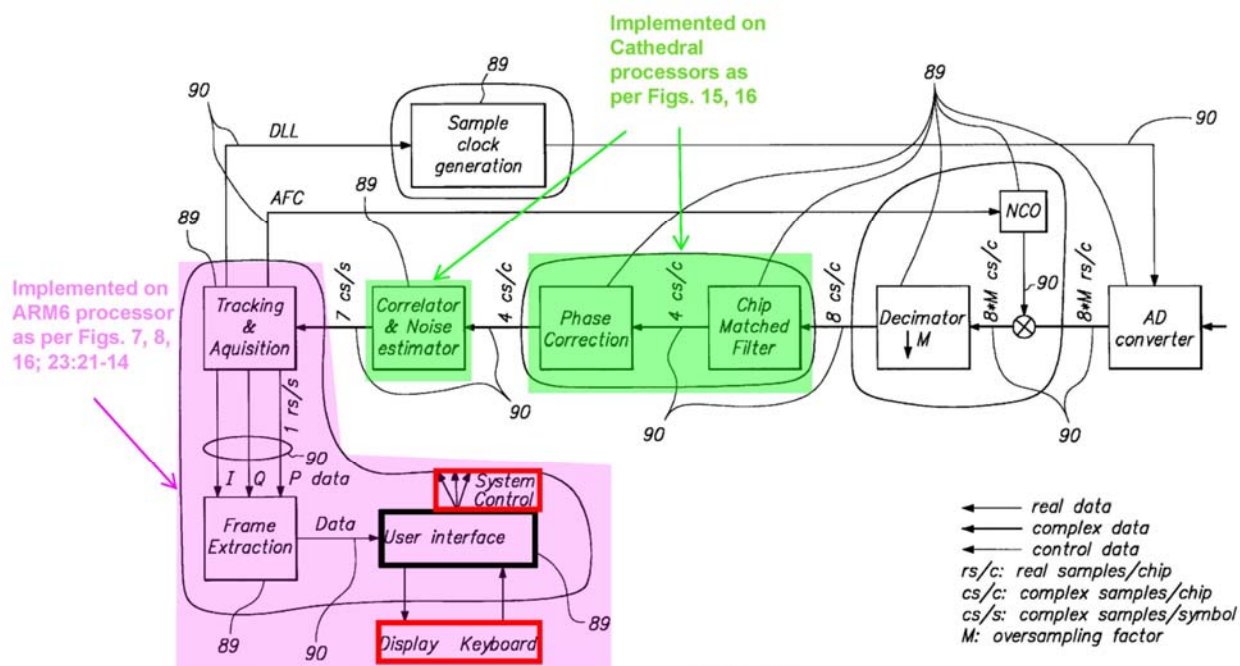
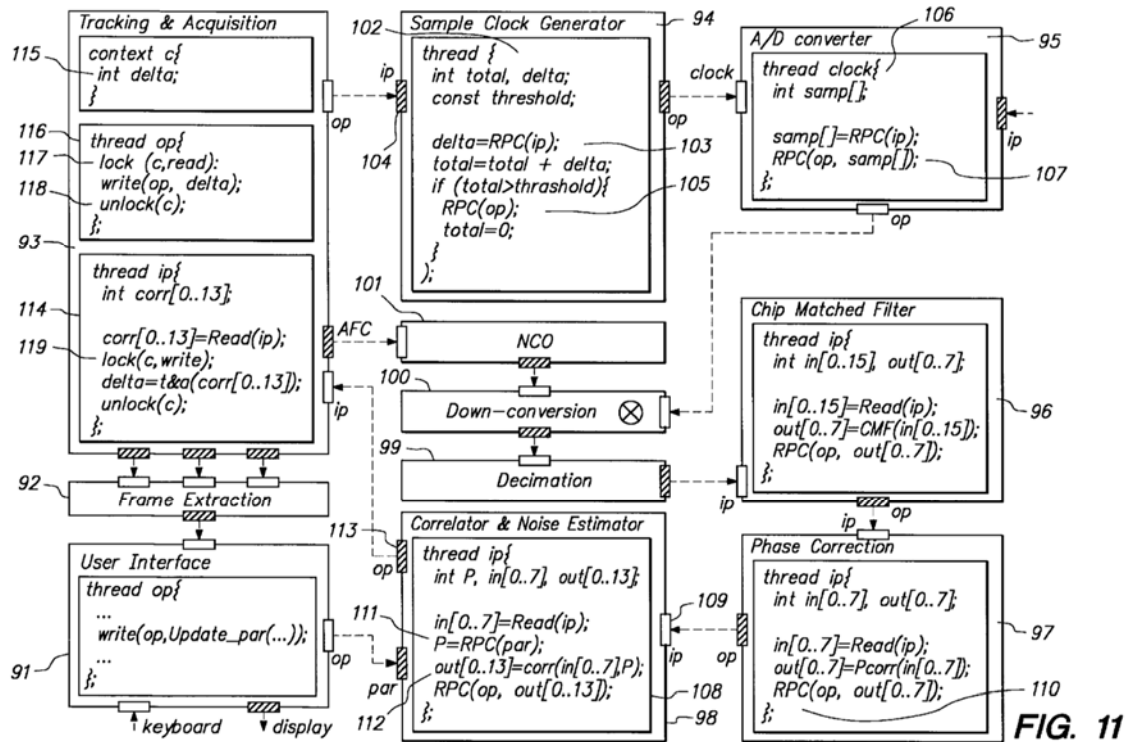


FIG. 10

EX-1004, FIG. 10. In designing the pager, the designer selects 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 inputs and outputs. EX-1003, ¶¶149-152.

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. 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 (output type) (and the requisite functionality) and a keyboard (input type). EX-1003, ¶¶153-154.



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. ... FIG. 11 shows the RPC communication in detail for part of the pager design. The blocks [91-101] in FIG. 11 correspond to the processes (89) from FIG. 10.

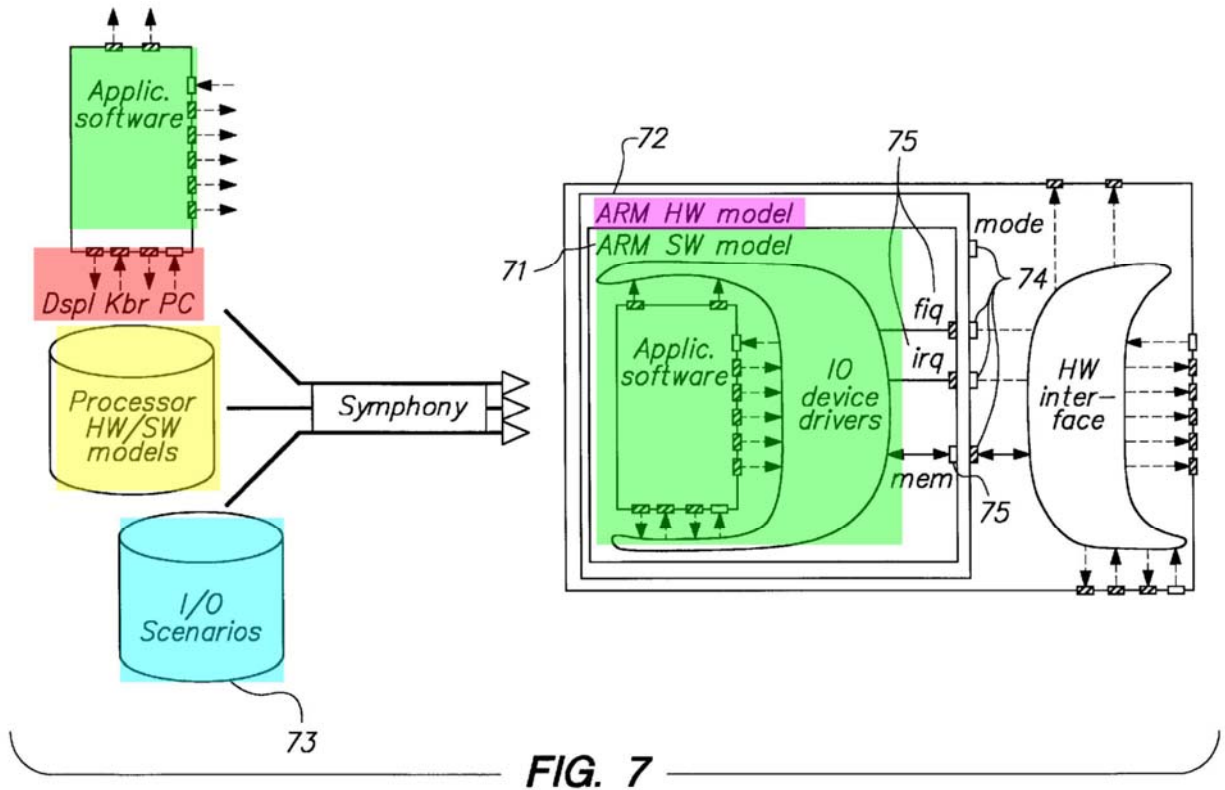
EX-1004, 21:34-41. In a CoWare design, a designer may select functionalities through the use of “processes” in the system specification. “Each process is described in the language that is best fit for the characteristics of the function it implements” such as the “C” language. EX-1004, 22:26-33. EX-1003, ¶¶157-158.

Figs. 3 and 4 depict how the designer can select the specific characteristics

associated with various processes. Fig. 3 shows an “encapsulation” that the designer builds by selecting the various subsystems to design the overall system (device) using channels, ports, and protocols. EX-1004, 12:51-14:57, 9:61-63, Fig. 3. “By using primitive channels, ports, and protocols, the designer first concentrates on the functionality of the system [device]....” EX-1004, 14:41-42.

The second type of action is CoWare “receiving” the data provided by 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”). EX-1003, ¶¶159-182. “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.

Rompaey’s Figure 7 gives a specific example of the receipt of those categorized selections of inputs and outputs:



EX-1004, Fig. 7.

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.

Rompaey’s “I/O scenario” example shows a specific user selection of input / output characteristics:

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.**

EX-1004:9-15. This is a very explicit example of how a designer selects (using CoWare) a specific I/O scenario (from the categories of different I/O scenarios identified below) that is within a hierarchical classification database (the libraries that store the I/O scenarios). EX-1003, ¶¶159-168.

Fig. 8 shows the specific hardware interfaces and associated software to control that input / output hardware selected by the designer for an exemplary ARM6 processor. EX-1004, 19:10-28.

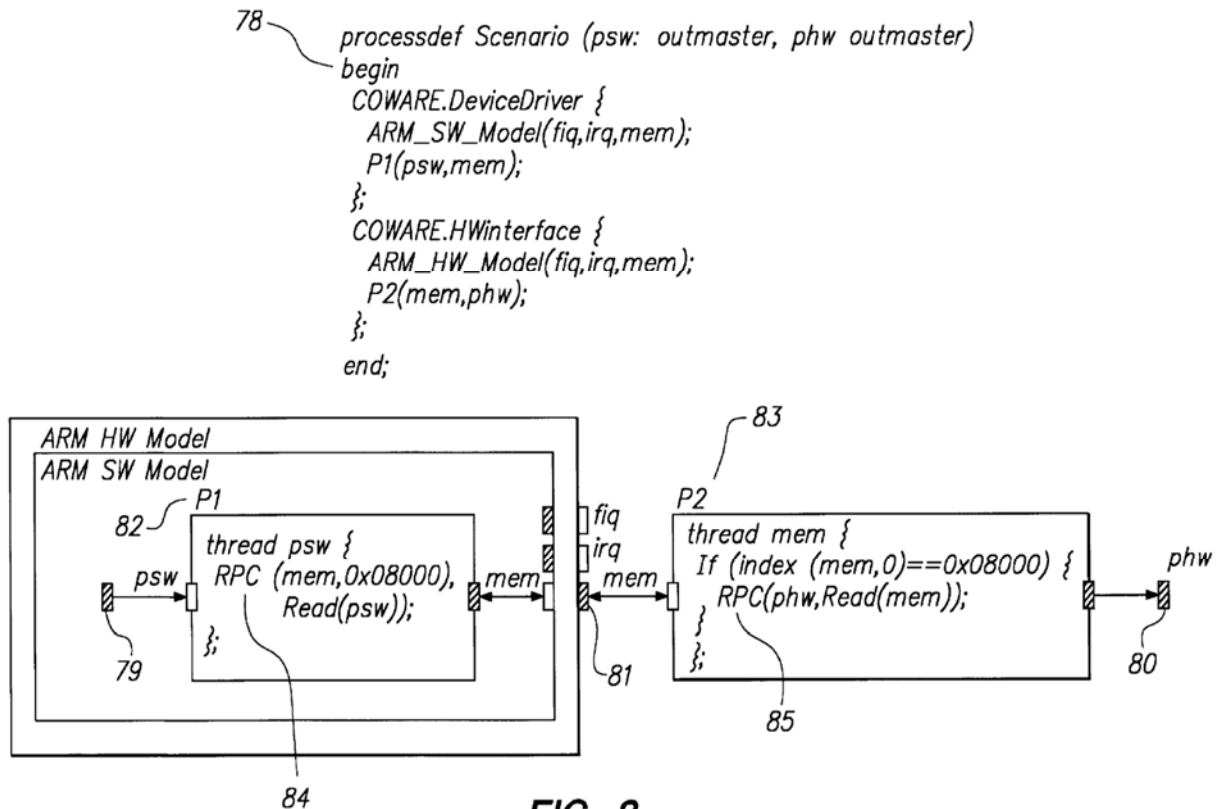


FIG. 8

EX-1004, Fig. 8. In addition, the “fiq” and “irq” are interrupts and the “mem” is a “memory-mapped I/O” – each being an input / output characteristic of the device along with specific sub-characteristics. EX-1003, ¶¶170-171.

“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 input / output characteristics – each with their own set of categorized characteristics selected after the selection of the top-level category. EX-1003, ¶¶168-169.

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. 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.

The third type of action showing the claimed “receiving” happens during the

interactive refinement process. EX-1003, ¶¶183-191. After the library and the initial design specification are received by CoWare, “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 (FIG. 5 and related discussion), assignment tools (FIG. 5 and related discussion) ... and hardware/software interface tools (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.”).

In each of the three “receiving” actions above, Rompaey’s different functions and components are categorized to effectuate Rompaey’s “reuse” goal which is a “key.” EX-1004, 4:64-5:2. EX-1003, ¶¶185-188. “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.

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

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. EX-1003, ¶¶187-188.

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. Thus, at a minimum, it would be obvious to store the re-used components in a hierarchal classified database or library for ease of use. EX-1003, ¶¶185-188.

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. UARTs implement functions for input and output types for “peripherals” such as displays and keyboards. EX-1003, ¶¶189-191 (citing U.S. Pat. No 4,430,731 (filed in 1980)).

Each form of input above is received into the claimed “a memory of a processor” 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. EX-1003, ¶¶192.

If PO argues that Rompaey does not expressly disclose any aspect of this element, Rompaey renders obvious every aspect of this element. EX-1003, ¶¶193-201. For example, if PO 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. First, PO asserts that the claimed “*selection of categorized input and output characteristics of an electronic device*” “correspond[s] to ‘the configuration and attributes of the integrated circuit being designed.’” EX-1008, 000008. Similarly, PO asserts “[e]xamples of controlled devices include computers, smartphones, wearables, mobile devices, autonomous vehicles, cameras, televisions, and IoT devices (and/or their subsystems, chips, or boards).” EX-1008, 000007. 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. EX-1003, ¶¶193-195.

Second, Rompaey discloses (and claims) specifying characteristics 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 characteristics of 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 ‘899 patent. EX-1001, 8:51-53. For example, the “I/O scenarios” implement specific components that are used as the inputs and outputs of the controlled device. EX-1003, ¶¶196-197.

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 characteristics of Rompaey’s pager “application” / device and its associated “application specific integrated circuits,” and any receiving of “device” characteristics. EX-1003, ¶198.

Similarly, if PO argues that Rompaey does not disclose “categorized” characteristics, Rompaey renders obvious such categorized characteristics. As

discussed above, Rompaey teaches the ability to specify specific different types (categories) of components such as processors, ASICs, hardware accelerators, micro controllers, communications blocks and memory, and peripherals (DMA, UART...). 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 receiving of categorized data. Finally, Rompaey’s system for inputting data uses a “hierarchical” set of different types (categories) 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 ‘899 patent used similar “hierarchal” language as Rompaey (seventeen years after Rompaey) to describe its “categorized” inputs. EX-1004, Fig. 4, 4:9-13, 6:5-7, 7:41-43, 8:31-33, 9:31-35. Thus, at a minimum, Rompaey’s “hierarchical” classification renders obvious the claimed “receiving ... categorized” characteristics. EX-1003, ¶¶199-201.

Thus, Rompaey discloses this element.

Rompaey combined with CoWare I (EX-1007) further discloses this element.

EX-1003, ¶¶202-208. CoWare I is a published paper that largely mirror's Rompaey's figures and disclosure. CoWare I includes the following Figure:

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 ‘899 patent as corresponding to input characteristics. Second, Fig. 6(b) shows that the “UART” is a component that is selected for the inputs / outputs (display and buttons). Rompaey discloses a “UART” as one of its categorized selectable functionalities. EX-1004, 4:38-49. The selection of a UART and refinement of its characteristics and functionalities would be an example of a I/O scenario (and refinement of its specific functionalities) as discussed above. EX-1003, ¶¶203-204.

A POSITA would be motivated to combine Rompaey and CoWare I with a reasonable expectation of success. The references show largely the same content, have the same pager example, are authored by the same people, were filed / published contemporaneously, and the inventors submitted CoWare I to the PTO while prosecuting Rompaey. EX-1003, ¶¶205-207.

ii. 1[b]: “analyzing by the processor the selection of categorized input and output characteristics of the electronic device”

Rompaey discloses this element. EX-1003, ¶¶210-231. Once the designer specifies the system design and libraries for implementation, the CoWare tool analyzes those selections to generate the hardware circuits and software corresponding to the selections. This is depicted in Figures 2, 6, 7, 11, 18 (showing

the receiving of several different forms of input with the claimed characteristics including input of libraries, scenarios, and models that include elements 180, 182, 186, and 190 in Figure 18).

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 and libraries to iteratively and interactively map the design onto hardware and software components. EX-1004, 12:12-43. EX-1003, ¶¶212-213.

Fig. 6 shows another example of “analyzing” in the “implementation process for hardware / software co-design according to ... FIG. 2.” EX-1004, 10:3-5.

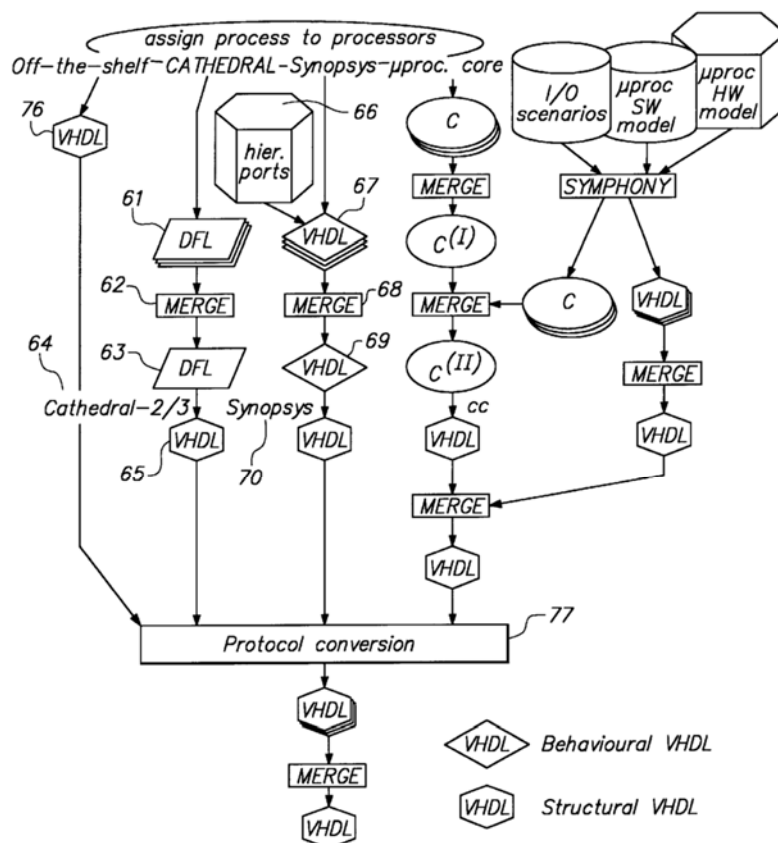


FIG. 6

EX-1004, Fig. 6. The various “merge” and “assign process to processors” blocks are all forms of the claimed “analyzing.” For example, 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 characteristics); 25:1-26 (and Fig. 15) (detailing the implementation of portions of the pager). EX-1003, ¶¶214-221.

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... 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.”). EX-1003, ¶222.

“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. Figure 17 shows two ports implemented by the SYMPHONY analysis (shown in Fig. 7). EX-1004, 25:45-26:49.

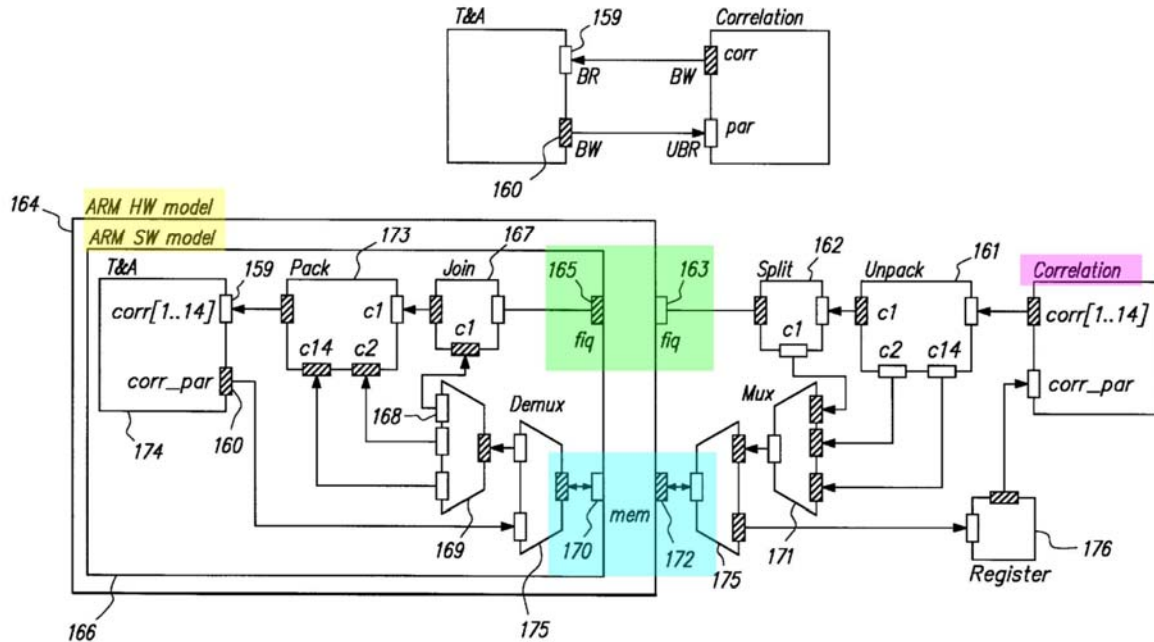
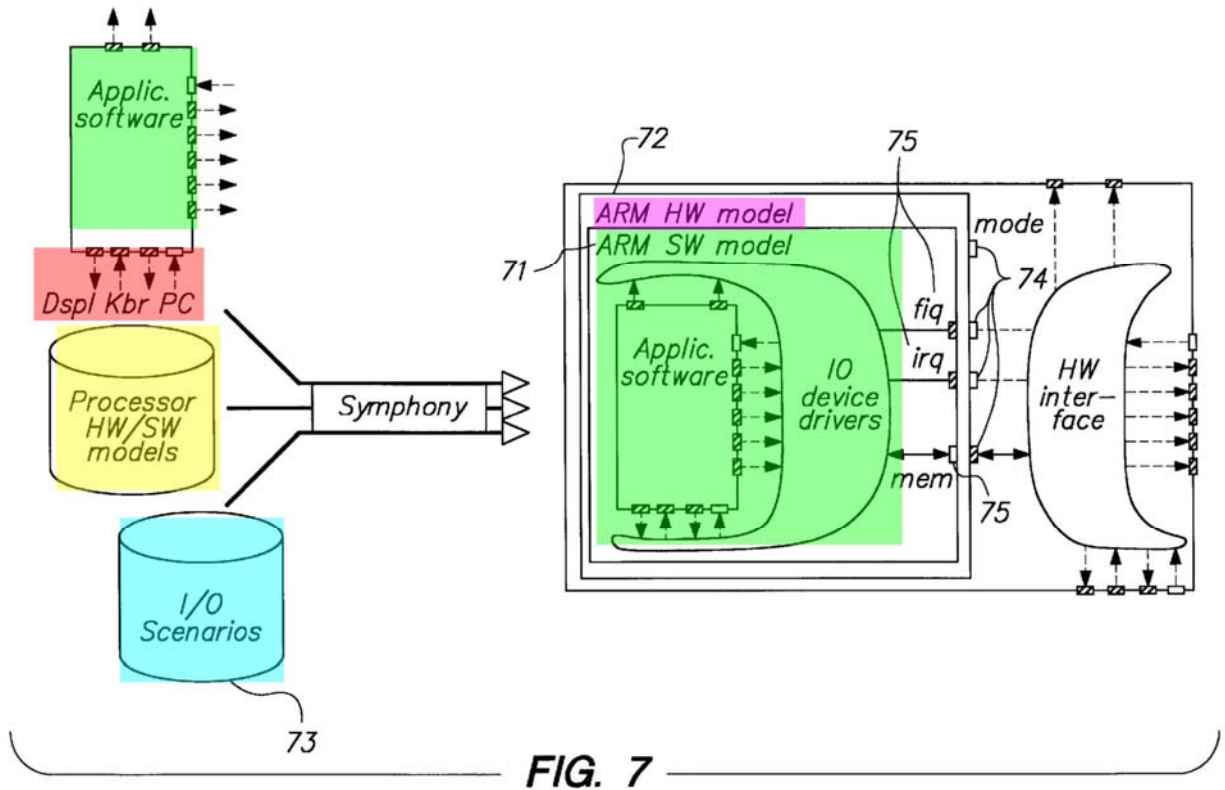


FIG. 17

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 characteristics such as protocols, data rates, and data directions. EX-1003, ¶¶223-224.

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 display, keyboard, and program controls.



EX-1004, Fig. 7.

Rompaey's claims detail this process of analyzing the selected characteristics 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 (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 (specifying characteristics, refining characteristics, generating hardware and software subsystems implementing characteristics). EX-1003, ¶¶225-226.

If PO argues that Rompaey does not expressly disclose this element, Rompaey renders obvious every aspect of this element. EX-1003, ¶¶227-230. The term “analyzing” (or any form of “analysis”) is not in the ‘899 specification and was added without explanation in response to an Office Action. EX-1002, 000044-000051. The ‘899 patent teaches only one action on such characteristics: “[e]ach characteristic ... may be linked with a predetermined standardized IC for performing and controlling those characteristics....” EX-1001, 14:64-66. As detailed above, Rompaey contains extensive disclosure regarding linkage of the input characteristics and the ability to link and use those characteristics with the libraries of existing ICs for performing the characteristics / functions. Thus, at a minimum, Rompaey teaches the only example of actions identified in the ‘899 as being performed using “characteristics” and therefore renders obvious the claimed “analyzing.” EX-1003, ¶227.

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, 000009. 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 and thus renders this element at least obvious. EX-1003, ¶¶228-230.

Thus, Rompaey discloses this element.

iii. 1[c] “outputting by the processor of a hardware integrated circuit design based on the selection of input and output characteristics of the device; and”

Rompaey outputs the generated hardware subsystem (integrated circuit design) in well-known formats including HDL, VHDL, and netlist. EX-1003, ¶¶232-242. EX-1004, 11:12-20 (“FIG. 2 shows the architecture of the CoWare system. 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.”). *See also* EX-1004, 17:46-48 (“The result is a VHDL net list (65) of the implementation...”); 20:32-38 (“VHDL net list” output); 28:22-30 (“the system designer can ... generate a compiled description of the system ...as shown in state 198. This compiled description may be ... a net list describing the final implementation of the system.”); 1:53-59; 4:19-29; 5:14-16, claims 38, 44; Figs. 1-2, 6-7, 9, and 18. The hardware

output includes “Register Transfer level” (RTL) code. EX-1004, 26:38-42; 3:35-39. EX-1003, ¶¶232-240.

If PO argues that Rompaey does not expressly disclose this element, Rompaey renders obvious this element. As detailed above, Rompaey’s 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. EX-1003, ¶241.

Thus, Rompaey discloses this element.

- 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.”*

Rompaey discloses that CoWare outputs software programming commands to be executed by the hardware circuit. EX-1003, ¶¶244-252. The discussion in element 1[b] above regarding “SYMPHONY” highlights this output of software specific to the designed software. *See also* EX-1004, 20:22-25 (“In addition SYMPHONY, adds RAM and ROM blocks to store the program code and data...”).

As noted above, 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. EX-1003, ¶¶245-249.

Rompaey describes the output of “I/O device driver” software. 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. EX-1003, ¶250.

If PO argues that Rompaey does not expressly disclose this element, Rompaey renders obvious this element. Rompaey’s 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. EX-1003, ¶251.

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 input or output characteristics.”

Rompaey discloses the use of predetermined hardware integrated circuit components throughout its disclosure of the libraries, databases, and models. EX-1003, ¶¶254-260. As detailed in 1[b] above, Rompaey emphasizes re-use with “off-the-shelf” components. EX-1003, ¶¶254-258.

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.”). The “I/O scenario models” (*see* Figs. 4, 8) are also predetermined hardware integrated circuit components controlling one of the input or output characteristics as detailed above. EX-1003, ¶¶259-260.

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

As detailed for element 1[c], CoWare generates hardware integrated circuits in formats including HDL, VHDL, netlist, and RTL which discloses this element.

EX-1003, ¶¶262-263. *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.

d. Claim 8: Preamble “A system comprising:”

Rompaey discloses this system. *See* 1[Preamble]. EX-1003, ¶¶265-284.

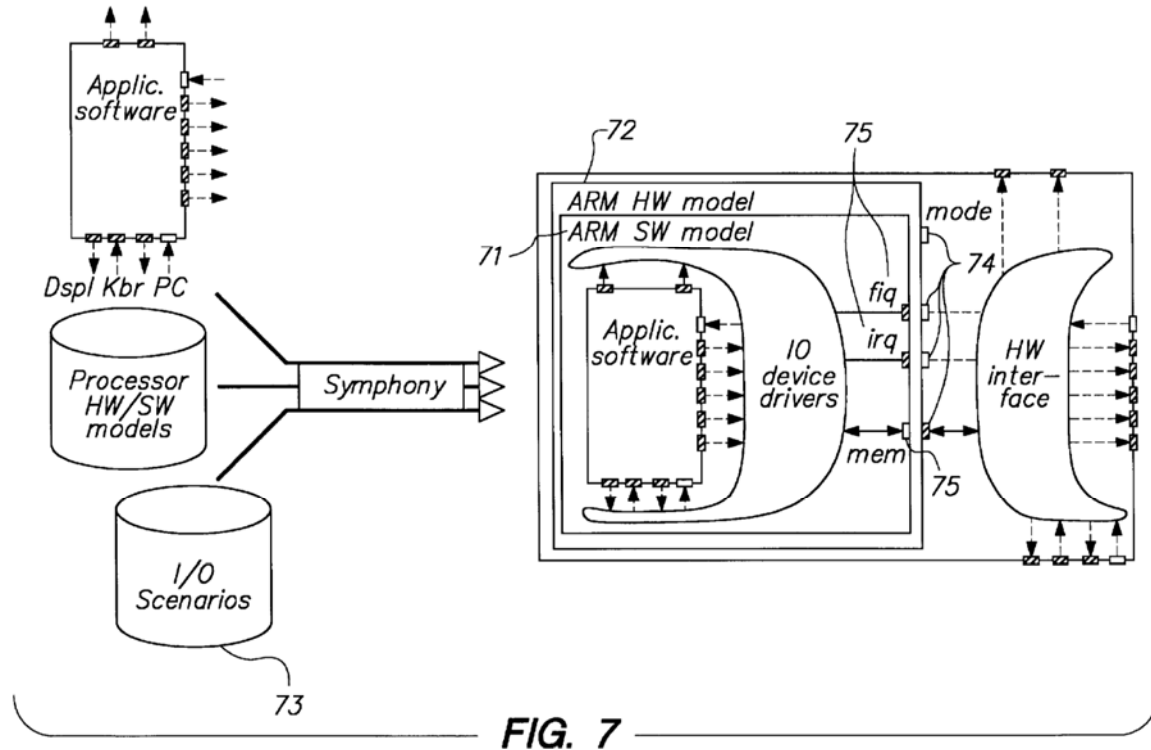
i. 8[a] “one or more processors;”

Rompaey’s CoWare design environment is implemented on a computer with executing software, thus disclosing (including rendering obvious) a processor. EX-1003, ¶¶266-269.

The CoWare model is implemented on a computer ... the tools are executed on a Hewlett Packard 715/50 workstation, a SUN, a DEC or other commercially available workstation, or a personal computer ... EX-1004, 15:50-62. *See* EX-1004, 7:32-8:3, 26:50-27:26; Fig. 18; claims 1-33 (all claiming a tool in a “computer environment”).

ii. 8[b] “logic encoded in one or more non-transitory computer-readable media that, when executed by the one or more processors, is operable to.”

As detailed above in 1[preamble], 1[a], 8[a], Rompaey’s CoWare design tool was software that executed on processors preferably in workstations. EX-1003, ¶¶270-275. Rompaey uses the well-known symbol for disk drives associated with its workstation executing the software.



EX-1004, Fig. 7. See also EX-1004, Figs. 2, 6, 18. A POSITA would have understood that CoWare software for the workstation would have been stored on non-transitory media such that the software could be used between instances of the workstation being powered off. Disk drives on the workstation provides the mechanism for storing such software. EX-1003, ¶¶271-274.

This discloses, or renders obvious, this element.

iii. 8[c]: “receive a selection of categorized input and output characteristics of a device”

As detailed in 1[a], Rompaey’s CoWare receives selections of categorized input and output characteristics as claimed in this element. EX-1003, ¶¶276-277, ¶¶99-208.

- iv. ***8[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”***

As detailed in 1[a], 1[b], and 1[c] above, CoWare designs and outputs hardware integrated circuits based on the selections of characteristics shown in element 1[a]. EX-1003, ¶¶278-285, ¶¶210-243; *see also* 1[b], 1[c], 5 above describing the design and creation of the hardware IC design.

Rompaey teaches generating software to be executed on well-known “off-the-shelf” processor models, such as “Cathedral” and “ARM6.” The designs would disclose, or render obvious, a communications bus because such buses are well-known and pervasive throughout processors and would be needed to effectively operate the software. EX-1003, ¶¶280-285.

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].” EX-1004, 19:1-29. EX-1003, ¶¶282-283.

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. EX-1003, ¶¶284-285.

- v. *8[e] “output software programming commands to executed [sic] by the hardware integrated circuit for controlling the device in accordance with the selection of categorized input and output characteristics of the device using a communication protocol compatible with the device through the communication bus of the hardware integrated circuit.”*

As detailed in 1[a], 1[b], 1[c], and 8[d] above, CoWare would create 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] to properly operate the software. EX-1003, ¶¶286-288, 244-252.

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.

Thus, Rompaey discloses this element.

- e. **Claim 14: “The system of claim 8, 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"

As detailed in 1[b] above, Rompaey discloses simulating the hardware design. EX-1003, ¶¶289-294. This is shown throughout Rompaey. *E.g.*, EX-1004, 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...”). Figure 9 depicts simulation at phase (4):

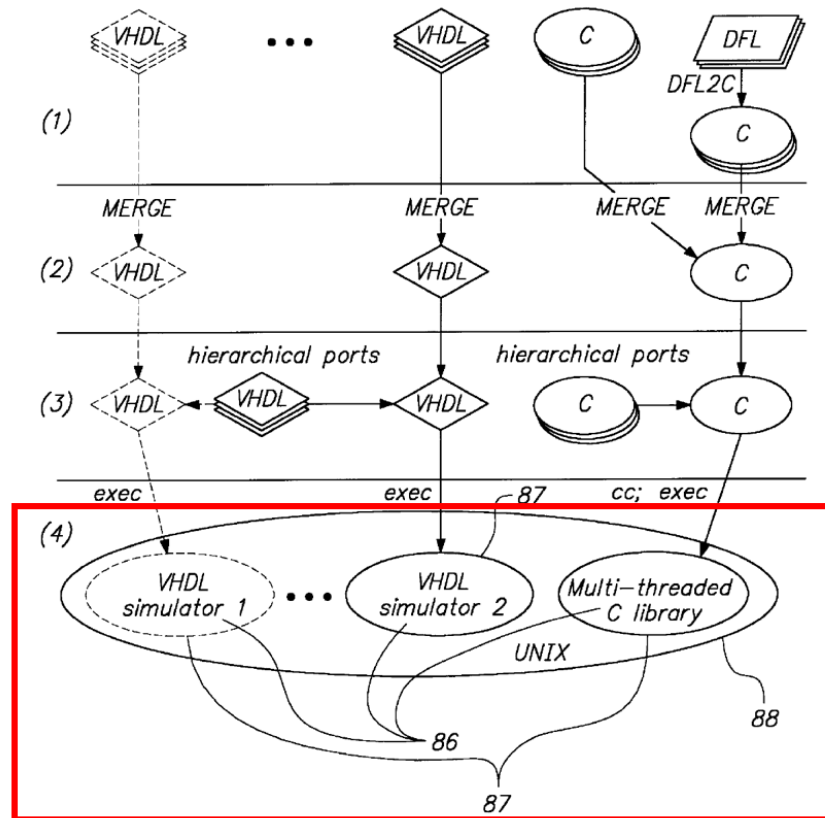


FIG. 9

EX-1004, Fig. 9; 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.

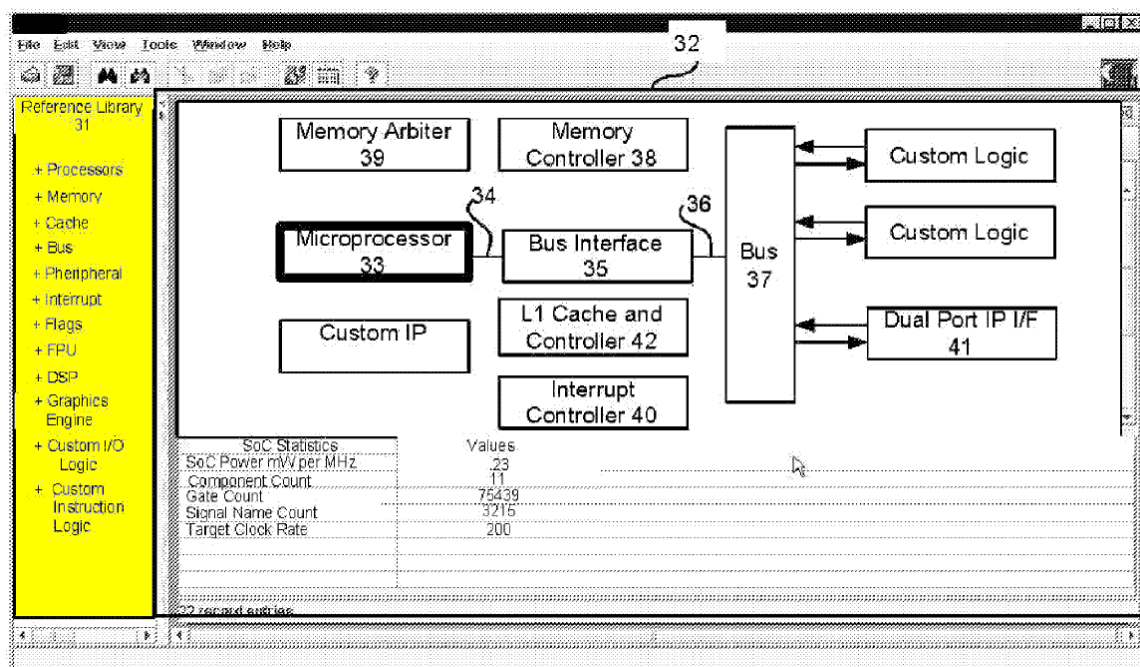
2.”); 6:36-34; 7:32-8:3; 9:14-15; 11:12-20; 12:6-11; 18:63-67; 20:39-21:28; 22:38-40; 23:51-56; 22:65-67; 28:13-30; claims 2-5, 27, 35.

D. Ground 2: The Challenged Claims Are Obvious Over Banerjee.

1. Overview of Banerjee

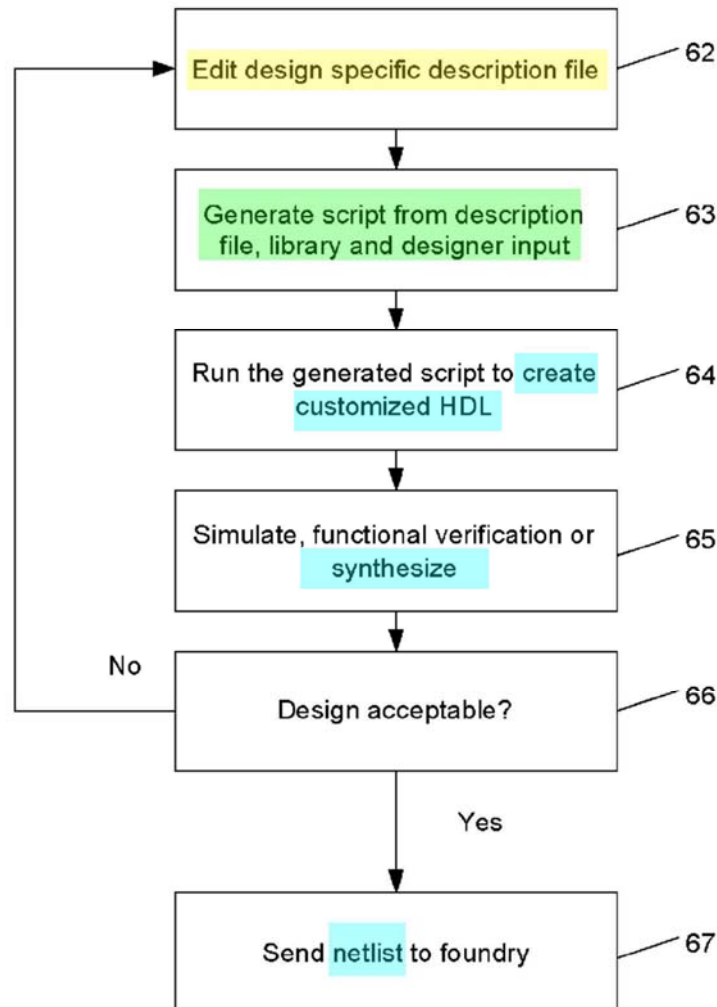
Banerjee discloses, or renders obvious, every element of the Challenged Claims. EX-1003, ¶¶296-406. Banerjee teaches a software tool for designing systems / devices with hardware integrated circuits and software that executes on the hardware. The designer enters “parameters,” “characteristics,” and “attributes” to tailor the designed hardware and software for a specific “application” (functions of the device controlled by the hardware circuit and software), for example, devices that require “real-time processing of video or audio” or “media players” or “signal processing,” “embedded applications,” or “custom I/O logic.”

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



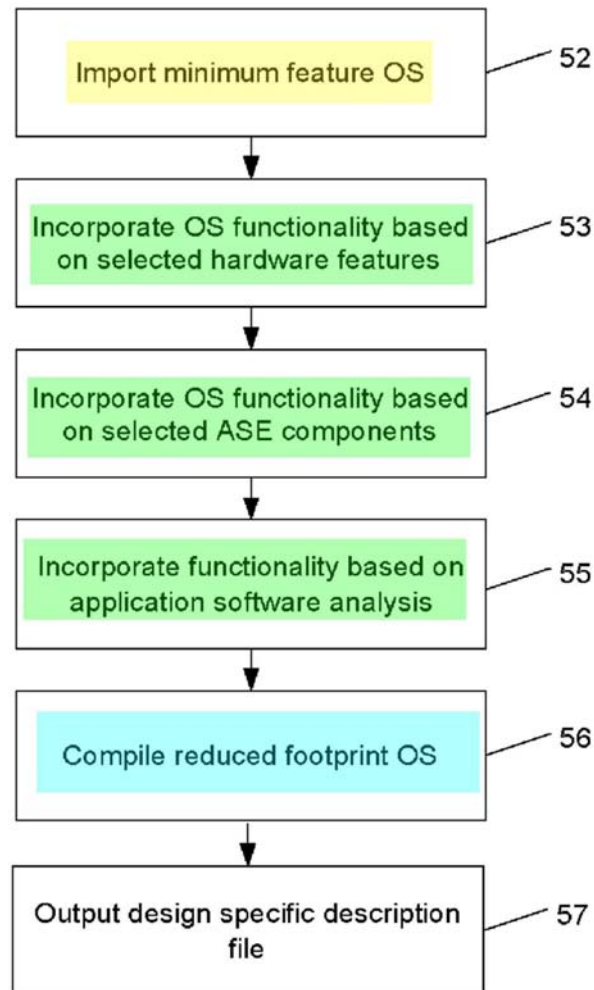
EX-1005, Fig. 3.

Banerjee's 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**” (hardware integrated circuit design).



EX-1005, Fig. 6.

Fig. 5 exemplifies how one software portion (“OS”) is designed and output. Banerjee’s tool receives 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 Banerjee’s “application software” and “middleware.” EX-1005, ¶¶0072, 0078.



EX-1005, Fig. 5.

Thus, as detailed below, Banerjee discloses all features of the Challenged Claims and renders each invalid.

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

a. Claim 1: Preamble:

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 controlling specific applications (devices) and then output the integrated circuit design and the associated software. EX-1005, ¶¶0027-0028; claim 17 (“... selecting components... generating ... HDL ... file describing the integrated circuit.”); EX-1003, ¶¶95-98.

i. Element 1[a]

Banerjee’s software tool “receiv[es] a selection of categorized input and output characteristics,” as claimed, in multiple ways. EX-1003, ¶¶300-345. Notably, Banerjee’s description of “inputs and outputs” mirrors that of the (later) ‘899 patent.

<u>‘899 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	“the designer compil[e]s a set of system specifications based on the desired functionality of the SoC....

using IC components that are developed for generic <u>characteristics of electronic devices</u>. <u>A developer can select the functionality of their device</u> and an IC can be designed off the standard components.” EX-1001, 4:35-45.	These specifications can be used to define a SoC in terms of the <u>desired inputs and outputs</u>, as well as <u>desired functionality</u> ...” EX-1005, 0004.
---	---

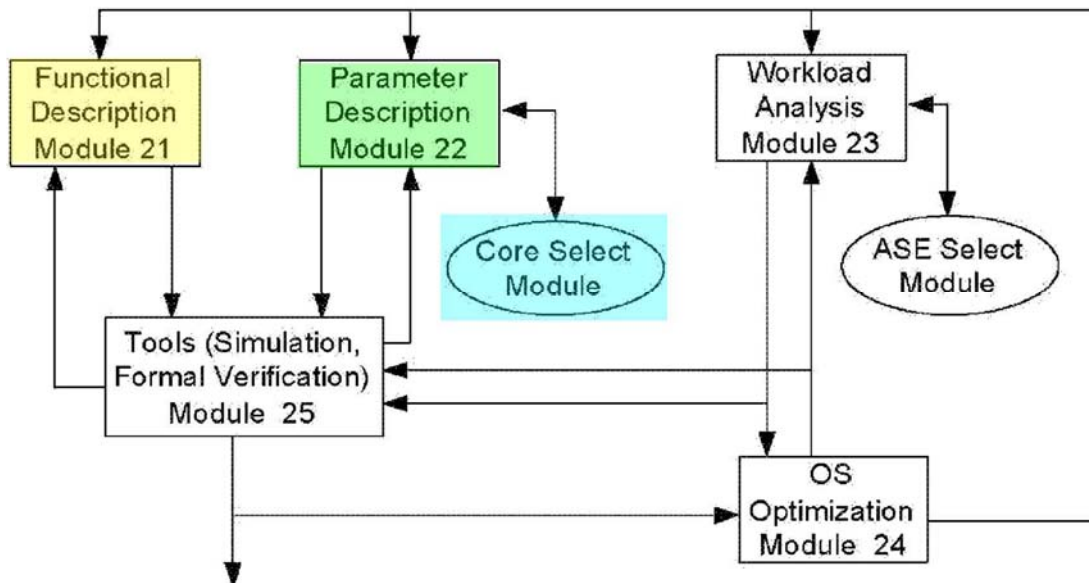
Banerjee teaches many ways for a designer to select the “desired inputs and outputs” for the device. First, the tool receives from a designer the system design, the related components implementing the design, and the “parameters” and “characteristics” thereof. These inputs are received via multiple different user interfaces including a GUI, “design scripts,” or “in the form of an interactive software prompt, or ... [other identified inputs].” EX-1005, ¶¶0033-0034, 0047-0048, 0058-0061, 0088; Figs 2, 3. EX-1003, ¶¶305-315.

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. 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.”). EX-1003, ¶¶316-317.

Third, Banerjee’s tool receives selected system design, parameter, and characteristics data from “libraries” and “repositories.” EX-1005, Fig. 1 (“Repositories”); Fig. 3 (“Reference library”), Fig. 4 (“Processor library”), Fig. 6, Fig. 11 (“library files”). EX-1003, ¶318.

Banerjee’s Fig. 2 depicts an iterative, interactive design process:



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. Computer program

15 also includes the capability to upload design scripts that describe customized elements created by the designer.

EX-1005, 0033. EX-1003, ¶¶319-320.

Second, Banerjee’s tool then receives further data about “parameters” and “characteristics” of the system design and its components (one example of claimed “characteristics”):

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 ... 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.

The “**core select module**” works with “**parameter description module 22**” to allow designers to select from categories of processors and select their parameters, features, and attributes tailored to the specific desired application / device. EX-1005, ¶¶0064-0068, Figs. 4, 8. EX-1003, ¶¶321-322.

In Banerjee, a designer tailors components and characteristics thereof for specific “applications” (*e.g.*, the devices controlled by the designed hardware / software). For example, Banerjee teaches selecting specific parameters (characteristics) 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 are essentially logical control oriented” (EX-1005, ¶0086).

For each of Banerjee’s categories of components, a designer selects

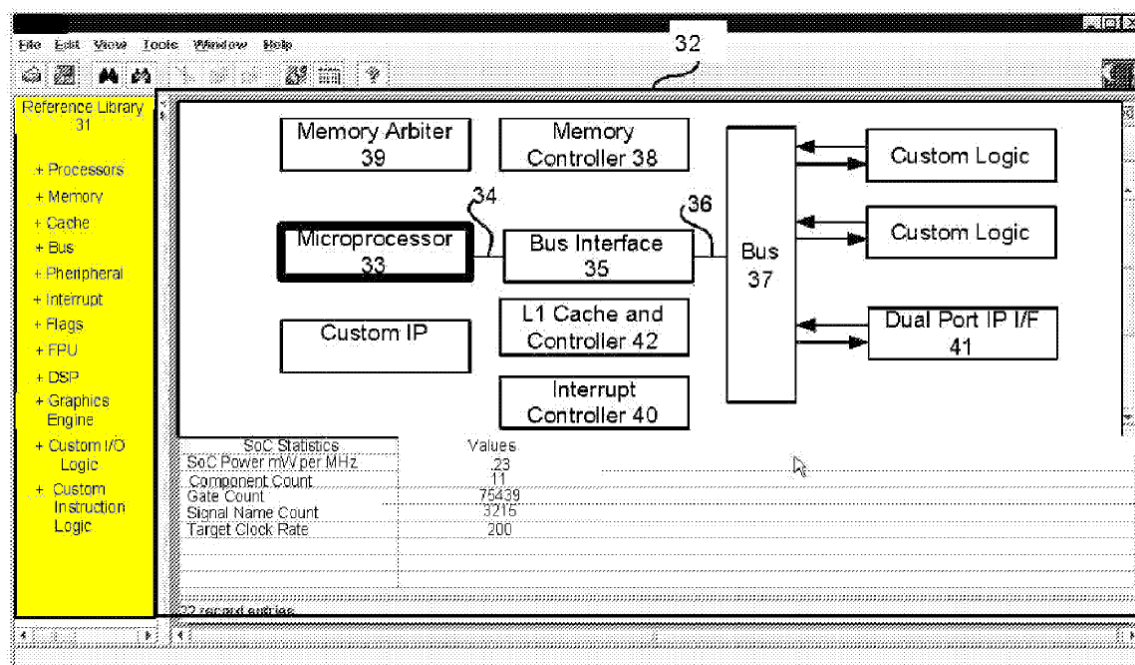
application specific “parameters,” “attributes” and “characteristics” – which all disclose the claimed “characteristics.” EX-1005, ¶¶0067-0071, 0074-0092. Banerjee describes a “cell phone” device that might have specific requirements. EX-1005, ¶0077. Thus, Banerjee’s system allows the system designer to select the components and characteristics of those components that are specific to the devices for which the hardware and software is being developed. EX-1003, ¶¶323-324.

The selections (from the designer and the repositories / libraries) allow Banerjee’s system to select the functions / characteristics 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. EX-1003, ¶325.

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). EX-1003, ¶326.

One example of an invalidating selection would be to use the Fig. 3 GUI to

specify an “application” device that includes multimedia and graphics by selecting those functionalities for the device.



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.

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, ¶0048. EX-1003, ¶¶327-329. These categories are “hierarchical” such that, when “selected,” each category displays further selectable options:

[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...

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

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.

EX-1005, ¶0051. The designer may specify additional parameters and characteristics 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. “[A] second menu, if necessary, enables selection” of additional characteristics and “attributes.” EX-1005, ¶0056. EX-1003, ¶¶330-335.

Examples of the claimed selection of input / outputs characteristics shown on Fig. 3 (and its description) include at least the “peripheral” (sic), “graphics engine,” “video processor,” “DSP,” “custom I/O logic,” “Custom Instruction Logic,”

“Interrupt,” and “Memory.” EX-1003, ¶336.

The “custom I/O logic” allows for selection of categories of such input / outputs specific for the designed application (device). EX-1005, ¶0048 (selection of “I/O interfaces and custom logic”); ¶0034 (“parameter description module ... provides a GUI for selecting ... I/O requirements”). Such I/O requirements disclose (or render obvious) the selection of input and output characteristics. EX-1003, ¶¶337.

The “graphics engine” (or “graphics processor”), “video processor” and “digital signal processor (‘DSP’)” are categories and types of input / output characteristic of the overall device (*e.g.*, a device that outputs/displays graphics and/or video) that the designer may select. EX-1005, ¶0048. This matches the ‘899 description that “input and output of the device may be audio, video....” (EX-1001, 14:46-57) and a “display that amounts to a word or graphic.” EX-1001, 9:5. EX-1003, ¶338.

The designer may then “select” a specific type of graphics or video processor and attributes thereof. EX-1005, ¶0066. Within that graphics engine characteristic, the designer may select additional characteristics. *E.g.*, EX-1005, ¶0068 (“module 23 may include one or more IP blocks of the processor core architecture to 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. EX-1003, ¶¶339-340.

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.* EX-1003, ¶341.

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. EX-1003, ¶342.

Banerjee’s selection of interrupts and “peripherals” also disclose (or render obvious) forms of “input / output characteristics” to be selected. EX-1005, Fig. 3, ¶¶0053, 0076, 0097. As detailed for Rompaey, a POSITA understood “peripherals” to include devices such as keyboards, buttons, and displays. EX-1003, ¶343.

Banerjee specifically discloses a “cell phone” as a device. EX-1005, 0077. A POSITA would understand “cell phone” to include devices with functionality such as touch screens or buttons – which exactly matches the ‘899 statement that “the input device may be push buttons or touch sensitive elements....” EX-1001, 9:7-8.

EX-1003, ¶¶344-345.

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. EX-1003, ¶¶346-352.

According to PO, the “memory” represents an “input.”³ 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.

Banerjee’s “ASICs” are tailored toward the “application” (the device) that the IC device will control. 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, or define, the device. Banerjee gives an example of tailoring the IC for a cell phone. Thus, at a minimum, Banerjee renders obvious receiving characteristics for the device. EX-1003, ¶¶348-349.

Moreover, PO asserts that the claimed “*selection of categorized input and output characteristics of an electronic device*” “correspond[s] to ‘the configuration and attributes of the integrated circuit being designed.’” EX-1008, 000008, 000010-000012. Similarly, PO asserts “[e]xamples of controlled devices include computers,

³ EX-1008, 000025 (“an input type parameter (e.g., memory and other system inputs)”).

smartphones, wearables, mobile devices, autonomous vehicles, cameras, televisions, and IoT devices (and/or their subsystems, chips, or boards).” EX-1008, 000007. 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.

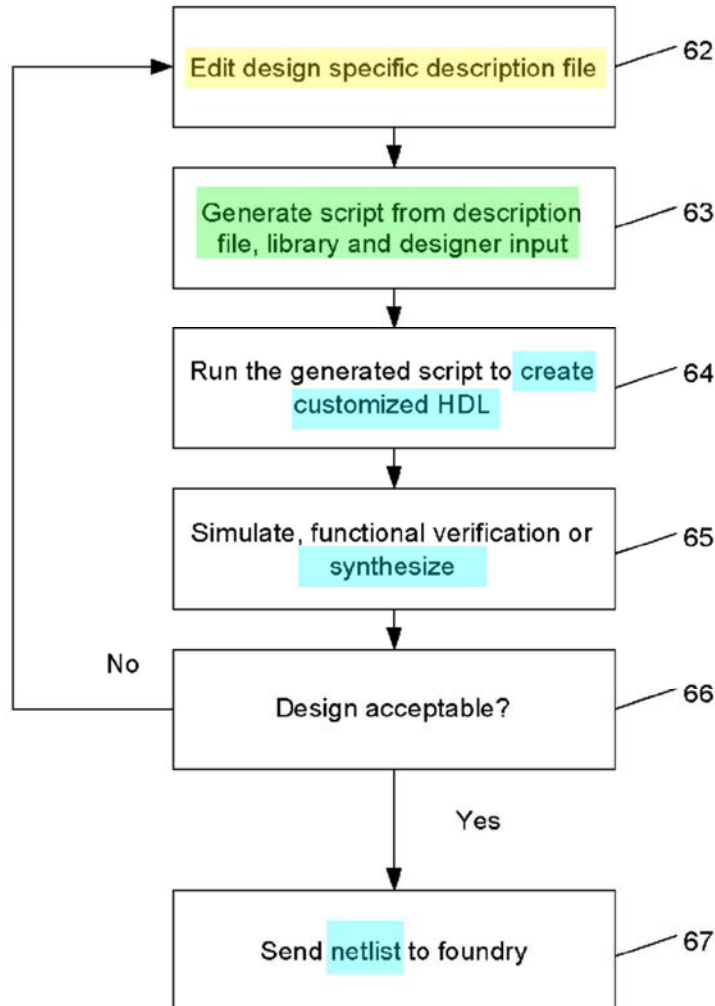
PO also asserts that this element is met by “a predetermined integrated circuit component in a cellular modem SoC in a smartphone that controls connectivity (input or output characteristics)....” EX-1008, 000012. This matches Banerjee’s disclosure of designing an ASIC SoC for Banerjee’s “cell phone.” EX-1005, ¶0077. Banerjee’s discussion of “Custom I/O Logic” (above) also would be recognized by a POSITA to correspond to such a “cellular modem SoC” and therefore, at a minimum, render this element obvious. EX-1003, ¶350.

Regarding “categorized ... characteristics,” Banerjee discloses a GUI of components and subcomponents and their “parameters” and “characteristics” and “attributes” –which can be progressively refined with further selections using the sub-menus. At a very minimum, this renders obvious the receipt of selected categorized characteristics. EX-1003, ¶351.

Thus, Banerjee discloses this element.

ii. Element 1[b]

Banerjee's tool performs different forms of the claimed "analyzing." EX-1003, ¶¶354-368. 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** with the system ultimately "**create the customized HDL**" in the form of a "**netlist**" that could be sent to a foundry:



EX-1005, Fig. 6. 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.”). EX-1003, ¶¶356-358.

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.

Such analysis includes Banerjee’s processes of “merging,” “compiling,” and “optimizing.” “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. *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. EX-1003, ¶¶359-361.

As detailed in 1[a], Banerjee’s “modules 21, 22, and 23” allow the designer to generate 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; *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. EX-1003, ¶¶362-363.

Furthermore, after the designer specifies the design and intended applications (e.g., device functions 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.

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”).

Thus, Banerjee teaches many forms of analyzing the received selections to generate and output the hardware circuit design and associated software. EX-1003, ¶¶364-366.

If PO argues that Banerjee does not disclose this element, Banerjee renders obvious every aspect of this element. EX-1003, ¶¶367-368. As noted in Section VII.C.2.a.ii, the ‘899 patent has sparse disclosure of “analyzing” and, at most,

discloses associating characteristics with predefined ICs. As detailed above, Banerjee discloses the same association between the functional design (including characteristics) and the implementations supplied by libraries and other inputs for those designs. Thus, at a minimum, Banerjee renders this “analyzing” obvious.

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, 000009. 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”).

iii. Element 1[c]

As detailed in 1[b], Banerjee’s tool outputs the hardware integrated circuit designs based on the selections detailed in 1[a] in formats including Verilog, VHDL, System Verilog, HDL, netlist, GDSII (GDS2), and RTL. EX-1003, ¶¶370-374; *see* 1[b]; EX-1005, ¶0094 (“The output of computer program 15 is an HDL synthesizable model of the design.”); Figs. 5-7, 10-12; ¶¶0028, 0040, 0043-0045, 0050, 0066, 0071, 0089-0093, 0095 (“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"); 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.

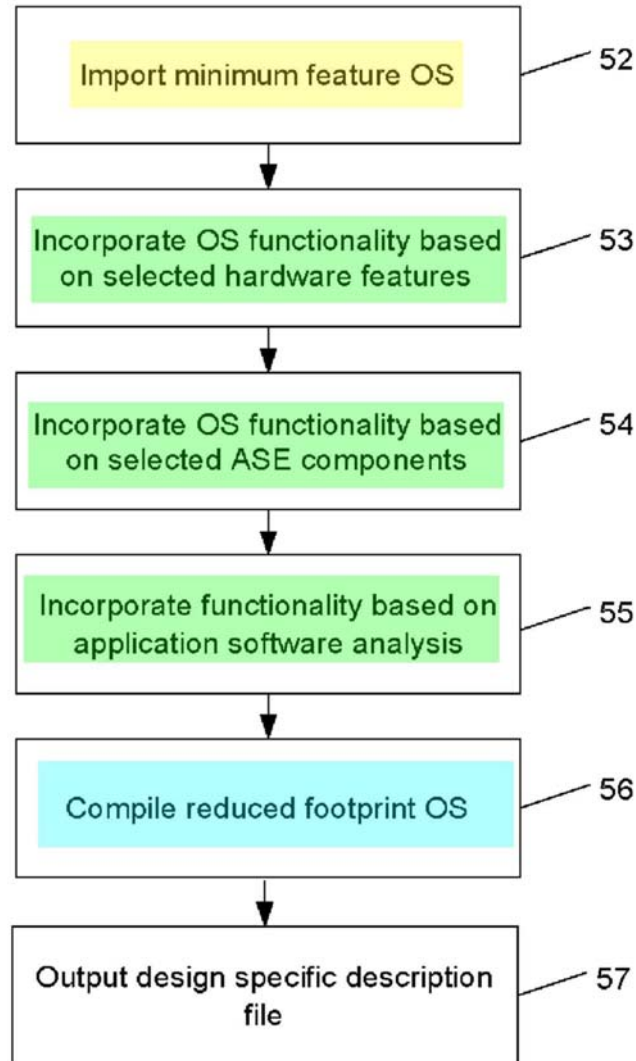
iv. Element 1[d]

Banerjee's tool outputs the software specific to the designed system and application thereof and based on the selections identified in 1[a]. EX-1003, ¶¶375-380. 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.

Fig. 5 exemplifies how the operating system software is compiled and output:



EX-1005, Fig. 5; ¶¶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. EX-1003, ¶¶376-379.

Similarly, Banerjee’s tool generates and outputs software instructions related

to the application specific instructions. EX-1005, ¶¶0086, 0038. EX-1003, ¶380.

b. Claim 5

Banerjee uses predetermined hardware integrated circuit designs that have been stored in, for example, the repositories and libraries. EX-1003, ¶¶382-383. *See* 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

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. *See also* EX-1005, ¶0091 (“a synthesis engine may create a specific physical layout of the design”). These disclose, or render obvious, this claim. EX-1003, ¶¶384-387.

d. **Claim 8: Preamble**

Banerjee teaches this “system.” See claim 1 above.

i. *Element 8[a]*

Banerjee discloses this claim. Banerjee’s “**computer program**” executes on a “**server**” computer which discloses, or renders obvious, the claimed processor[s] as part of a “system” because a processor executes software. EX-1003, ¶¶390-394.

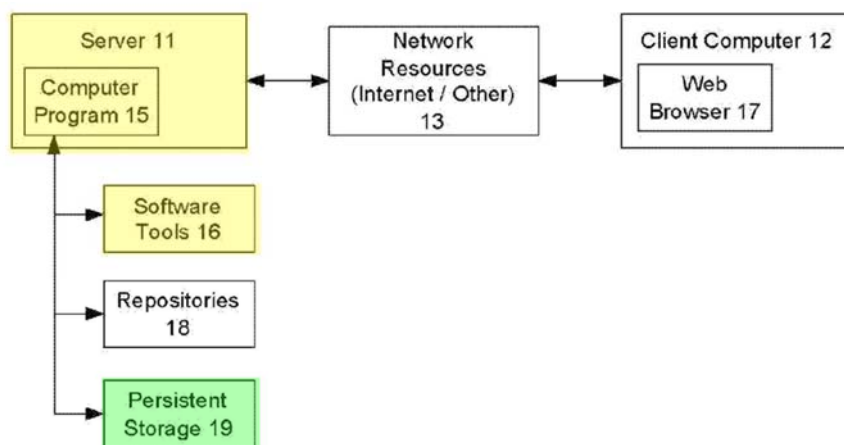


Figure 1

EX-1005, Fig. 1; see also EX-1005, Fig. 13; ¶¶0027-0028 (“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** ... server 11 may be one server in a data center environment 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.

ii. Element 8[b]

As noted in 8[a], Banerjee’s “[c]omputer program 15, software tools 16 and repositories 18 are retained on a persistent storage device 19 that is capable of storing computer readable information.” which is part of the server and is non-transitory computer readable media. EX-1005, ¶0028, Figs. 1, 13. EX-1003, ¶¶395-396.

iii. Element 8[c]

Banerjee’s program receives a selection of categorized input and output characteristics as detailed in 1[a]. EX-1003, ¶¶397, ¶¶300-352.

iv. Element 8[d]

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 8[c] / 1[a]. That hardware design is output in many different formats. *See* 1[d], 5. EX-1003, ¶¶398, ¶¶354-374.

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. EX-1003, ¶¶399-400.

v. Element 8[e]

As detailed in 1[d], Banerjee discloses outputting software as claimed. The software is purposefully designed to use communications protocols compatible 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 (with a compatible protocol) to access data in memory and control the device. EX-1003, ¶¶402-404; EX-1005, ¶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.”)

Moreover, Banerjee teaches that compatibility (such as communication

protocol compatibility for the bus(es)) among the components is tested and resolved during the development phase. EX-1005, ¶¶0041, 0125.

e. **Claim 14.**

Banerjee's tool simulates the hardware integrated circuit functionality. EX-1003, ¶¶405-406. *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.

VIII. THE BOARD SHOULD NOT DENY INSTITUTION UNDER SECTION 314

A. The Board Should Not Deny Institution Under 35 U.S.C. § 325(d)

The '899 patent has not been previously challenged in an IPR. The Examiner did not consider the references herein. These references present compelling merits. PO asserts that the claim scope encompasses design tools going well-known since the 1980's.

B. The Board Should Not Deny Institution Under 35 U.S.C. § 314(a)

Under the Director's June 2022 interim guidance,⁴ this Petition is particularly strong in the underlying merits as it demonstrates that the exact functionality and structure recited in the Challenged Claims existed in the prior art. Moreover, the asserted prior art shows exactly the functionality that the PO asserted was patentable to the Examiner. Thus, under the Interim Guidance, the Board should not deny institution.

⁴ Interim Procedure For Discretionary Denials In AIA Post Grant Proceedings With Parallel District Court Litigation, PTO Director's Memorandum (June 21, 2022).

Nevertheless, the Board's decision in *Sotera Wireless, Inc. v. Masimo Corp.*, IPR2020-01019, Paper 12 (Dec. 1, 2020) instructs that a holistic view of the *Fintiv* factors also weighs in favor of institution.

Factor 1 is neutral because no request for stay has been filed.

Under factor 2, the scheduled Litigation trial date is March 3, 2025, but that is not a date certain. For example, the Judge assigned to the Litigation has two patent trials scheduled in February 2024. The first case (*Sable Networks*) was filed in March 2021 and its initial trial date (3/6/2023) was slipped to 6/12/2023, 11/6/2023, and finally 2/5/2024. EX-1016, 000015, 000019, 000025, 000041. The second case (*Demaray*) was filed in March 2020 and its initial trial date (12/27/2021) was slipped to 4/4/2022, 5/8/2023, 9/11/2023, 1/22/2024, and 2/12/2024. EX-1017, 000036, 000043, 000047, 000058, 000059. A Final Written Decision is expected in the present matter in August of 2025 – which would be before any trial if trial schedules similar to the currently scheduled trials (*Sable Networks* and *Demaray*) occur. Thus, this factor favors institution.

Factor 3 favors institution. The schedule (EX-1015) required binding Infringement Contentions on September 12, 2023. Thus, this Petition is filed within five months of Petitioners knowing the asserted claims. Merits discovery has not opened, and the *Markman* hearing is scheduled for February 27, 2024. Fact and expert discovery close September 12, 2024, and November 7, 2024, respectively.

EX-1015. Thus, the Litigation is in its early stages, and the investment has been minimal.

Under factor 4, there will not be complete overlap in the issues raised in the IPR and the Related Litigation. Petitioners challenge all claims asserted in the Litigation, but PO has asserted 13 claims from two patents in the Litigation – an amount that likely exceeds the number of claims allowed at trial. Given that the patentee will likely be required to reduce the asserted claims, it is highly unlikely the district court will address the validity of all Challenged Claims. Petitioner stipulates that it will not pursue invalidity against the asserted claims in the district court using the combination of prior art references in the grounds in this Petition for establishing obviousness. *Sand Revolution II LLC v. Continental Intermodal Group-Trucking LLC*, IPR2019-01393, Paper 24 at 11-12 (PTAB June 16, 2020) (holding a similar stipulation weighs against discretionary denial)

Under factor 5, Petitioner is a defendant in the Related Litigation. Therefore, this factor favors institution.

Under factor 6, other circumstances heavily weigh in favor of institution. Here, the merits of the Petition are particularly strong. For example, each asserted reference directly and unambiguously teaches an identical architecture and the identical concepts disclosed and claimed in the ‘899 patent, demonstrating that the

Petition is particularly strong in the underlying merits. Each asserted reference also directly discloses exactly what PO asserted was patentable during prosecution.

When viewed holistically, the timing of the present Petition is reasonable, there has been relatively limited investment in the Litigation, and there is minimal overlap between the present IPR and the Litigation. Coupling these *Fintiv* considerations with the compelling evidence of unpatentability presented in the Petition, the efficiency and integrity of the IPR process is best served by instituting review.

IX. MANDATORY NOTICES UNDER 37 C.F.R. §42.8

A. Real Party-in-Interest (37 C.F.R. § 42.8(b)(1))

The real party-in-interest in this Petition is Arm Limited.

No entity other than Petitioner is funding, controlling, or otherwise has an opportunity to control or direct this Petition or Petitioner's participation in any resulting IPR. No entity other than Petitioner has reviewed this Petition. None of the entities identified above as potential parties-in-interest have agreed to be named as real parties-in-interest. In any event, because this Petition is being filed within one year of the filing of the Complaint in the Litigation (EX-1014), and this is the only petition that has ever been filed against the '899 patent, there is no time bar, estoppel or anything else material to the case relevant to the naming of real parties-in-interest, and thus the party-in-interest issue need not be addressed by the Board

Petition IPR2024-00476 for *Inter Partes* Review of U. S. Patent No. 8,924,899

as part of its institution analysis. *See Sharkninja Operating LLC, et al. v. iRobot Corp.*, IPR2020-00734, Paper No. 11 (PTAB Oct. 06, 2020).

Petitioner separately notes that Petitioner has other corporate subsidiary entities, including Arm Inc., each of which agrees to be estopped under the provisions of 35 U.S.C. § 315 to the same extent that Petitioner is estopped.

B. Related Matters (37 C.F.R. § 42.8(b)(2))

1. Judicial Matters

As of the filing date of this Petition and to the best knowledge of Petitioner, the ‘899 patent is involved in the following litigations:

- *ICPillar LLC v. ARM Ltd, et. al.*, Case No. 1:23-cv-00282 (WDTX) (the “Litigation”).

2. Administrative Matters:

As of the filing date of this Petition and to the best knowledge of Petitioner, the ‘899 patent has not been subject to any petitions for *Inter Partes* Review.

3. Related Patents

To the best of Petitioner’s knowledge, the ‘899 patent is related to the following United States Patent Nos.: 9,367,657 and 10,216,174.

C. Lead/Back-up Counsel (37 C.F.R. § 42.8(b)(3))

Lead Counsel:

Kevin P. Anderson, Reg. No. 43,471

kpanderson@duanemorris.com

DUANE MORRIS LLP

901 New York Ave NW 700 E

Washington, D.C. 20001

T: (202) 776-5213

F: (202) 776-7801

Back-Up Counsel:

Patrick D. McPherson, Reg. No. 46,255

PDMcPherson@duanemorris.com

DUANE MORRIS LLP,

901 New York Ave NW 700 E

Washington, D.C. 20001

P: (202) 776-5214

F: (202) 776-7801

D. Notice of Service Information (37 C.F.R. § 42.8(b)(4))

Please direct all correspondence to lead and back-up counsel at the above addresses. Petitioner consents to electronic service at the email addresses above.

E. Fees Under 37 C.F.R. § 42.103

Petitioner submits fees of \$41,500. If more fees are necessary to accord this Petition a filing date, authorization is granted to charge the same to Deposit Account No. 04-1679.

X. CONCLUSION

Petitioner respectfully requests the Board institute IPR and cancel all Challenged Claims as unpatentable.

Respectfully submitted,

DUANE MORRIS LLP

BY: /Kevin P. Anderson/
Kevin P. Anderson, Reg. No. 43,471
kpanderson@duanemorris.com
DUANE MORRIS LLP
901 New York Ave NW 700 E
Washington, D.C. 20001
T: (202) 776-5213
F: (202) 776-7801

Dated: January 26, 2024

CERTIFICATION OF SERVICE ON PATENT OWNER

Pursuant to 37 C.F.R. §§ 42.6(e), 42.8(b)(4) and 42.105, the undersigned certifies that on January 26, 2024, a complete and entire copy of this Petition for *Inter Partes* Review of U.S. Patent No. 8,924,899 and all supporting exhibits were served, as per agreement between the parties, via email and/or electronic download to ICPillar LLC by serving the following counsel for ICPillar, LLC:

Jaime W. Marquart (Cal. Bar No. 200344)

WAYMAKER LLP

515 S. Flower Street, Suite 3500

Los Angeles, California 90071

Telephone: (424) 652-7800

jmarquart@waymakerlaw.com

John Jeffrey Eichmann (Cal. Bar No. 227472)

EICHMANN, a professional corporation

662 N. Sepulveda Blvd., Suite 300

Los Angeles, California 90049

310-237-9190 (tel.)

jeichmann@eichmann.com

Nate Lowenstein

Kenneth Weatherwax

Lowenstein & Weatherwax LLP

1016 Pico Boulevard

Santa Monica, California 90405

weatherwax@lowensteinweatherwax.com

lowenstein@lowensteinweatherwax.com

/Kevin P. Anderson/

Kevin P. Anderson, Reg. No. 43,471

kpanderson@duanemorris.com

DUANE MORRIS LLP

901 New York Ave. NW 700 E

Washington, D.C. 20001

T: (202) 776-5213

F: (202) 776-7801

ATTORNEY FOR PETITIONER

CERTIFICATE OF COMPLIANCE

Pursuant to 37 C.F.R. § 42.24 *et seq.*, the undersigned certifies that this document complies with the type-volume limitations. This document contains 13,963 words as calculated by the “Word Count” feature of Microsoft Word, the word processing program used to create it for the applicable sections.

Dated: January 26, 2024

/Kevin P. Anderson/
Kevin P. Anderson, Reg. No. 43,471
kpanderson@duanemorris.com
DUANE MORRIS LLP
901 New York Ave. NW 700 E
Washington, D.C. 20001
T: (202) 776-5213
F: (202) 776-7801

ATTORNEY FOR PETITIONER