DOCKET NO.: 0107131.00562US4

Filed on behalf of Intel Corp.
By: Jason Kipnis, Reg. No. 40,680
David Cavanaugh, Reg. No. 36,476
Joseph F. Haag, Reg. No. 42,612
Wilmer Cutler Pickering Hale and Dorr LLP
950 Page Mill Road
Palo Alto, CA 94304
Tel: (650) 858-6000
Email: Jason.Kipnis@wilmerhale.com

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

Intel Corporation
Petitioner

v.

Qualcomm Incorporated
Patent Owner

Case IPR2018-01344

PETITION FOR *INTER PARTES* REVIEW OF U.S. PATENT NO. 9,535,490 CHALLENGING CLAIMS 20, 28, AND 30

TABLE OF CONTENTS

I. INTRODUCTION

Intel Corporation ("Petitioner") respectfully submits this Petition for *Inter Partes* Review of claims 20, 28, and 30 of U.S. Patent No. 9,535,490 (the "'490 patent") (Ex-1301). The '490 patent discloses a device comprising two processors that communicate over a bus using a trigger protocol. *See id.* at claims 20, 28, and 30. However, there was nothing inventive about the claimed device as of the earliest priority date of the '490 patent, and the claimed concepts had been well-known long before the '490 patent's earliest priority date. Thus, claims 20, 28, and 30 of the '490 patent should be canceled.

II. MANDATORY NOTICES

A. Real Party-in-Interest

Intel Corporation is the real party-in-interest and submits this *inter partes* review petition for review of certain claims of U.S. Patent No. 9,535,490. Petitioner also identifies Apple Inc. ("Apple") as a real party-in-interest.

B. Related Matters

Qualcomm Incorporated ("Qualcomm" or "Patent Owner") has asserted the '490 patent against Apple in *Certain Mobile Elec. Devices and Radio Frequency Components Thereof*, Inv. No. 337-TA-1065 (Int'l Trade Comm'n) currently pending before the International Trade Commission. Qualcomm also has asserted the ’490 patent against Apple in *Qualcomm Inc. v. Apple. Inc.*, Case No. 3:17-cv-01375-DMS-MDD (S.D. Cal.).

Concurrently with this *inter partes* review petition, Petitioner is also filing *inter partes* review petitions for claims 9, 11-13, 26, and 27 of the ’490 patent (IPR2018-001346). Petitioner has also previously filed *inter partes* review petitions for claims 1-6 and 8 of the ’490 patent (IPR2018-01293), claim 31 of the ’490 patent (IPR2018-01261), and claims 16-17 and 22-24 of the ’490 patent (IPR2018-01295). Petitioner requests that these petitions be assigned to the same panel.

C. Counsel

Lead Counsel: Jason Kipnis (Registration No. 40,680)

Backup Counsel: David L. Cavanaugh (Registration No. 36,476)

Backup Counsel: Joseph F. Haag (Registration No. 42,612)

Petitioner also plans to file *pro hac vice* applications for Joseph Mueller, Nina Tallon, and Todd Zubler, each counsel of record in the pending litigation.

D. Service Information

Email: Jason.Kipnis@wilmerhale.com; David.Cavanaugh@wilmerhale.com; Joseph.Haag@wilmerhale.com.

Post and hand delivery: Wilmer Cutler Pickering Hale and Dorr LLP
950 Page Mill Road

Palo Alto, CA 94304

Telephone: 650-858-6000 Facsimile: 650-858-6100

Petitioner consents to service by email.

III. CERTIFICATION OF GROUNDS FOR STANDING

Petitioner certifies pursuant to Rule 42.104(a) that the patent for which review is sought is available for *inter partes* review and that Petitioner is not barred or estopped from requesting an *inter partes* review challenging the patent claims on the grounds identified in this Petition.

IV. OVERVIEW OF CHALLENGE AND RELIEF REQUESTED

Pursuant to Rules 42.22(a)(1) and 42.104(b)(1)-(2), Petitioner challenges claims 20, 28, and 30 of the '490 patent (Ex-1301).

A. Prior Art Patents and Printed Publications

Petitioner relies upon the following patents and printed publications:

1. U.S. Patent No. 9,329,671 to Heinrich et al. ("Heinrich") (Ex-1304), which was filed January 29, 2013 and issued May 3, 2016, is prior art to the '490 patent at least under post-AIA 35 U.S.C. § 102(a)(2).
2. U.S. Patent No. 8,160,000 to Balasubramanian ("Balasubramanian") (Ex-1305), which was filed October 3, 2006 and issued April 17, 2012, is prior art to the '490 patent at least under post-AIA 35 U.S.C. § 102(a)(1).

B. Grounds for Challenge

Petitioner requests cancellation of claims 20, 28, and 30 as unpatentable under 35 U.S.C. § 103. This Petition, which is supported by the Declaration of Dr. Bill Lin ("Lin") (Ex-1302), demonstrates a reasonable likelihood that Petitioner will prevail with respect to at least one challenged claim and that each challenged claim is unpatentable for the reasons cited herein. *See* 35 U.S.C. § 314(a).

The grounds for challenge based on the foregoing prior art references include the following:

	Grounds	**Reference(s)**	**Challenged Claims**
1.	§ 103	Combination of Heinrich and Balasubramanian	20, 28, and 30

V. TECHNOLOGY BACKGROUND

A. Processor-To-Processor Communications

The '490 patent generally relates to communications between two processing nodes within a computing device—(1) a modem processor, which typically manages the transmission and reception of data over a network (*e.g.*, over a cellular or Wi-Fi network) and (2) an application processor, which typically runs applications on the device (*e.g.*, email, text messaging, and web browsing programs). Lin-¶28.

For example, when a mobile device user composes a text message using a software application running on the device's application processor, the application processor transmits the text data to a modem processor. After processing the data to allow it to be transmitted wirelessly, the modem processor manages transmission of the data to the relevant network. Similarly, when a network transmits data to the mobile device (*e.g.*, data for a voice call or a requested web page), the modem processor processes the incoming data and then sends it to the application processor. The relevant application on the application processor can use the data (*e.g.*, a phone application can play received voice data or a web browser application can display received web page data). Lin-¶29.

These processor-to-processor communications typically occur over a wire or set of wires commonly known as a communication "bus." For instance, Figure 1C of the '490 patent below shows application processor 34 and mobile device modem ("MDM") 32 connected by interconnectivity bus 36:

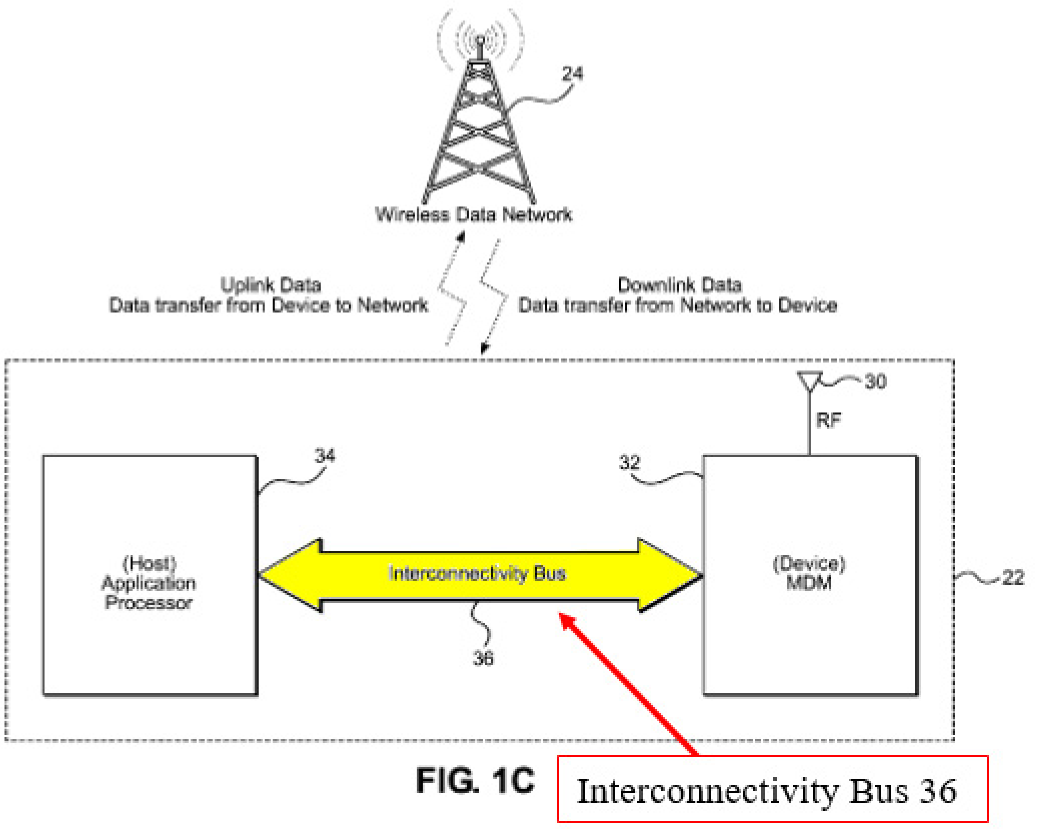

FIG. 1C

Ex-1301, Fig. 1C (annotations added). As shown in the figure, the ’490 patent refers to data sent from the application processor to the modem processor and then to the wireless data network as “uplink data”, and it refers to data sent from that network to the modem processor and then to the application processor as “downlink” data. Lin-¶30.

B. Communication Bus Power-Saving States

Like other electronic components, a communication bus must be powered for electrical signals to flow across it. But maintaining a bus in an “active” state consumes power. Therefore, buses are often designed to have one or more power-saving (“low power”) states, during which the bus or components attached to the bus are powered down (partially or fully) such that data cannot flow across the bus. Ex-1301, 8:6-19 ("In conventional mobile terminals that have a PCIe interconnectivity bus (i.e., the interconnectivity bus 36), the PCIe standard allows the interconnectivity bus 36 to be placed into a sleep mode ... This problem is not unique to the PCIe interconnectivity bus 36."). Lin-¶31.

If a processor has data to transmit to another processor, the data can be sent right away if the bus connecting them is already in an active state. However, if the bus is in a low power state at the time, the bus must transition to the active state before the processor can send the data. As the '490 patent notes, these bus transitions themselves require power. Ex-1301, 8:9-12 ("While placing the interconnectivity bus 36 in a sleep mode generally saves power, such sleep modes do have a drawback in that they consume relatively large amounts of power as they transition out of the sleep mode."). Lin-¶32.

C. Data Buffering

A first processor may have data to send to a second processor, but the communication bus between them, or the second processor itself may be inactive, thus preventing the transmission of the data. The first processor could wake the bus whenever it has data to send, but frequent transitions of the bus from a low power state to a high power state can waste power. The first processor could simply discard the data—but that would result in lost or incomplete data. Lin-¶33.

Given these obvious drawbacks, it has long been known that processor-to-processor data can be held in a "buffer" (a temporary, short-term memory) during periods when the bus (or receiving processor) is in a low power state. Lin-¶34. *See, e.g.*, Ex-1305, 5:47-51, 6:40-43, and 9:4-7; Ex-1304, 8:21-64. Collecting multiple data packets over time and sending them together after the bus transitions to an active state—rather than waking the bus from a low power state each time the processor receives a data packet—saves power by reducing the number of bus power state transitions. Lin-¶34.

Buffers, however, have limitations. Once a buffer is filled with data, additional incoming packets cannot be stored. Moreover, if a buffer stores data for too long, the data may become too old to be useful (*i.e.*, stale), or the delay may negatively affect applications expecting to receive the data. For example, in a real-time phone conversation, if voice data is held in a buffer too long, users can experience unpleasant gaps in communication. Therefore, a processor must typically send buffered data before the buffer fills up and before the data becomes stale. Lin-¶35.

The prior art describes many ways to determine when to send buffered data. One example is a "timer," which tracks how long data has been held in a buffer; when the timer expires, the buffered data is sent. *See*, *e.g.*, Ex-1304, 9:22-24; Ex-1305, 1:66-2:2; 6:55-65; 9:7-9. Another example is a "counter," which can track the number of data packets or bytes held in the buffer or count a number of specified events that occur. *See*, *e.g.*, Ex-1305, 2:3-8; 6:55-65; 9:7-9. When the counter reaches a predefined threshold (*e.g.*, a predefined number of packets, data, or events), the buffered data is sent. *Id.* Such timers and counters can be used together to ensure that (1) the buffer does not hold data for too long and (2) the buffer does not fill up. Lin-¶36.

VI. THE ’490 PATENT

The application leading to the ’490 patent (Ex-1301) was filed as U.S. Application No. 14/568,694 on December 12, 2014. The ’490 patent claims priority to U.S. Provisional Application No. 61/916,498, filed December 16, 2013, and U.S. Provisional Application No. 62/019,073, filed June 30, 2014.[1]

The ’490 patent is directed to systems and methods that claim to conserve power by limiting the number of transitions of a device between an active state and a low power state.

[1] For purposes of this Petition, Petitioner treats December 16, 2013 as the effective filing date, but does not take any position regarding whether the claims in the ’490 patent are enabled by or have written description support in the ’498 and/or ’073 provisional applications.

A. Alleged Problem of the Prior Art

The ’490 patent admits that, for prior art mobile devices containing an application processor and modem processor coupled via a communication bus, it was known that power-savings could be achieved by putting the bus into a low power state during certain periods of time. Ex-1301, 8:6-10 (“In conventional mobile terminals that have a PCIe interconnectivity bus ..., the PCIe standard allows the interconnectivity bus 36 to be placed into a sleep mode.... [P]lacing the interconnectivity bus 36 in a sleep mode generally saves power....”). According to the ’490 patent, however, these prior art devices wasted power by transitioning power states too frequently, because they: (1) transitioned the bus from a low power state to an active state to transmit downlink data, and (2) separately performed the same bus transition again at a later time to transmit uplink data—as shown in Figure 3 below:

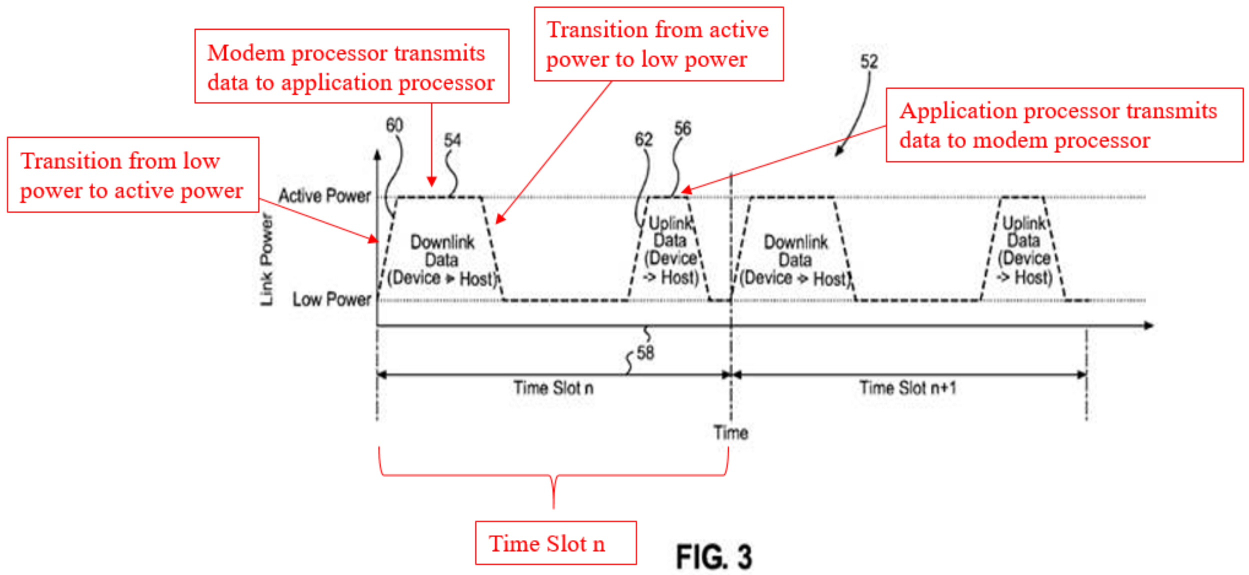
Modem processor transmits data to application processor
Transition from active power to low power
Transition from low power to active power
Application processor transmits data to modem processor
Time Slot n
52
60
54
62
56
Active Power
Low Power
Link Power
Downlink Data (Device ▸ Host)
Uplink Data (Device -> Host)
58
Time Slot n
Time Slot n+1
Time

FIG. 3

Ex-1301, Fig. 3 (annotations added); *id.*, 8:35-40 (stating that "two transitions (i.e., 60, 62) from low power to active power [] each time slot 58 … [will] consume substantial amounts of power and reduce the battery life of the mobile terminal 22"); *id.*, 8:10-12 ("[S]uch sleep modes do have a drawback in that they consume relatively large amounts of power as they transition out of the sleep mode.").

B. Purported Solution of the '490 Patent

To address this supposed "problem," the '490 patent does not claim to invent a new type of processor, a new type of communication bus, or a new type of bus power-saving state. Instead, the patent claims to improve power-savings merely by transmitting buffered downlink data followed by buffered uplink data during the ***same active period of the communications bus***—as shown in Figure 5 below:

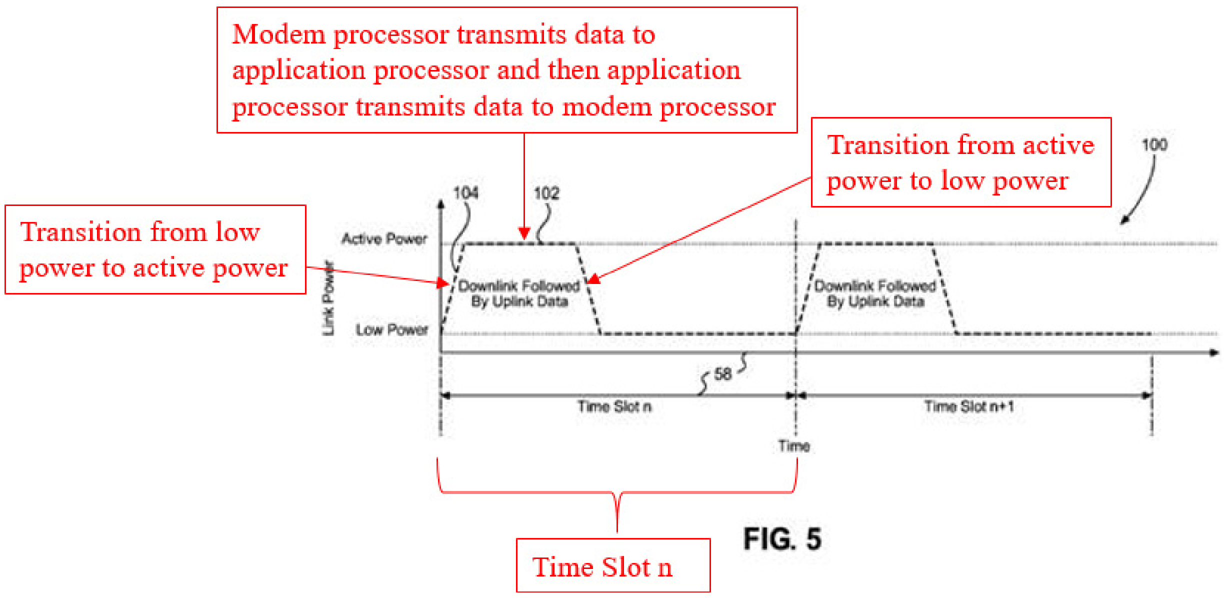
Modem processor transmits data to application processor and then application processor transmits data to modem processor
Transition from active power to low power
100
Transition from low power to active power
104
102
Active Power
Link Power
Low Power
Downlink Followed By Uplink Data
Downlink Followed By Uplink Data
58
Time Slot n
Time Slot n+1
Time
Time Slot n

FIG. 5

Ex-1301, Fig. 5 (annotations added); *id.*, 10:40-45 ("Thus, by consolidating the data into a single active period 102, the overall time that is spent in low power may be increased, thus resulting in power savings. Additionally, power spent transitioning from a low power to active power state is reduced by the elimination of the second transition 62.").

The '490 patent discusses an "exemplary" embodiment using this single bus transition scheme as shown below in Figure 4:

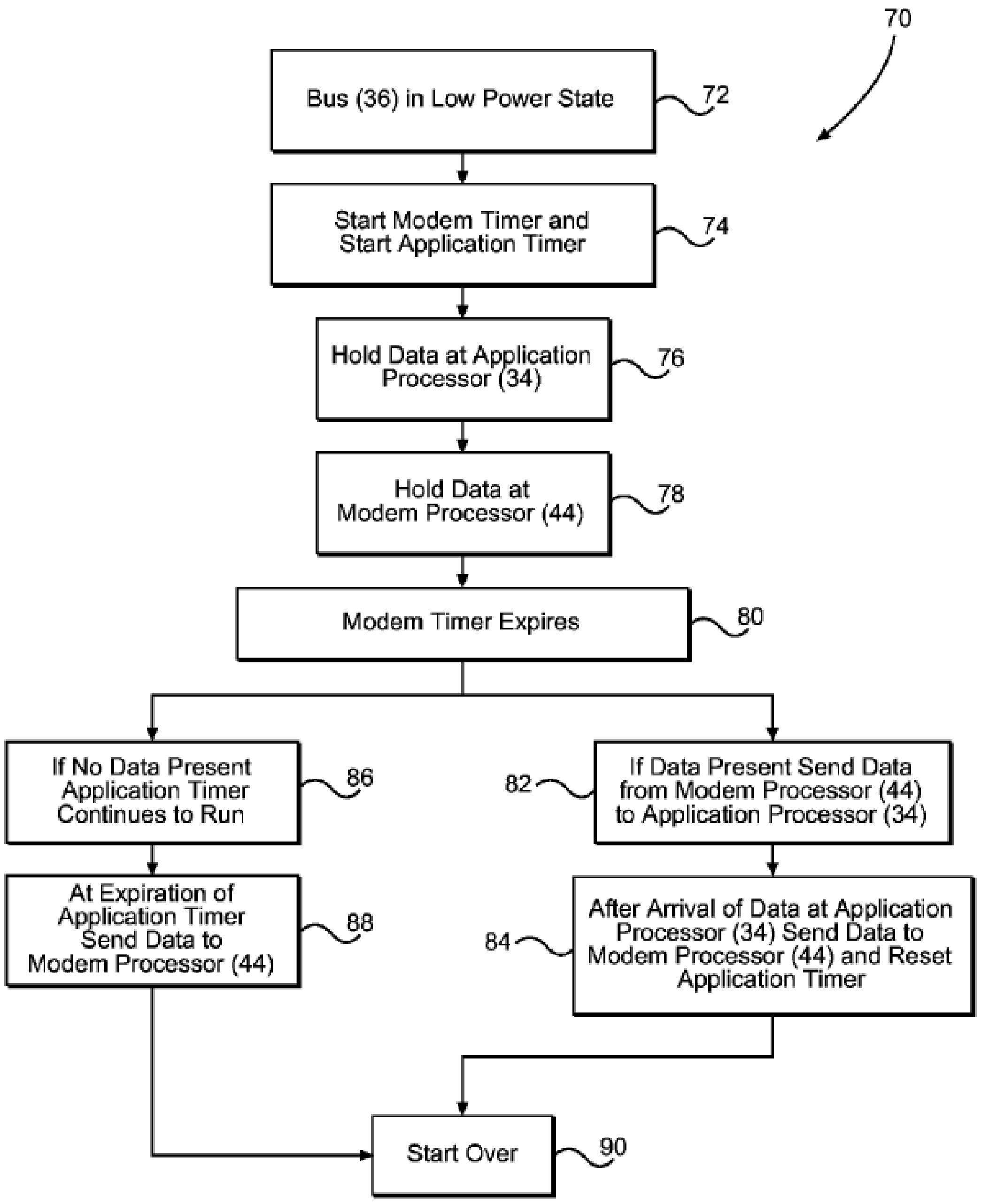
70
Bus (36) in Low Power State
72
Start Modem Timer and Start Application Timer
74
Hold Data at Application Processor (34)
76
Hold Data at Modem Processor (44)
78
Modem Timer Expires
80
If No Data Present Application Timer Continues to Run
86
82
If Data Present Send Data from Modem Processor (44) to Application Processor (34)
At Expiration of Application Timer Send Data to Modem Processor (44)
88
84
After Arrival of Data at Application Processor (34) Send Data to Modem Processor (44) and Reset Application Timer
Start Over
90

FIG. 4

Ex-1301, Fig. 4. At step 72 the bus is in a "low power" state during which data cannot be transmitted over the bus. *Id.*, 9:22-23. At step 74, a timer starts at both the modem processor and application processor. *Id.*, 9:23-27. While the bus is inactive and the timers are running, the application processor 34 holds any data that applications generate for sending to the modem processor (step 76) and the modem processor 44 holds any data that it receives from the network for sending to the application processor (step 78). *Id.*, 9:27-32 ("Data is generated by the application processor 34 and data is received from the network 12 by the modem processor 44. The application data is held at the application processor 34 (block 76) and the modem data is held at modem processor 44 (block 78) while the timers are running.").

When the modem timer expires at step 80, if the modem processor has buffered data, the bus transitions from a low power state to an active state—after which modem processor 44 sends its buffered data to application processor 34 over the communication bus (at block 82). *Id.*, 9:37-40. After receiving all the buffered data from the modem processor, the application processor treats the receipt of that data as a "trigger" that causes the application processor to send any data that it has buffered to modem processor 44 before the bus transitions back to a low power state (at block 84). *Id.*, 9:61-64.[2]

[2] For steps 86 and 88 of Figure 4, the patent explains that if the modem processor has not buffered any data when the modem timer expires, the application

As detailed further below, transmitting all accumulated downlink and uplink data during the same active state was known—long before the claimed priority date of the '490 patent—as a common sense and predictable way to reduce the power consumed by a computing system. *See, e.g.*, Ex-1305, 6:63-7:6 ("[T]he transceiver 110 ***then transmits the queued uplink packets over the communication link 116***. Advantageously, the queued packets may be grouped for transmission such that ***all of the packets are transmitted*** during ***a single wake state of the transceiver 110***. For example, as discussed above the transceiver 110 may send the queued packets in relative close succession (e.g., back-to-back) over the communication link 116. As represented by block 210, during the ***same single wake state the transceiver 110 also receives any downlink packets queued in the network interface 112***.").

Also well-known was the specific scheme disclosed in the '490 patent for how to transmit all buffered downlink and uplink data during the same active state: namely, using the receipt of buffered data from a first processing node as a "trigger" for a second processing node to send its buffered data. For example, nearly a decade before the '490 patent was filed, Balasubramanian disclosed a

processor will send any data that it has buffered to the modem processor when the application timer later expires. Ex-1301, 10:4-10.

system in which receipt of data from a first processing node "trigger[ed]" the transmission of data from a second processing node. *Id.*, 7:6-8 ("For example, the network interface 112 may use the ***receipt*** of an uplink packet as a ***trigger*** to transmit any downlink packets in its queue.").

C. Prosecution History of the '490 Patent

U.S. Application No. 14/568,694 ("the '694 application"), which issued as the '490 patent, was filed on December 12, 2014. The original application was filed with 29 claims, including 9 independent claims. On April 27, 2015, the Applicant added two more claims via preliminary amendment. Ex-1303 [April 27, 2015 Preliminary Amendment], 8.

Initially, all claims of the '490 patent were rejected over the prior art. The Examiner allowed the application only after the Applicant amended most of the claims to include the "trigger" limitation—which requires a second processor to be triggered to send held data to a first processor in response to receiving data from the first processor. In particular, on June 10, 2016, the Examiner rejected all pending claims over PCT Publication No. WO 2009/039034 ("the Intel PCT") (Ex-1306) and U.S. Patent No. 6,021,264 ("Morita") (Ex-1307). *See* Ex-1303 [6/10/2016 Office Action], 3-9. In response to this rejection, the Applicant amended claim 15 (which issued as claim 16) to include the "trigger" limitation as shown below:

15. (Currently Amended) A method of controlling power consumption in a computing device, comprising:

holding data received by a modem processor from a remote network until expiration of a downlink timer;

passing the data received by the modem processor to an application processor over an interconnectivity bus; and

holding application data generated by an application associated with the application processor until receipt of the data from the modem processor or expiration of an uplink timer, whichever occurs first,

wherein receipt of the data from the modem processor triggers passing the data received by the application processor to the modem processor over the interconnectivity bus before the interconnectivity bus transitions from an active power state to a low power state.

Ex-1303 [8/24/2016 Response], 2. The Applicant similarly amended original independent claims 1, 20, and 24-28 to include a "trigger" limitation. *Id.*, 14 (commenting on the amendment to claim 1 and explaining that "[i]ndependent claims 15, 20, and 24-29 have been amended to recite similar features and are therefore allowable for at least the same reasons as claim 1....").

The Applicant argued that Morita does not disclose that the transmission of data from the application processor to the modem processor is "triggered by" receipt of data from the modem processor. *Id*. Instead, the Applicant argued, Morita discloses "delay[ing]" transmission of data "by a predetermined time length." *Id*. As a result, the Applicant argued, that transmission could not be "triggered by" or "responsive to" receipt of data from the modem processor because the data is held for a "predetermined time length," which, by its definition, is a length of time determined in advance of receiving any data. *Id*. The Applicant did not separately argue that the Intel PCT fails to disclose any of the other limitations. *Id.*, 13-15.

The Examiner subsequently allowed the claims as amended.[3]

VII. CLAIM CONSTRUCTION

Petitioner has set forth below its proposed constructions of certain terms of the '490 patent and its support for the constructions. 37 C.F.R. 42.100(b) states that claims must be given their broadest reasonable construction in light of the specification ("BRI standard"). On May 8, 2018, the USPTO proposed rulemaking that would change the standard for construing claims from the BRI standard to the

[3] The claims were re-ordered such that original claims 15, 20, and 24-29 became issued claims 16, 22, and 26-31, respectively.

Phillips standard. In anticipation that the rule change will apply to these proceedings, Petitioner construes the claims based on the standard set forth in Phillips. Petitioner is not aware of any difference in how the claims would be construed under the BRI standard. The scope of the challenged claims could not be broader under the proposed Phillips construction than it could be under the BRI standard. Therefore, the challenged claims would also be unpatentable under the BRI standard.

A person of ordinary skill in the art ("POSA") of the '490 patent would have had a Master's degree in Electrical Engineering, Computer Engineering, or Computer Science plus at least two years of experience in mobile device architecture and multiprocessor systems, or alternatively a Bachelor's degree in one of those fields plus at least four years of experience in mobile device architecture and multiprocessor systems.

A. "triggered by" / "triggers"

As used in the '490 patent, a POSA would have understood the phrase "triggered by" to mean "initiated in response to". The '490 patent does not specifically define the term "trigger," but implies that "triggering" occurs when an action is taken in response to a stimulus. *See* Ex-1301, 2:3-6 ("On receipt of the data from the modem processor, the application processor sends data held by the application processor to the modem processor over the PCIe interconnectivity bus."); *see also id.*, 11:15-18, 11:39-46. This understanding of "trigger" is supported by the claims themselves. For example, claim 1 states that the application processor is "triggered by the receipt of" data from the modem processor, and suggests that this is synonymous with being "responsive to the receipt of" data from the modem processor. *See id.*, claim 1. The prosecution history also supports this understanding of "triggered by." *See* Ex-1303 [8/24/2016 Response], 14 ("[T]here is no disclosure or suggestion that the 'predetermined time length' is even capable of being 'triggered by' or otherwise 'responsive to the receipt of the modem processor to application processor data,' as required by claim 1."). And, this understanding of "triggered by" is also supported by various dictionaries. *See* Ex-1312 [*Heritage Dictionary*], 1444 ("trigger[:] ... 1. to set off; initiate"); Ex-1313 [*Oxford Dictionary*], 1541 ("trigger ... cause to happen or exist"); Ex- 1314 [*Merriam-Webster's Dictionary*], 1337 ("trigger[:] ... to initiate, actuate, or set off by a trigger").

VIII. OVERVIEW OF PRINCIPAL PRIOR ART REFERENCES

A. U.S. Patent No. 9,329,671 To Heinrich et al.

U.S. Patent No. 9,329,671 to Heinrich et al. was filed January 29, 2013 and issued May 3, 2016. Heinrich is therefore prior art under at least 35 U.S.C. § 102(a)(2).

As shown below, Heinrich discloses a device with a baseband processor 104 and an application processor 106 that communicate with each other over an interconnectivity bus:

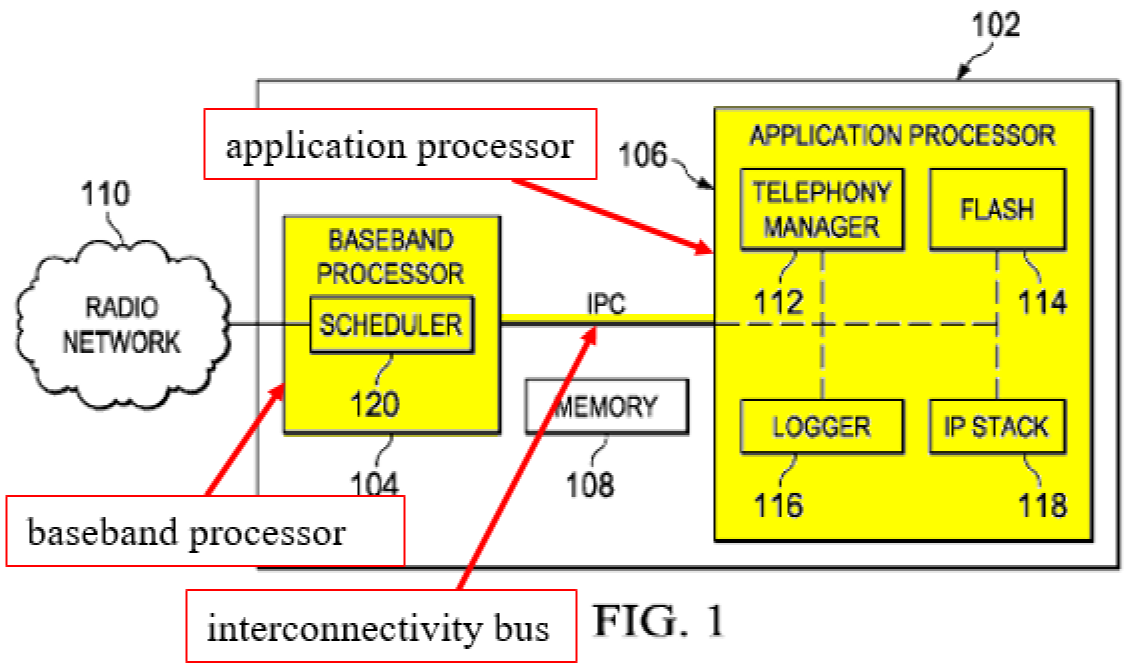

FIG. 1

Ex-1304, Fig. 1 (annotations added); *id.*, 4:26-46 ("FIG. 1 shows ... a ***baseband processor 104***..., an ***application processor (AP) 106***..., [and] a physical interface configured for communicating IPC activities between the baseband processor 104 and the application processor 106.").

In Heinrich, data communications over the interconnectivity bus—referred to as "IPC [inter processor communication] activities"—are buffered so that transmission to the application processor can be delayed. *Id.*, 7:65-8:1 ("IPC activities that are not real-time sensitive... can be ***delayed*** until it is deemed profitable to run them."); 8:14-15 ("Those IPC activities which are not real-time sensitive can be ***delayed***."). A scheduler implemented as software in the baseband processor includes "timers" that, upon expiration, trigger transmission of the buffered data over the bus to the application processor. *Id.,* 7:7-17 ("[T]here is a ***centralized scheduler 120*** which is associated with the baseband processor 104...."); 9:2-6 ("[T]he scheduler 120 allocates a respective ***timer***, herein referred to as a '***lazy timer'***, to each of the non real-time sensitive IPC activities identified in step S302...."); 9:14-16 ("***When one of the lazy timers fires this causes the respective IPC activity to be communicated on the IPC interface to the application processor 106***,...").

Each data communication ("IPC activity") is assigned a timer. *Id.*, 9:2-6. When any one of the assigned timers expires ("fires"), the baseband processor sends the buffered data over the bus to the application processor. *Id.*, 9:11-21. Heinrich explains that, by grouping together the transmissions, the processor can reduce the number of times that the application processor enters and exits sleep mode, thereby reducing the power consumption of the computer system. *Id.*, 4:6-12.

Heinrich also discloses that the same technique—buffering data and sending it all at once upon expiration of a timer—can be used to control transmissions from the application processor over the bus to the baseband processor. *Id.*, 12:52-55 ("A

scheduler may implement the same scheduling techniques as those described above, but configured to schedule IPC activities ***from the application processor 106 to the baseband processor 104***.").

Heinrich also discloses that the scheduler of its baseband processor can detect that the physical IPC interface is in an active state, and the application processor is awake, when the baseband processor transmits data to the application processor. *Id.*, 9:43-46 ("In a second example, the scheduler 120 can deduce that the application processor 106 will be in the awake mode by determining that real-time sensitive IPC activities are being sent to the application processor 106...."); 9:54-62 ("The scheduler 120 can perform the determination that the application processor 106 is in the awake mode by receiving a notification that the physical IPC interface is in an active state... [N]on real-time sensitive IPC activities are sent to the application processor 106 at a time when the application processor 106 is in the awake mode due to the communication of a previous real-time sensitive IPC activity.").

B. U.S. Patent No. 8,160,000 to Balasubramanian

U.S. Patent No. 8,160,000 to Balasubramanian was filed October 3, 2006 and issued April 17, 2012. Balasubramanian is therefore prior art under at least 35 U.S.C. § 102(a)(1).

Balasubramanian describes a system where two processing nodes—for example, a transceiver 110 in a user device such as a cell phone, and a network interface 112 to a packet-switched network—communicate with each other over a communication link 116, which can be a wired connection:

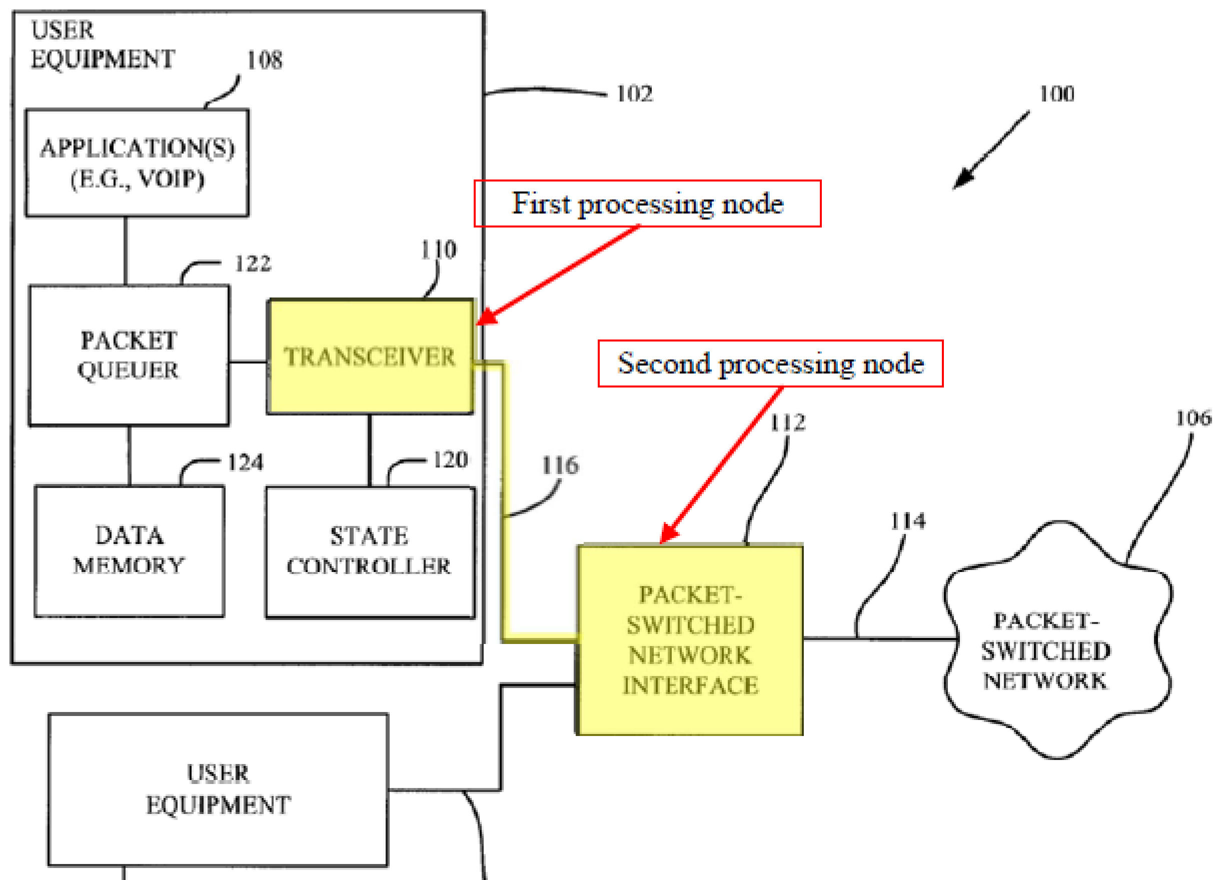

FIG. 1

Ex-1305, Fig. 1 (annotations added).

Balasubramanian discloses a scheme to synchronize transfers over the communication link 116, in which the transceiver 110 sends packets to the network interface 112, which ***triggers*** the network interface 112 to send any buffered packets back to the transceiver 110 during the same active state of the transceiver 110. *Id.*, 6:63-7:8 ("[T]he transceiver 110 ***then transmits the queued uplink packets over the communication link 116***. Advantageously, the queued packets may be grouped for transmission such that ***all of the packets are transmitted*** during a single wake state of the transceiver 110.... As represented by block 210, during the ***same single wake state the transceiver 110 also receives any downlink packets queued in the network interface 112***. For example, the network interface 112 may use the ***receipt of an uplink packet as a trigger*** to transmit any downlink packets in its queue."); *see also id.,* 2:23-24.

Balasubramanian discloses two processing nodes communicating over a wired link, like the bus in Heinrich. *Id.*, 4:50-53 ("[T]he subnetwork [which includes the transceiver 110 and the network interface 112] may communicate via some other protocol (e.g. a wire-based protocol or a wireless-based protocol) over communication links 116 and 118."). Lin-¶61.

Like Heinrich and the '490 patent, Balasubramanian is directed to conserving power by reducing the number of transitions from a low power mode to an active mode. Balasubramanian explains that a "transceiver" must transition from a "suspend" (low power) state to an "active" state to allow the transceiver to transmit its buffered data to the network interface. According to Balasubramanian, reducing the number of times the transceiver transitions from a suspend state into an active state reduces power consumption. Ex-1305, 5:55-61 ("Here, ***power may be conserved*** by not transitioning the transceiver 110 from the suspended state to the active state every time a packet has been generated for transmission.... Accordingly, power savings may be achieved by rescheduling the packet traffic into groups of traffic.").

In order to reduce the number of power state transitions, Balasubramanian uses the same technique as the '490 patent—buffering data intended for another processor during periods when the communication bus or other processor is inactive, and then later transmitting the buffered data from both processors during the same active state. *Id.*, 5:65-6:4 ("[T]he user equipment 102 may include a packet queued component 122 that facilitates queuing and temporarily ***storing the packets***.... When the transceiver 110 is transitioned to an active state, the queued packets may be provided to the transceiver 110 for transmission to the network 106 (via interface 112)."); *id.*, 6:65-67 ("Advantageously, the queued packets may be grouped for transmission such that ***all of the packets are transmitted during a single wake state*** of the transceiver 110.").

In addition to the "trigger" mechanism described above, Balasubramanian also uses a timer to determine when to transmit packets from the transceiver 110 to the network interface 112 (similar to Heinrich and the '490 patent). *Id.*, 9:33-34 (describing a "***timer***/counter 426 to determine when to make the packets in the packet queue 422 available....”); *id.*, 6:48-49 (“[T]he packets may be queued for configurable amount of time.”); *id.*, 6:55-60 (“[O]nce the ***configurable amount of time has elapsed*** or the configurable number of packets have been queued, the ***transceiver 110 transitions*** to an active (e.g., wake) state…, and ***establish[es] communications with the network interface 112***.”); *id.*, 9:7-9 (“[P]ackets may be queued for a configurable amount of time….”).

IX. SPECIFIC GROUNDS FOR PETITION

Pursuant to Rule 42.104(b)(4)-(5), the following sections describe in detail how the prior art discloses each and every limitation of the challenged claims of the ’490 patent, and how the prior art renders these claims obvious. *See also* Lin-¶¶ 71-156.

A. Ground 1: Claims 20, 28, And 30 Are Rendered Obvious By Heinrich In View Of Balasubramanian

Claim 20 depends from claim 16. While claim 16 is not challenged in this petition, a mapping of claim 16 is provided below because of this dependency.

1. Claim 16

a) [16a] Preamble: “[a] method of controlling power consumption in a computing device”

Heinrich discloses “[a] method of controlling power consumption in a computing device.” Heinrich discloses a computing device that can be “a mobile phone, a tablet, a laptop computer or other embedded device able to connect to the network." *See* Ex-1304, 4:18-43. Heinrich teaches that "[i]t is beneficial to minimize the power consumed by the computer system." *Id.*, 1:54-2:8. To minimize power consumption, Heinrich teaches buffering data that is not real-time sensitive and sending it all at once to reduce the number of times the bus and/or processors transition from a low power or sleep mode into a high power or active mode. *Id.*, 4:6-11 ("By grouping the non real-time sensitive IPC activities together and scheduling them for communicating to the second processor during a period in which the second processor is continuously in the first [active] mode, the number of times that the second processor enters and exits the second mode (e.g. sleep mode) is reduced."). Thus, Heinrich discloses a computing device that performs "[a] method of controlling power consumption in a computing device." Lin-¶72.

b) [16b] "holding data received by a modem processor from a remote network until expiration of a downlink timer"

Heinrich discloses "holding data received by a modem processor from a remote network until expiration of a downlink timer."

As shown in Figure 1, Heinrich discloses a mobile communication system that includes a "baseband processor 104":

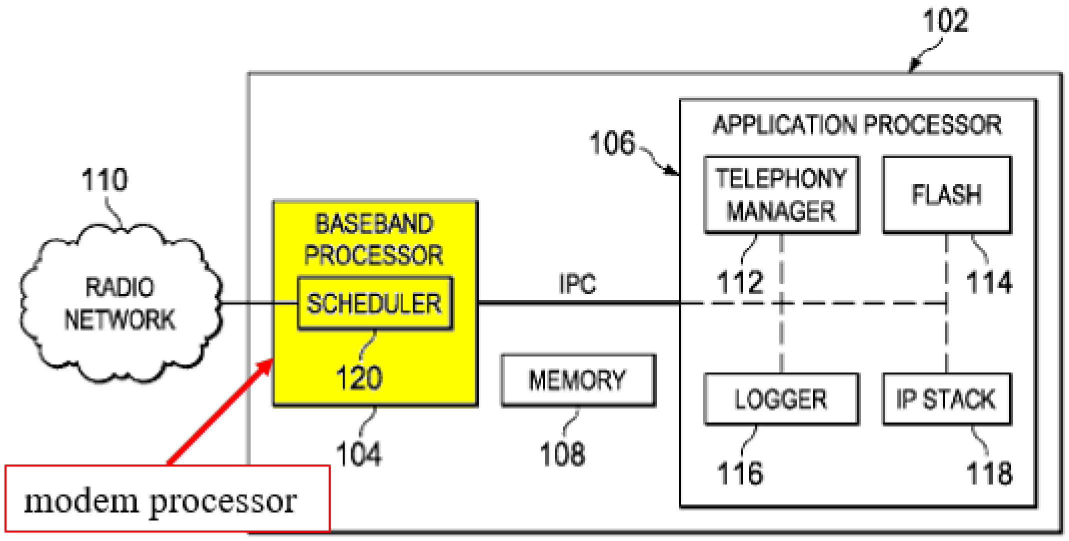

FIG. 1

Ex-1304, Fig. 1 (annotations added); Lin-¶74.

Heinrich discloses that baseband processor 104 is a "modem" processor that communicates with a network. Ex-1304, 4:30-36 ("The baseband processor 104 acts as a Radio Frequency (RF) ***modem*** to process data for ***communication between the user device 102 and the network 110*** .... [T]he baseband processor 104 is implemented at the user device 102 for ***communicating with the radio network 110***."). The modem processor of Heinrich—*i.e.*, baseband processor 104—communicates with and receives data from a remote network (*e.g.*, network 110). *Id.*

As the '490 patent explains, a "downlink timer" is a "modem timer" that, upon expiration, triggers the transmission of held data from the modem processor to the application processor. Ex-1301, 9:37-40 ("The modem timer expires (block

80). If modem data is present, the modem data is released by the modem processor 44 through the interconnectivity bus 36 to the application processor 34 (block 82)."); *id.*, 10:13 ("the downlink timer (i.e., the modem timer)"), *id.*, 11:32-33 ("the downlink timer (i.e., the modem timer)"). Thus, the lazy timer on the baseband processor in Heinrich is a downlink timer that, upon expiration, triggers the transmission of baseband data from the baseband processor to the application processor over the bus. Heinrich teaches that baseband processor 104 delays sending certain data to the application processor 106 over the IPC interface, and transmits such data upon expiration of a downlink timer. Ex-1304, 7:65-8:1 ("[T]he scheduler 120 identifies IPC activities that . . . can be ***delayed*** until it is deemed profitable to run them."). These transactions are aggregated for later transmission. *Id.*, 9:5-13 ("Each IPC activity is sent on the IPC interface when its respective lazy timer fires.... However, when one of the registered timers fires, ***all registered timers expire at the same time, causing all the aggregated IPC activities to be served at the same time***."). Heinrich also explains that buffering communications and sending them all at once when a timer expires results in "fewer transitions between the sleep and awake modes" and thereby "reduce[s] the power consumed." *Id.*, 10:44-51. A POSA would understand that such delayed and "aggregated IPC activities" would include "data received by a modem processor from a remote network" that would have to be held (i.e., buffered) until the baseband processor determines that it is time to transmit the held data to the application processor, such as when the timer expires. *See also id.*, 11:57-58 ("In order to delay file write accesses they are stored in a cache memory of the baseband subsystem."); Lin-¶76.

Heinrich describes "baseband memory" where baseband data is buffered. Ex-1304, 11:58-66 ("The scheduler 120 schedules the file write accesses as described above such that a group of them may be retrieved from the cache memory of the baseband subsystem together and sent to the application processor 106 on the IPC interface as a group.... These files may be cached in ***baseband memory*** with no user impact.").

It would have been obvious to a POSA that the baseband processor in Heinrich can hold data received from a remote network in its own memory on the same chip as the processor circuitry. A processor can hold data in two known ways, on-chip or off-chip. Both well-known ways of holding data are discussed, for example, in Panda et al., "On-Chip vs. Off-Chip Memory: The Data Partitioning Problem in Embedded Processor-Based Systems," Ex-1311 [Panda, ACM Transactions on Design Automation of Electronic Systems, Vol. 5, No. 3, July 2000], 682-704.

First, the processor could hold the data in its own memory—*i.e.*, on the same chip that includes the processor circuitry, such as in on-chip SRAM. *Id.*, 683

("The types of on-chip memory commonly integrated with the processor on the same chip are instruction cache, data cache, and on-chip SRAM...[I]t is possible to incorporate *embedded DRAMs* along with a processor core in the same chip....").

Second, the data could be held on a separate or external chip—such as a memory chip. *Id.*, 683 ("[A]ccess to an off-chip memory (usually DRAM) requires relatively longer access times."). *See also id.*, 686 (Fig. 2 (annotations added), showing offchip DRAM storage and onchip SRAM and data cache storage):

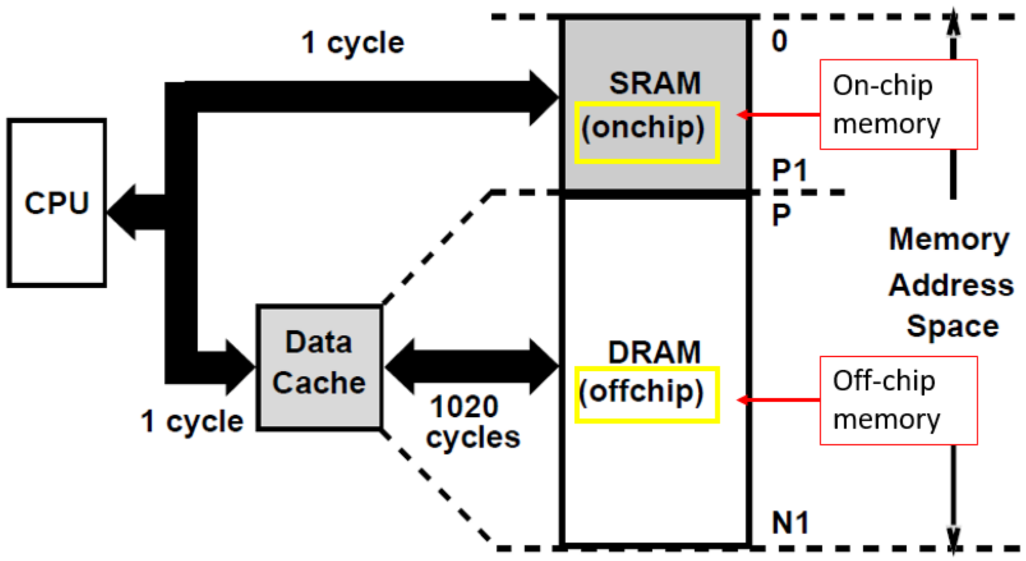

Fig. 2. Division of data address space between SRAM and DRAM.

A POSA would have been motivated to use "on-chip" memory to store data on the baseband processor in Heinrich for at least two reasons. ***First***, using "on-chip" memory to store a processor's data requires less physical space than using a separate chip dedicated to memory. *See id.*, 682-683. This allows for fewer chips in a device, reducing cost. *Id.*, 683. ***Second***, accessing data from "on-chip" memory is much faster than accessing data from another chip over a separate connection, which improves device operation. *Id.* Because using on-chip memory to hold data was well-known in systems like Heinrich, a POSA would have reasonably expected the design to succeed. Lin-¶81. "When there is a design need or market pressure to solve a problem and there are a ***finite number of identified, predictable solutions***, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp." *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 421 (2007).

c) [16c] "passing the data received by the modem processor to an application processor over an interconnectivity bus"

Heinrich discloses "passing the data received by the modem processor to an application processor over an interconnectivity bus." As shown in Figure 1, Heinrich discloses an interconnectivity bus labeled "IPC" ("inter processor communication") that couples the baseband processor 104—*i.e.*, modem processor—to the application processor 106 and allows for communication between the two processors:

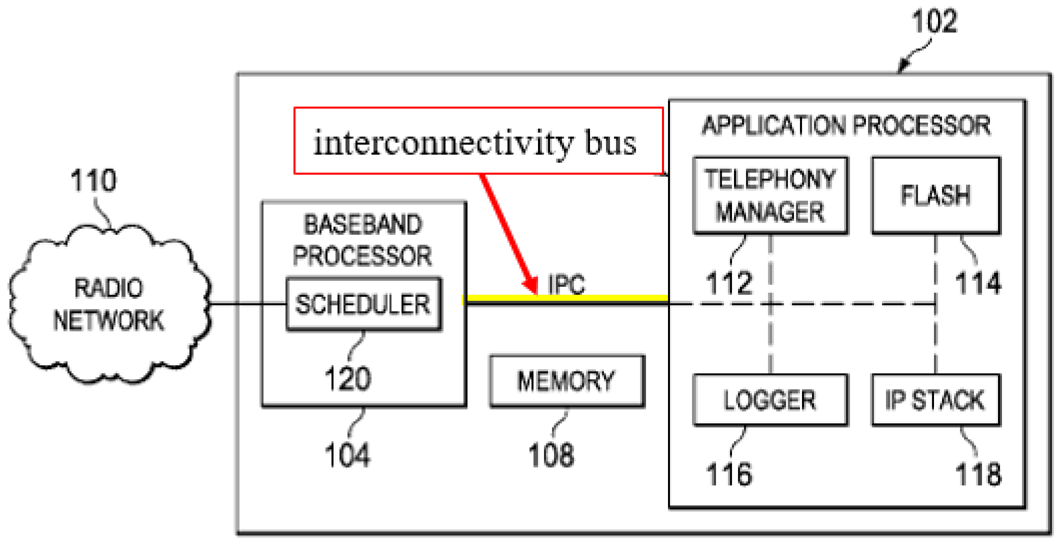

FIG. 1

Ex-1304, 4:44-46 ("There is a physical interface ***configured for communicating IPC activities*** between the baseband processor 104 and the application processor 106."); *id.*, Fig. 1 (annotations added).

The IPC bus in Heinrich is a "physical interface" that can be any one of many industry-standard buses including "one of: (i) a Universal Serial Bus (USB) interface, (ii) a Mobile Industry Processor Interface (MIPI), such as a High-Speed Synchronous Interface (HSI), [or] (iii) a Serial Peripheral Interface (SPI) ...." *Id.*, 4:46-50.

The baseband processor of Heinrich receives data from the network. *Id.*, 4:30-33 ("The baseband processor 104 acts as a Radio Frequency (RF) modem to process data for communication between the user device 102 and the network 110...."). The baseband processor then passes the data received from the network to the application processor over the IPC bus. *Id.*, 4:65-5:11 ("Communication between the two sub-systems (i.e. communication between the processors 104 and 106) is referred to as Inter Processor Communication (IPC). 'IPC activities' are communications between the two processors 104 and 106. The IPC activities convey various types of information..., including... data, (e.g. Internet Protocol (IP) data for transmission between the user device 102 and another node of the radio network 110, which may be processed by the IP stack module 118)...."); Lin-¶84.

d) [16d] "holding application data generated by an application associated with the application processor until receipt of the data from the modem processor or expiration of an uplink timer, whichever occurs first"

Heinrich in view of Balasubramanian discloses "holding application data generated by an application associated with the application processor until receipt of the data from the modem processor or expiration of an uplink timer, whichever occurs first."

"holding application data generated by an application associated with the application processor": The combination of Heinrich and Balasubramanian discloses "holding application data." As discussed above, Heinrich discloses that data is buffered prior to being sent over the bus from the baseband [modem] processor to the application processor. *See* claims [16b], [16c]. Heinrich further explains that the same methods used to transmit data from the baseband [modem] processor to the application processor—*i.e.*, buffering data and sending it all at once—can be used to transmit data from the application processor to the baseband [modem] processor. Ex-1304, 12:52-55 ("A scheduler may implement the ***same scheduling techniques*** as those described above [for scheduling IPC activities from the baseband processor 104 to the application processor 106], but configured to schedule IPC activities ***from the application processor 106 to the baseband processor 104***." (emphasis added)); Lin-¶86.

It would have been obvious to a POSA to construct Heinrich's application processor to hold data for the same reasons described in claim [16b]. While Heinrich does not expressly specify where such data is held, it would have been obvious to a POSA that the application processor in Heinrich can hold its data—*i.e.*, the application data generated by an application associated with the application processor—in its own memory on the same chip as the processor circuitry.

As explained above in claim limitation [16b], a POSA would have understood that there were two well-known ways for data to be buffered: (1) on the same chip that includes the processor circuitry; or (2) on a separate or external chip. *See* Ex-1311, 683, 686 (Fig. 2); Lin-¶88. As with the baseband processor, a POSA would have been motivated to use "on-chip" memory to store data on the application processor in Heinrich for at least the two reasons identified above regarding claim limitation [16b].

Heinrich further discloses "data generated by an application associated with the application processor." In particular, Heinrich discloses that the application processor generates data from various multimedia applications, including, for example, applications that generate Internet Protocol data intended for communication over the remote network via the IP stack module. Ex-1304, 1:35-39; 4:51-56; 4:67-5:11 ("'IPC activities' are communications between the two processors 104 and 106. The IPC activities convey various types of information …, including … data, (e.g. Internet Protocol (IP) data for transmission between the user device 102 and another node of the radio network 110, which may be processed by the IP stack module 118)...."). Lin-¶89.

To the extent the Patent Owner argues that Heinrich does not explicitly disclose "data generated by an application associated with the application processor," a POSA would have understood the limitation to be inherent in Heinrich's disclosure of an application processor that includes an IP stack module. The IP stack module in Heinrich is used to process IP data transmitted between user device 102 and another node of the radio network 110. It would have been obvious to a POSA that, in order to generate such IP data intended for transmission over the network, an ***application*** processor—as its name implies—would execute an application. *See* Lin-¶90.

Thus, Heinrich in combination with Balasubramanian discloses "holding application data generated by an application associated with the application processor…." Lin-¶85.

<u>"until receipt of the data from the modem processor:"</u>

The combination of Balasubramanian and Heinrich discloses holding application data "until receipt of the data from the modem processor."

Heinrich teaches various techniques to determine when to send data from the application processor to the baseband processor. These techniques include (1) a timer that triggers transmission of data when the timer expires or (2) more generally, upon any other determination that informs the application processor that the baseband processor is in an "awake mode" and can therefore receive data. Ex-1304, 9:22-26 ("In general, each lazy timer is configured to fire in response to the earlier of: (i) the expiry of a respective deadline provided to the lazy timer before which it is expected to fire, or (ii) a determination that the application processor 106 is in the awake mode.").[4]

[4] The quoted portion of Heinrich describes the transmission of data from the baseband processor. However, as explained above, Heinrich discloses that the same techniques can be used to control transmission of data from the application processor to the baseband processor. *See* Ex-1304, 12:52-55.

Heinrich also discloses that the scheduler of its baseband processor can detect that the physical IPC interface is in an active state, and the application processor is awake, when the baseband processor transmits data to the application processor. *Id.*, 9:50-62 (emphasis added) ("In order to determine that ***real-time sensitive IPC activities are being sent to the application processor 106***, the scheduler 120 registers to the underlying physical IPC interface in order to receive notifications of state changes of the IPC interface. The ***scheduler 120 can perform the determination that the application processor 106 is in the awake mode by receiving a notification that the physical IPC interface is in an active state***. When the IPC interface enters an active state the scheduler 120 deems it appropriate to fire all registered lazy timers. In this way the non real-time sensitive IPC activities are sent to the application processor 106 at a time when the application processor 106 is in the awake mode due to the communication of a previous real-time sensitive IPC activity."). Because Heinrich notes that the scheduling techniques described with respect to the baseband processor are also applicable to the application processor (*see id.*, 12:52-55), Heinrich teaches that its application processor can include a scheduler that can detect that the physical IPC interface is in an active state, and the baseband processor is awake. *Id.*, 9:50-62, 12:47-64; Lin- ¶94.

While Heinrich does not explicitly disclose holding data intended for transmission to another processor until receipt of data from the other processor, Balasubramanian discloses such a technique. In particular, Balasubramanian discloses holding (*e.g.*, queuing), at a second processing node (*e.g.*, network interface 112), data (*e.g.*, downlink packets) to be sent to a first processing node (*e.g.*, transceiver 110) until "trigger[ed]" by "receipt" of data (*e.g.*, uplink packets) from the first processing node over a bus (*e.g.*, communication link 116):

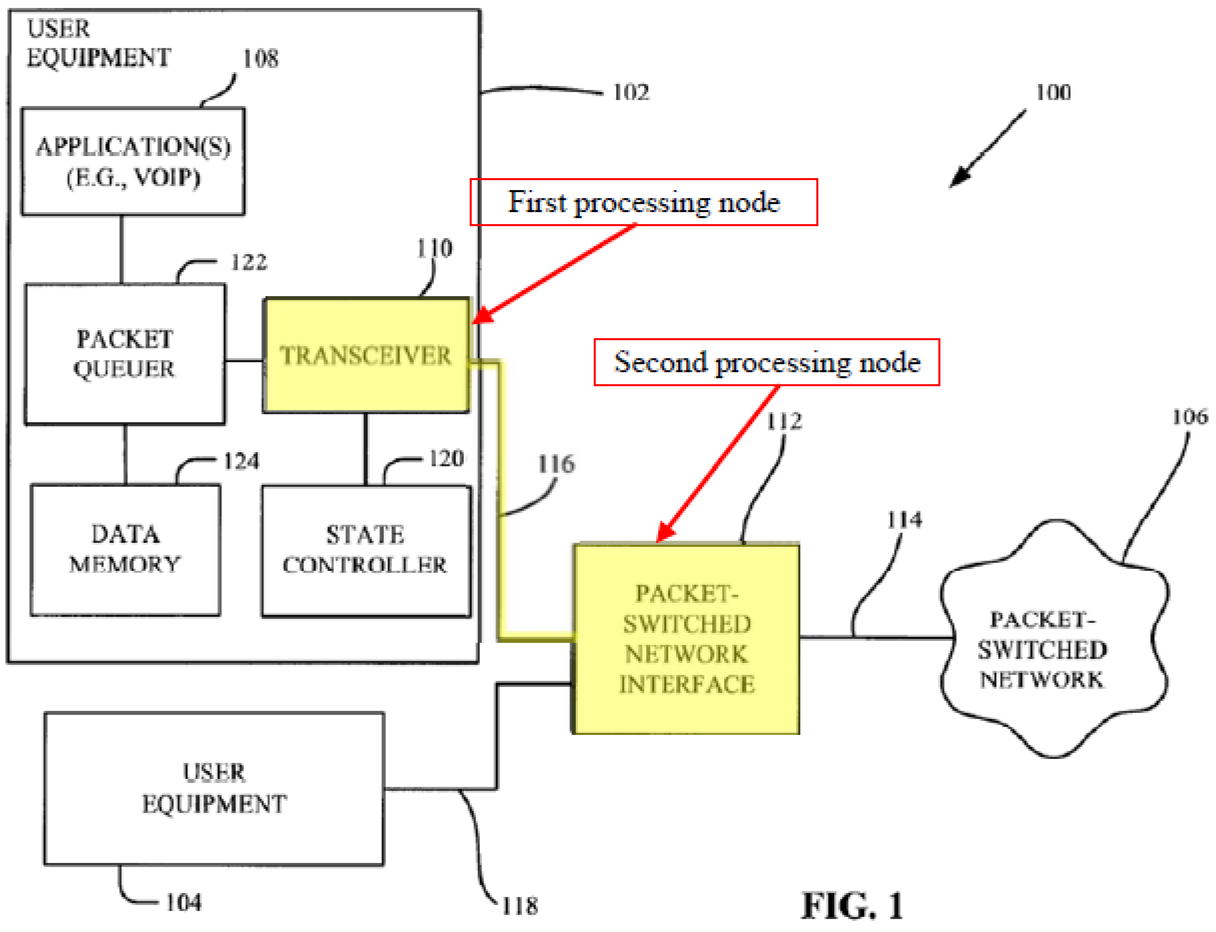

FIG. 1

Ex-1305, Fig. 1 (annotations added); *id.*, 6:63-7:8 ("As represented by block 208, the ***transceiver 110 then transmits the queued uplink packets over the communication link 116***. Advantageously, the queued packets may be grouped for transmission such that ***all of the packets are transmitted*** during a single wake state of the transceiver 110. . .As represented by block 210, during the same single wake state the transceiver 110 also receives any downlink packets queued in the network interface 112. For example, the ***network interface 112*** may use the ***receipt*** of an ***uplink*** packet as a ***trigger*** to transmit any ***downlink*** packets in its queue.").

Figure 2 of Balasubramanian also discloses this feature. Consistent with the disclosure in Balasubramanian, column 6, line 63 through column 7, line 8, Figure 2 discloses that at step 208, the transceiver 110 transmits all its queued packets over the communication link 116 to the network interface 112. In response, as seen in step 210, the network interface 112 will transmit its queued packets to the transceiver 110.

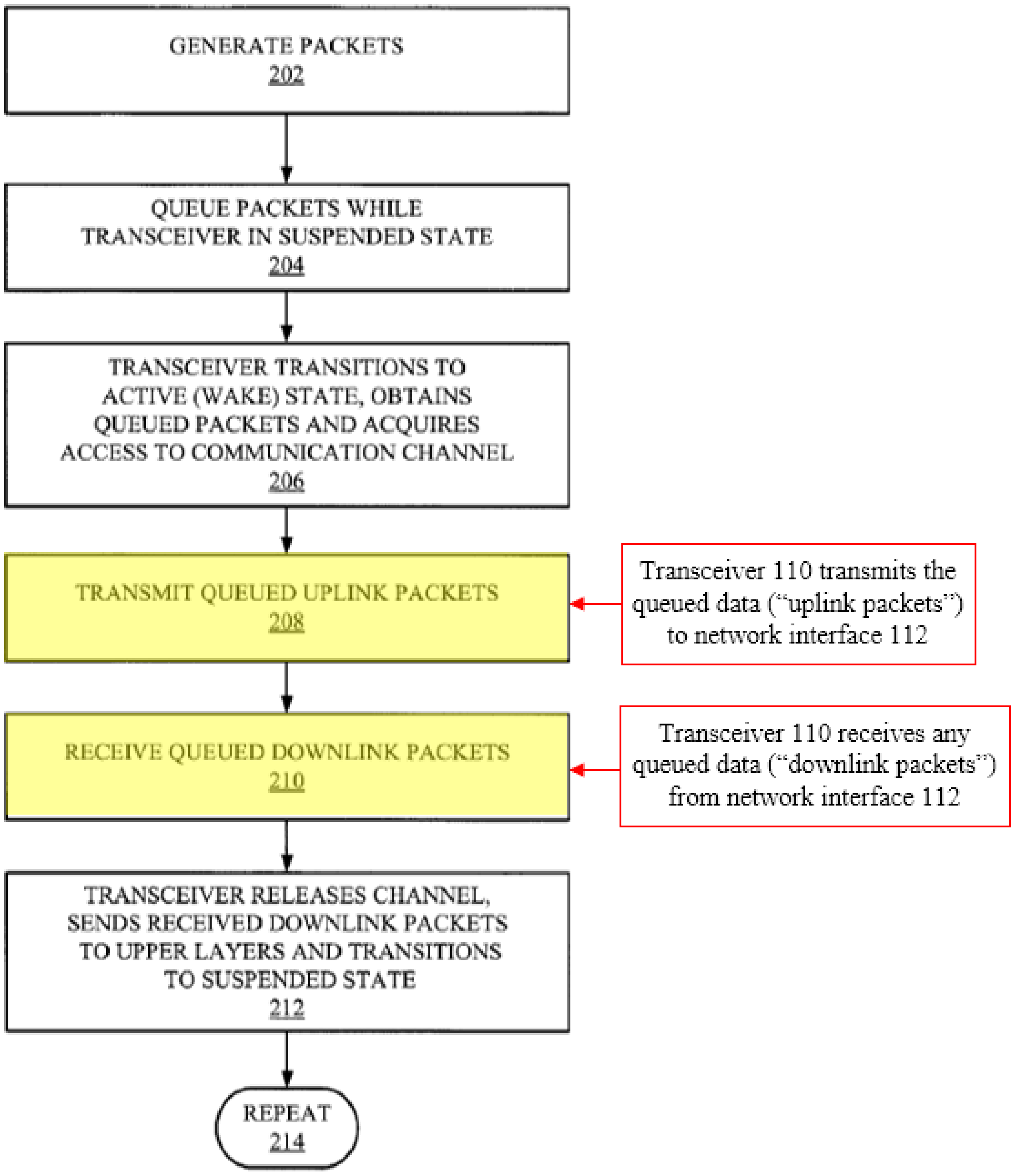
GENERATE PACKETS
202
QUEUE PACKETS WHILE TRANSCEIVER IN SUSPENDED STATE
204
TRANSCEIVER TRANSITIONS TO ACTIVE (WAKE) STATE, OBTAINS QUEUED PACKETS AND ACQUIRES ACCESS TO COMMUNICATION CHANNEL
206
TRANSMIT QUEUED UPLINK PACKETS
208
Transceiver 110 transmits the queued data ("uplink packets") to network interface 112
RECEIVE QUEUED DOWNLINK PACKETS
210
Transceiver 110 receives any queued data ("downlink packets") from network interface 112
TRANSCEIVER RELEASES CHANNEL, SENDS RECEIVED DOWNLINK PACKETS TO UPPER LAYERS AND TRANSITIONS TO SUSPENDED STATE
212
REPEAT
214

FIG. 2

Ex-1305, Fig. 2 (annotations added); *id.*, 6:63-7:8.

Accordingly, Balasubramanian discloses that the "trigger" for the transmission of data from the second processing node (network interface 112) to the first processing node (transceiver 110) is the reception at the second processing node of the queued packets from the first processing node. Lin-¶97. Thus, the combination of Heinrich and Balasubramanian teaches the claim limitation "until receipt of the data from the modem processor."[5]

<u>"holding application data … until … expiration of an uplink timer":</u>

Heinrich discloses holding application data until a timer—referred to as a "lazy timer"—expires. *See, e.g.*, Ex-1304, 8:14-15 ("Those IPC activities which are not real-time sensitive can be delayed."); 9:2-5 ("[T]he scheduler 120 allocates a respective timer, herein referred to as a 'lazy timer', to each of the non real-time sensitive IPC activities identified in step S302."); *See id.*12:55-59. When the timer expires, the application processor sends the held data to the modem processor. *Id.*, 9:11-13 ("[W]hen one of the registered timers fires, all registered timers expire at the same time, causing all the aggregated IPC activities to be served at the same time."); 12:52-55 ("A scheduler may implement the same scheduling techniques as those described above, but configured to schedule IPC activities ***from the application processor 106 to the baseband processor 104***."); Lin-¶156 (emphasis added).

[5] This Petition sets forth reasons why a POSA would combine Heinrich and Balasubramanian in the section immediately below.

As the ’490 patent explains, an “uplink timer” is an “application timer” that, upon expiration, triggers the transmission of held data from the application processor to the modem processor. Ex-1301, 10:7-9 (“At the expiration of the application timer, the application processor 34 sends any held data to the modem processor 44 through the interconnectivity bus 36. . . .”); 10:11-12 (“the uplink timer (i.e., the application timer)”), 11:30-31 (“the uplink timer (i.e., the application timer)”). Thus, the lazy timer on the application processor in Heinrich is an uplink timer that, upon expiration, triggers the transmission of application data from the application processor to the baseband processor over the bus. Lin-¶99.

“whichever occurs first”: As described above, Heinrich in view of Balasubramanian teaches two trigger mechanisms that result in sending data from the application processor to the modem processor: (1) receipt of previously stored data from the modem processor; and (2) expiration of an uplink timer at the application processor.

A POSA would have understood that since both trigger mechanisms result in the transmission of held data from the application processor, whichever trigger occurs first causes transmission of the data over the bus. In fact, Heinrich itself acknowledges that when an action can be triggered by two different events, it will be triggered in response to the first to occur of those two events. Ex-1304, 9:22-26 (emphasis added) ("In general, each lazy timer is configured to fire in response to ***the earlier of***: (i) the expiry of a respective deadline provided to the lazy timer before which it is expected to fire, or (ii) a determination that the application processor 106 is in the awake mode."); Lin-¶93.

e) [16e] "wherein receipt of the data from the modem processor triggers passing the data received by the application processor to the modem processor over the interconnectivity bus before the interconnectivity bus transitions from an active power state to a low power state"

Heinrich in view of Balasubramanian discloses that "receipt of the data from the modem processor triggers passing the data received by the application processor to the modem processor over the interconnectivity bus before the interconnectivity bus transitions from an active power state to a low power state."

For the reasons described above, the combination of Heinrich and Balasubramanian discloses that "receipt of the data from the modem processor triggers passing the data received by the application processor to the modem processor over the interconnectivity bus ..." *See* claim [16d]. Element [16e] further requires that the transmission of data from the application processor to the modem processor must occur "before the interconnectivity bus transitions from an active power state to a low power state." Lin-¶102.

Heinrich teaches that one can use various techniques to determine when to send data from the application processor to the baseband processor. These techniques include a timer that triggers transmission of data (1) when the timer expires or (2) more generally, upon any other determination that informs the application processor that the baseband processor is in an "awake mode" and can therefore receive data. Ex-1304, 9:22-26 ("In general, each lazy timer is configured to fire in response to the earlier of: (i) the expiry of a respective deadline provided to the lazy timer before which it is expected to fire, or (ii) a determination that the [application] processor 106 is in the awake mode.").[6] More generally, Heinrich discloses waiting until the "best possible time to trigger" transmission of data from one processor to another. *Id.*, 7:65-8:6 (emphasis added) ("[T]he scheduler 120 identifies IPC activities that are not real-time sensitive and

[6] The quoted portion of Heinrich describes the transmission of data from the baseband processor. However, as explained, Heinrich discloses that the same techniques can be used to control transmission of data from the application processor to the baseband processor. Ex-1304, 12:52-55 ("A scheduler may implement the same scheduling techniques as those described above, but configured to schedule IPC activities from the application processor 106 to the baseband processor 104.").

which can be delayed until it is deemed profitable to run them. The identified IPC activities are aggregated by a software framework that allows registering non-urgent requests to communicate with the remote processor, (e.g. the application processor 106). In some embodiments, the scheduler 120 ***finds the best possible time to trigger*** all pending non real-time sensitive IPC activities.").

Balasubramanian discloses that one such best possible time to "trigger" transmission of data from a second processing node (*e.g.*, an application processor) to a first processing node (*e.g.*, a modem/baseband processor) is upon receipt of data from the first processing at the second processing node. Ex-1305 7:6-8 (emphasis added) ("For example, the network interface 112 [at a second processing node] may use the ***receipt*** of an uplink packet as a ***trigger*** to transmit any downlink packets in its queue."). Balasubramanian's teaching of transmitting data from a second processing node upon receipt of data from a first processing node means that the first processing node is awake and therefore able to receive data over the bus. Lin-¶105.

This transmission from the second processing node to the first processing node upon receiving data from the first processing node occurs during a "single wake state" of the first processing node (i.e., transceiver). Ex-1305 7:4-6 ("[D]uring the ***same single wake state*** the transceiver 110 also receives any downlink packets queued in the network interface 112."). In Balasubramanian, a "single wake state" refers to a period of time where the transceiver, and its associated bus, are in an active power state (such that data can be transmitted over the bus) and before the bus transitions back into a low power state. Lin-¶106.

This approach is consistent with Heinrich's goal of reducing the number of transitions between a processor (or bus) in a sleep or low power mode and an active or high power mode. *See* Ex-1304, 8:37-42, ("In this way, the non real-time sensitive IPC activities are aggregated and sent to the application processor 106 ***during one awake phase*** of the application processor 106. This ***reduces the number of times that the application processor 106 is woken up*** from its sleep mode for processing the IPC activities."); 8:50-55 ("It can therefore be seen that there is provided a setup . . . that optimizes system power consumption by scheduling non real-time sensitive IPC activities in a way that ***minimizes the number of times the remote processor is woken up*** from a sleep mode."). Lin-¶107.

While Heinrich describes the power conservation technique primarily for processors, Heinrich explains that the same power conservation functionality occurs in the bus connecting the baseband and application processors. In particular, Heinrich discloses that the bus over which the processors communicate can be a USB bus. *See* Ex-1304, 4:44-48; Lin-¶108.

Heinrich explains that—like the processors—the bus can transition from a sleep or low power mode to an active or high power mode. Specifically, each processor in Heinrich includes a "USB interface" that can transition from sleep to active mode to place the bus into sleep (*i.e.*, low power) or active (*i.e.*, high power) mode. Ex-1304, 3:37-40 ("For example, a Universal Serial Bus (USB) interface typically requires at least one second of idle time before switching to a USB suspend state (or "***sleep mode***")."); *id.*, 11:38-40 ("For some IPC interfaces, such as a USB interface, sending logging information every second would cause the USB interface to remain in an ***active mode***."). Heinrich explains that the USB interface—and, as a result, the USB bus—transitions back to a sleep mode after the application processor finishes its data transmission. *Id.,* 13:61-62 ("Time to set IPC (USB) interface to sleep mode after last IPC packet"); *id.*, 14:26-27 ("AP [application processor] inactivity timer fires; USB selective suspend state occurs [*i.e.*, sleep mode]; both AP and BB [baseband] enter low power mode"); Lin-¶109.

Obviousness

Combination of Heinrich and Balasubramanian: It would have been obvious for a POSA to substitute the "trigger" scheme from Balasubramanian into the processor-to-processor communication system of Heinrich in order to form the claimed combination. For example, as set forth above, Heinrich discloses the "modem processor," "downlink timer," "application processor," and "interconnectivity bus" limitations from claim 16. Balasubramanian teaches a very similar architecture as Heinrich—including a first processing node (transceiver 110), a timer (Ex-1305 12:3-10), a second processing node (network interface 112), and a wired communication link connecting the first and second processing nodes (an "interconnectivity bus"). Moreover, Balasubramanian teaches the "trigger" scheme of claim 16, as set forth above. Accordingly, substituting the "trigger" scheme from Balasubramanian into the processor-to-processor communication system of Heinrich results in the claimed combination.

Motivation to Combine

A POSA would have been motivated to combine Heinrich and Balasubramanian for at least four reasons.

First, Heinrich and Balasubramanian are in the same field (communications between processing nodes) and are concerned with the same issues—power consumption when transitioning from low power-to-high power states. Ex-1304 4:6-11 ("By grouping the non real-time sensitive IPC activities together and scheduling them for communicating to the second processor during a period in which the second processor is continuously in the first [active] mode, ***the number of times that the second processor enters and exits the second mode (e.g. sleep mode) is reduced***."); Ex-1305, 5:55-61 ("Here, ***power may be conserved*** by not transitioning the transceiver 110 from the suspended state to the active state every time a packet has been generated for transmission . . . Accordingly, power savings may be achieved by rescheduling the packet traffic into groups of traffic." (emphasis added)). Further, as set forth above, both Heinrich and Balasubramanian teach similar architectures that include two processing nodes coupled by a bus, buffering data on both sides of the bus to allow for power savings states, and the use of a timer in one or both processing nodes to determine when to send queued data to the other processing node. Therefore, it would be natural for a POSA to look to Balasubramanian when considering the power consumption issues of Heinrich. Lin-¶112.

Second, Heinrich and Balasubramanian both solve the power consumption problem in the same way: minimizing the number of transitions from low power-to-high power states. Thus, a POSA would have been motivated to use the techniques taught by Balasubramanian to address the same issues described in Heinrich. Lin-¶113.

Third, a POSA would realize that Heinrich is open to modification in that it discloses the detection of the state of the physical IPC interface in making decisions about when to transmit data. For instance, Heinrich notes that the scheduling techniques described with respect to its baseband processor are also applicable to the application processor (*see* Ex-1304 12:52-55), and therefore Heinrich teaches that its application processor can include a scheduler that can detect that the physical IPC interface is in an active state, and the baseband processor is awake. *Id.*, 9:50-62, 12:47-64. Because Heinrich's application processor can have a scheduler that can detect when the physical IPC interface is in an active state, it would have been obvious to a POSA that the application processor in Heinrich could detect that the IPC interface and baseband processor are in an active state due to the reception of data from the baseband processor, upon which the application processor would transmit its data to the baseband processor over the IPC interface. Balasubramanian teaches just that—its network interface 112 detects that the transceiver 110 is in an active state upon receiving data from the transceiver 110, which triggers the network interface 112 to transmit its held data to the transceiver 110 over the communication link 116. Balasubramanian also discloses that its teachings and techniques are not limited to a particular application. Ex-1305, 7:24-28. Lin-¶114.

Fourth, Heinrich recognizes that it is beneficial to transmit data from one processor—*e.g.*, an application processor—to a destination processor—*e.g.*, a baseband processor—when the destination processor is already awake/active. Ex-1304, 9:14-21 ("When one of the lazy timers fires, this [wakes] up the application processor 106. Therefore, this is a good time to schedule all the other pending, aggregated IPC activities to be sent to the application processor 106 because . . . the application processor 106 is in awake mode."); 10:32-37 ("As described above, if the scheduler 120 can determine whether the application processor 106 is in a sleep mode, IPC activities may be scheduled without waking the application processor 106 from sleep mode by scheduling the IPC activities for times when the application processor 106 is already awake.”). The same principle applies to the trigger scheme of Balasubramanian, which teaches that the network interface 112 transmits packets to the transceiver 110 upon receiving packets from the transceiver 110—*i.e.*, when the transceiver 110 and communication link 116 are already awake. Therefore, consistent with the goals of Heinrich, a POSA would have been motivated to piggyback the transmission from the application processor back to the baseband processor during the same “active” mode time period and right after data is sent from the baseband processor to the application processor, reducing the number of times the processor enters and exits the active mode. *See id.,* 4:6-11 (“By grouping the non real-time sensitive IPC activities together and scheduling them for communicating to the second processor during a period in which the second processor is continuously in the first [active] mode, the number of times that the second processor enters and exits the second mode (e.g. sleep mode) is reduced.”). Such a combination would have entailed merely incorporating a known technique (using the receipt of data from a processor as a way of determining that the other processor is awake) for a known purpose (to trigger a transmission to a processor upon receipt of data from the processor) with an expected result (fewer bus transitions between active mode and sleep mode). Lin-¶115.

Reasonable Expectation of Success: A POSA would have had a reasonable expectation of success in combining Balasubramanian's "trigger" scheme with the processor-to-processor communications system of Heinrich.

Heinrich discloses a system where an application processor can transmit held data to a baseband processor upon the occurrence of various triggering events. For example, Heinrich discloses a system where an application processor transmits held data to a baseband processor upon expiration of a timer. Ex-1304, 9:5-14 ("Each IPC activity is sent on the IPC interface when its respective lazy time fires. . . . However, when one of the registered timers fires, all registered timers expire at the same time, causing all aggregated IPC activities to be served at the same time."). As another example, Heinrich discloses a system where an application processor transmits held data to a baseband processor when the application processor determines that the baseband processor is in an "active" or "awake" mode. *Id.*, 9:22-26 ("In general, each lazy timer is configured to fire in response to the earlier of: (i) the expiry of a respective deadline provided to the lazy timer before which it is expected to fire, or (ii) a determination that the application processor 106 is in the awake mode."). Heinrich describes various techniques for determining that the other processor is awake, including: (1) receiving a notification that the bus has been placed in an awake mode (*Id.*, 9:54-58); (2) receiving a notification via a hardware connection that the other processor is active (*id.*, 10:1-6); and (3) receiving a notification via a shared memory that the other processor is active (*id.*, 10:24-21). Thus, Heinrich discloses various techniques that trigger the transmission of data from one processor to another processor. Lin-¶117.

A POSA would expect that one could successfully use the receipt of data from one processing node to trigger a return transmission of data from another processor node, as disclosed in Balasubramanian. Viewed from the perspective of the disclosure in Heinrich, Balasubramanian's technique is simply another way for one processor to determine that the other processor and bus are awake. A POSA would therefore understand that using Balasubramanian's trigger scheme would work as expected in the processor-to-processor communications system of Heinrich. Lin-¶118. In fact, Heinrich discloses that the "scheduler"—which determines when to transmit previously stored data from one processor to another—can detect when the baseband processor is active and trigger transmission of data from the application processor (or vice versa). Ex-1304, 10:1-5 ("In another example, the scheduler 120 can be notified when the remote application processor 106 is active. In this way the scheduler 120 performs the determination that the application processor 106 is in the awake mode by receiving a direct notification of such."); *see also id.*, 9:43-49 ("[T]he scheduler 120 can deduce that the application processor 106 will be in the awake mode by determining that real-time sensitive IPC activities are being sent to the application processor 106, and on that basis can schedule the non real-time sensitive IPC activities to be communicated to the application processor 106 to make use of the awake mode of the application processor 106."). Heinrich discloses that this notification "can be achieved through various means." *Id.*, 10:5-6. A POSA would recognize that one such "means" is the trigger scheme described in Balasubramanian. Lin-¶118.

Thus, Heinrich in combination with Balasubramanian discloses this limitation.

2. Claim 20

Claim 20 depends from claim 16, which as described in Section 0 is rendered obvious by Heinrich in view of Balasubramanian. Heinrich in combination with Balasubramanian renders obvious the limitation of claim 20, which recites "providing an override capability based on one of accumulated packet size, accumulated packet count, accumulated byte count, quality of service requirement, and control message status." Lin-¶120.

As previously described for claim [16b], Heinrich teaches sending accumulated data from the baseband processor to the application processor upon expiration of a timer. *See* Section 1.b). Heinrich arguably does not explicitly disclose a mechanism to override the timer. Balasubramanian renders obvious the claimed "override" feature. Balasubramanian teaches a queue that holds packets until a "timer/counter 526" indicates that a configurable amount of time or configurable number of packets is reached. Ex-1305, 6:55-65 ("[O]nce the configurable amount of time has elapsed or the configurable number of packets have been queued, the transceiver 110 transitions to an active (e.g., wake) state. . . [T]he transceiver 110 then transmits the queued uplink packets over the communication link 116."); 9:7-9 ("[P]ackets may be queued for a configurable amount [of] time or a configurable number of packets may be queued at a time.").

A POSA would have understood that Balasubramanian teaches that one or both of the timer or packet counter could have been implemented in the same system. *See* Lin-¶122. Doing so would have been combining known techniques taught by Balasubramanian (*i.e.*, the timer and packet counter) to yield the predictable result of clearing a buffer when a threshold condition is detected. A POSA would have further understood that if the configurable amount of packets criterion was reached before the configurable amount of time was reached, the accumulated packet count would override the timer. *See id.* . Thus, Balasubramanian renders obvious an "override capability based on … accumulated packet count."

In addition to the reasons provided for claim [16e], a POSA would have been motivated and had a reasonable expectation of success to make such a combination. *See* Section IX.A.1.e). Heinrich recognizes the risk of buffer overflow. For example, Heinrich explains that a logging buffer size should be larger than an amount of data accumulated before the expiration of a timer. Ex-1X04, 14:40-43 ("Instead of sending logging data out, the BB logger accumulates data in memory and starts an infinite lazy timer. ***The logging buffer is big enough to accommodate more than 2 minutes of logging in idle mode***." (emphasis added)). A POSA would have been motivated to combine the timer/counter of Balasubramanian with the system of Heinrich because using both a counter and a timer would enable use of a smaller buffer, which has a greater risk of overflowing. Lin-¶123. Including a packet counter into Heinrich's system would have provided an additional guard against overfilling the buffer and would have been obvious to try given the limited number of ways to track an amount of data that is buffered—*i.e.*, by time or size of the data. *Id.* . A POSA would have had a reasonable expectation of success from this combination because sending data based on a timer or buffer fill level serves the same purpose of flushing data before the buffer fills.

3. Claim 28

a) [28a] Preamble: "[a] mobile terminal"

The '490 patent describes the claimed "mobile terminal" as a smart phone, cell phone, tablet, or similar device. Ex-1301, 6:37-41 ("The mobile terminal 22 may be a smart phone, . . . a cellular telephone, a tablet, a laptop, or other mobile computing device.").

Heinrich teaches this limitation by disclosing a "user device" that can be "a mobile phone, a tablet, a laptop computer or other embedded device able to connect to the network 110." Ex-1304, 4:21-23; *see also id.,* Fig. 1 (showing user device 102).

b) [28b] "a modem processor"

For the same reasons discussed above for claim [16b], Heinrich discloses the "modem processor" of claim 28. *See supra* Section IX.A.1.b); Lin-¶126.

c) [28c] "an application byte counter"

As discussed above for claim [16d], Heinrich teaches an application timer but arguably does not explicitly disclose the "application byte counter" of claim 28. *See supra* Section IX.A.1.d); Lin-¶127. Heinrich in view of Balasubramanian renders obvious this limitation.

As discussed above for claim 20, Balasubramanian discloses a "counter" that tracks a number of packets to queue before sending the packets. *See supra* Section

IX.A.2. It would have been obvious to modify Heinrich's system in which data is sent at the expiration of a timer with Balasubramanian's teaching of sending data if packet number limit is exceeded. Lin-¶128. As explained above, Heinrich recognizes that there is a risk of buffer overflow if the buffer does not flush its data prior to expiration of the lazy timer. *See supra* Section IX.A.2. Including a packet counter into Heinrich's system would have provided an alternative or additional guard against overfilling the buffer, would have allowed the use of a smaller buffer, and would have been obvious to try given the limited number of ways to track an amount of data that is buffered—*i.e.*, by time or size of the data. *Id.* . A POSA would have had a reasonable expectation of success from this combination because flushing a data buffer before it fills can be accomplished by sending data based either on a timer or buffer fill level. *Id.*

For the same reasons set forth above, a POSA would have found it obvious for the application processor to send data to the modem processor if a threshold associated with an application byte counter is exceeded. Including a byte counter, rather than a packet counter, into Heinrich's system would have provided an alternative or additional guard against overfilling the buffer in a system supporting variable packet sizes. In such systems, tracking a number of packets would not provide meaningful insight into the risk of buffer overflow, because a specific number of variable sized packets can occupy a wide range of memory sizes.

Accordingly, a POSA would have been motivated in such systems to track the buffer fill level by monitoring the number of bytes rather than the number of packets. Lin-¶129.

A POSA would have further understood that a buffer size could have been expressed in bytes. Expressing a buffer size in bytes would have been obvious to try as there were only a finite number of ways to express the size of data (e.g., bits, bytes) and Heinrich expressly recognized that data could be expressed in megabytes. Ex-1304, 13:55-59 ("The following data were observed on a customer mobile phone design for a scenario in which, over a 36 hour period there is 4 hours of talk time, ***500 MB*** of download data and 100 texts, with the rest of the time being idle" (emphasis added)); Lin-¶130.

d) [28d] "an application processor,"

For the reasons discussed above for claim [16c], Heinrich discloses the "application processor" of claim 28. *See supra* Section IX.A.1.c*)*; Lin-¶131.

e) [28e] "the application processor configured to hold application processor to modem processor data until expiration of the application timer"

For the reasons discussed above for claim [16d], Heinrich discloses the holding of "application processor to modem processor data [i.e., the "application data" of claim 16] until expiration of the application timer" of claim 28. *See supra* Section IX.A.1.d); Lin-¶132.

It would have been obvious to a POSA, for the same reasons discussed above for claim [16d], that the application processor in Heinrich could hold data — *i.e.*, the "application processor to modem processor data"—in its own memory on the same chip as the processor circuitry. *See supra* Section IX.A.1.d); Lin-¶150.

f) [28f] "an interconnectivity bus communicatively coupling the application processor to the modem processor"

For the same reasons discussed above for claim [16c], Heinrich discloses "an interconnectivity bus communicatively coupling the application processor to the modem processor" of claim 28. *See supra* Section IX.A.1.c); Lin-¶134.

g) [28g] "the modem processor configured to hold modem processor to application processor data until triggered by receipt of the application processor to modem processor data from the application processor through the interconnectivity bus after which the modem processor to application processor data is sent to the application processor through the interconnectivity bus responsive to the receipt of the application processor to modem processor data from the application processor through the interconnectivity bus"

The combination of Heinrich and Balasubramanian discloses "the modem processor configured to hold modem processor to application processor data until triggered by receipt of the application processor to modem processor data from the application processor through the interconnectivity bus after which the modem processor to application processor data is sent to the application processor through the interconnectivity bus responsive to the receipt of the application processor to modem processor data from the application processor through the interconnectivity bus." Lin-¶135.

"the modem processor configured to hold modem processor to application processor data": For the reasons described above for claim [16b], Heinrich discloses holding the "modem processor to application processor data [i.e., the "data received by a modem processor from a remote network" of claim 16]" of claim 28. *See supra* Sections IX.A.1.b); Lin-¶136.

It would have been obvious to a POSA, for the same reasons discussed above for claim [16b], that the modem processor in Heinrich could hold data —*i.e.*, the "modem processor to application data"—in its own memory on the same chip as the processor circuitry. *See supra* Section IX.A.1.b); Lin-¶137.

"until triggered by receipt of the application processor to modem processor data from the application processor through the interconnectivity bus": As explained above, Heinrich teaches that the modem processor can be triggered to send its data to the application processor upon determining that the application processor is in the awake mode. Ex-1304, 10:32-37 ("As described above, if the scheduler 120 can determine whether the application processor 106 is in a sleep mode, IPC activities may be scheduled without waking the application processor 106 from sleep mode by scheduling the IPC activities for times when the application processor 106 is already awake.”); *see also id.*, 9:43-49. Heinrich also discloses transmitting data from the application processor to the modem processor through an interconnectivity bus. *Id.*, Fig. 1, 4:44-50 (“There is a ***physical interface*** configured for communicating IPC activities between the baseband processor 104 and the application processor 106. The physical interface may, for example, be . . . a Universal Serial Bus (USB) interface. . . .”). Heinrich does not explicitly teach determining that the application processor is awake based on the receipt of data from the application processor. However, Balasubramanian discloses holding, at a second processing node, data to be sent to a first processing node until “triggered” by receipt of data transmitted by the first processing node to the second processing node through a communication link. Lin-¶138..

In particular, Balasubramanian discloses that receipt of data from a first processing node (*e.g.*, the transceiver 100) at a second processing node (*e.g.*, the network interface 112) “trigger[s]” transmission of held data from the second processing node back to the first processing node:

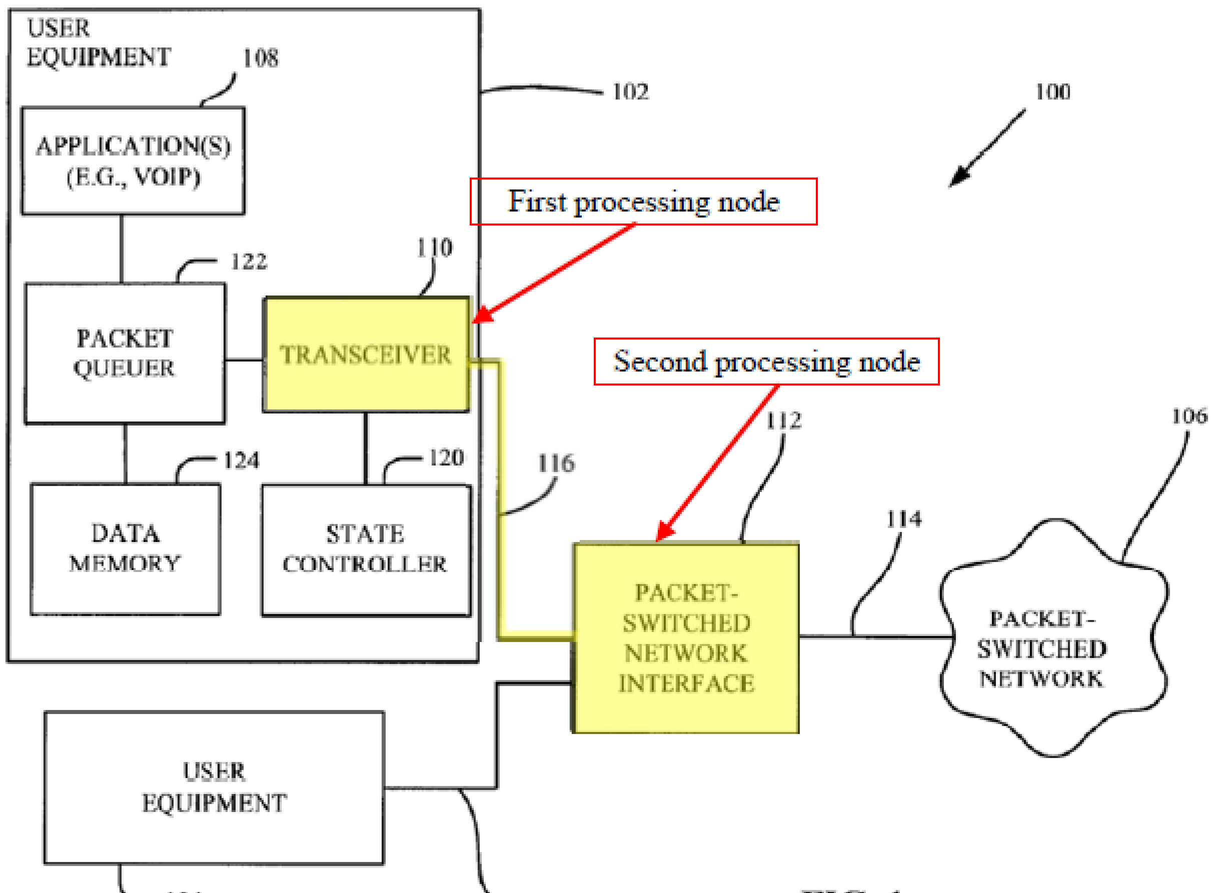

FIG. 1

Ex-1305, Fig. 1 (annotations added); *id.*, 7:4-8 ("As represented by block 210, during the same single wake state the transceiver 110 also receives any downlink packets [*i.e.*, data from the second processing node] queued in the network interface 112. For example, the network interface 112 may use the ***receipt*** of an uplink packet [*i.e.*, data from the first processing node] as a ***trigger*** to transmit any downlink packets in its queue [*i.e.*, data from the second processing node].).” The trigger mechanism of Balasubramanian can be used with the two-processor communication system of Heinrich for the same reasons described for claim 16. *See also* Section IX.A.1 *supra*. Lin-¶139.

As explained for claim [16d] above, Balasubramanian discloses that the "trigger" for the transmission of data from the second processing node (network interface 112) to the first processing node (transceiver 110) is the reception at the second processing node of the queued packets from the first processing node. Thus, the combination of Heinrich and Balasubramanian teaches the claim limitation "until triggered by receipt of the application processor to modem processor data from the application processor through the interconnectivity bus."

"after which the modem processor to application processor data is sent to the application processor through the interconnectivity bus responsive to the receipt of the application processor to modem processor data from the application processor through the interconnectivity bus": The combination of Heinrich and Balasubramanian discloses that "after which the modem processor to application processor data is sent to the application processor through the interconnectivity bus responsive to the receipt of the application processor to modem processor data from the application processor through the interconnectivity bus." As explained above, the combination of Heinrich and Balasubramanian discloses transmitting held data from the modem processor to the application processor upon receiving data from the application processor. *See* Section IX.A.1c) *supra*. Lin-¶141.

Heinrich discloses transmitting data from the modem processor to the application processor through an interconnectivity bus. Ex-1304, Fig. 1, 4:44-50 ("There is a ***physical interface*** configured for communicating IPC activities between the baseband processor 104 and the application processor 106. The physical interface may, for example, be . . . a Universal Serial Bus (USB) interface. . . ."); 7:19-23 ("The scheduler 120 may control the scheduling of IPC activities in both directions between the processors 104 and 106. Alternatively, the scheduler 120 may control the scheduling only of IPC activities communicated from the baseband processor 104 to the application processor 106. . . ."). As explained above, Balasubramanian discloses that receipt of data from a first processing node (*e.g.*, the transceiver 110) at a second processing node (*e.g.*, the network interface 112) can trigger the transmission of data from the second processing node back to the first processing node over the bus. Ex-1305, 7:6-8 ("For example, the network interface 112 may use the ***receipt*** of an uplink packet as a ***trigger*** to transmit any downlink packets in its queue."). The trigger mechanism of Balasubramanian can be used in Heinrich to trigger the modem processor to transmit data to the application processor over the bus. *See also* Section IX.A.1e) *supra*. Lin-¶142.

Thus, the combination of Heinrich and Balasubramanian disclose transmitting previously stored data from a modem processor to an application processor in response to—*i.e.*, triggered by—receipt of data from the application processor at the modem processor.

A POSA would have been motivated to combine Heinrich and Balasubramanian (and would have had a reasonable expectation that the combination would work for its intended purpose) for the same reasons described above for claim 16. *See* Section IX.A.1e) *supra*. Lin-¶144.

4. Claim 30

a) [30a] Preamble: "[a] method"

For the same reasons discussed above for claim [16a], Heinrich discloses the "method" of claim 30. *See supra* Section IX.A.1.a)IX.A.1.b); Lin-¶145.

b) [30b] "starting an application timer at an application processor"

For the same reasons discussed above for the "uplink timer" of claim [16d], Heinrich discloses the "application timer" of claim 30. *See supra* Section IX.A.1.d); Lin-¶146.[7]

[7] The '490 patent uses "application timer" and "uplink timer" synonymously. *See* Ex-1301, 11:30-34 ("uplink timer (*i.e.*, the application timer)").

Heinrich further discloses that the "application timer [is] at an application processor." In particular, Heinrich teaches a "scheduler" on the application processor. Ex-1304,7:24-27 ("[A] separate scheduler (e.g. ***implemented on the application processor***) controls the scheduling of IPC activities communicated from the application processor 106 to the baseband processor 104."). As explained above in limitation [16d], Heinrich teaches that the scheduler on the application processor uses application timers (Heinrich's "lazy timers") to control when data is transmitted from the application processor to the baseband processor. Lin-¶147.

Heinrich also discloses the "starting" of the application timer. Heinrich teaches an application programming interface (API) for "start[ing]" and "stop[ping]" an already started timer. Ex-1304, 11:1-22 ("The lazy timer API has the following functions: … Ipc_TimerStart(handle, handler, argument, timeout) This calls the handler with the specified argument after a maximum of timeout seconds. Ipc_TimerStop(handle) This stops a previously started timer…. The timeout provided to Ipc_TimerStart function is expressed in seconds, which is a voluntarily coarse unit."); Lin-¶148.

c) [30c] "accumulating data at the application processor until expiration of the application timer"

For the reasons discussed above for claim [16d], Heinrich discloses the accumulating of "data at the application processor [*i.e.*, the "application data" of claim 16] until expiration of the application timer" of claim 30. *See supra* Section IX.A.1.d); Lin-¶149.

It would have been obvious to a POSA, for the same reasons discussed above for claim [16d], that the application processor in Heinrich could accumulate data in its own memory on the same chip as the processor circuitry. *See supra* Section IX.A.1.d); Lin-¶150.

d) [30d] "sending the accumulated data from the application processor to a modem processor across an interconnectivity bus"

For the same reasons discussed above for claims [16d] and [16e], which recite in part "holding application data generated by an application associated with the application processor" and "passing the data received by the application processor to the modem processor over the interconnectivity bus," respectively, Heinrich discloses the "sending the accumulated data from the application processor to a modem processor across an interconnectivity bus" of claim 30. *See supra* Section IX.A.1.d), IX.A.1.e); Lin-¶151.

e) [30e] "holding modem processor data at the modem processor until triggered by receipt of the accumulated data from the application processor"

For the reasons discussed above for claim [28g], Heinrich and Balasubramanian render obvious the "holding modem processor data at the modem processor until triggered by receipt of the accumulated data from the application processor" of claim 30. *See supra* Section IX.A.1.d); Lin-¶152.

It would have been obvious to a POSA, for the same reasons discussed above for claim [16b], that the modem processor in Heinrich could hold data —*i.e.*, the "modem processor to application data"—in its own memory on the same chip as the processor circuitry. *See supra* Section IX.A.1.b); Lin-¶153.

f) [30f] "wherein receipt of the accumulated data from the application processor triggers passing the modem processor data to the application processor over the interconnectivity bus before the interconnectivity bus transitions from an active power state to a low power state"

For the same reasons discussed above for claims [16e] and [28g], Heinrich and Balasubramanian render obvious "wherein receipt of the accumulated data from the application processor triggers passing the modem processor data to the application processor over the interconnectivity bus before the interconnectivity bus transitions from an active power state to a low power state" of claim 30. *See supra* Section IX.A.1.e), IX.A.3.g); Lin-¶154.

Additionally, Heinrich teaches that the scheduling techniques described in Heinrich are applicable for data transfers in both directions between the modem processor and application processor. As explained above for claim [16e], Heinrich teaches, in a scenario where the application processor of Heinrich is the remote processor, that a bus can transition from an active mode back to a sleep mode after the application processor finishes its data transmission to the modem processor. Ex-1304*,* 13:61-62; *see also id.*, 14:26-27. Heinrich also teaches that the same scheduling techniques can apply to data transfers in the reverse direction—*i.e.*, where the modem processor of Heinrich is the remote processor and a bus can transition from an active mode back to a sleep mode after the modem processor finishes its data transmission to the application processor. *See id.*; *see also id.*, 12:47-64 ("In the examples described in detail above, the scheduler 120 schedules the IPC activities for communicating from the baseband processor 104 to the application processor 106. In that case, the application processor 106 is the remote processor for the IPC and the baseband processor 104 is the local processor for the IPC. A scheduler may implement the same scheduling techniques as those described above, but configured to schedule IPC activities from the application processor 106 to the baseband processor 104."); Lin-¶155.

X. CONCLUSION

Based on the foregoing, claims 20, 28, and 30 of the '490 patent recite subject matter that is obvious. The Petitioner requests institution of an *inter partes* review to cancel those claims.

Respectfully submitted,

Intel, Corp.,

Petitioner

By: [illegible]
Jason D. Kipnis
Registration No. 40,680
Wilmer Cutler Pickering
Hale and Dorr LLP

CERTIFICATE OF SERVICE

I hereby certify that on July 6, 2018, I caused a true and correct copy of the following materials:

- Petition for *Inter Partes Review* of U.S. Patent No. 9,535,490 Under 35 U.S.C. § 312 and 37 C.F.R. § 42.104
- Exhibits 1301-1317
- List of Exhibits for Petition for *Inter Partes* Review of U.S. Patent No. 9,535,490 (Exhibits 1301-1317)
- Power of Attorney
- Fee Authorization
- Certificate of Compliance

to be served via Federal Express on the following attorney of record as listed on PAIR:

W&T/Qualcomm
106 Pinedale Springs Way
Cary, NC 27511

[illegible]

Jason D. Kipnis
Registration No. 40,680

CERTIFICATE OF COMPLIANCE

I hereby certify that the foregoing, Petition for *Inter Partes Review* of U.S. Patent No. 9,535,490, contains 13,867 words as measured by the word processing software used to prepare the document, in compliance with 37 C.F.R. § 42.24 (d).

Respectfully submitted,

Dated: July 6, 2018

[illegible]

Jason D. Kipnis
Registration No.40,680

LIST OF EXHIBITS FOR PETITION FOR *INTER PARTES* REVIEW OF U.S. PATENT NO. 9,535,490

Exhibit	**Description**
1301	U.S. Patent No. 9,535,490
1302	Declaration of Dr. Bill Lin
1303	File History for U.S. Patent No. 9,535,490
1304	U.S. Patent No. 9,329,671 to Heinrich et al. ("Heinrich")
1305	U.S. Patent No. 8,160,000 to Balasubramanian ("Balasubramanian")
1306	PCT Publication No. WO 2009/039034 to Tsai ("the Intel PCT")
1307	U.S. Patent No. 6,021,264 to Morita ("Morita")
1308	Kaplan, Wiley Electrical and Electronics Engineering Dictionary, IEEE Press, John Wiley & Sons, 2004 ("Kaplan")
1309	Downing, et al, Dictionary of Computer and Internet Terms, Barron's Educational Series, Inc., 2013 ("Downing")
1310	Duan et al., "Push vs. Pull: Implications of Protocol Design on Controlling Unwanted Traffic," SRUTI. July 7, 2005 ("Duan")
1311	Panda et al., "On-Chip vs. Off-Chip Memory: The Data Partitioning Problem in Embedded Processor-Based Systems," ACM Transactions on Design Automation of Electronic Systems, Vol. 5, No. 3, July 2000, ("Panda")
1312	Houghton Mifflin Co., The American Heritage College Dictionary, 2000 ("Heritage Dictionary")
1313	Soanes et al., Concise Oxford English Dictionary, 11th Ed., Oxford University Press, 2004 ("Oxford Dictionary")
1314	Merriam-Webster, Inc., Merriam-Webster's Collegiate Dictionary, 11th Ed., 2000 ("Merriam-Webster's Dictionary")
1315	Gerber, et al., "P2P the gorilla in the cable," National Cable & Telecommunications Association (NCTA) 2003 National Show, June 2000 ("Gerber")

Exhibit	**Description**
1316	Kolbehdari, et al., "The Emergence of PCI Express in the Next Generation of Mobile Platforms," Intel Technology Journal, Vol. 9, No. 1, 2005 ("Kolbehdari")
1317	U.S. Patent No. 8,112,646 to Tsai ("Tsai")