DOCKET NO.: 0107131.00562US1
Filed on behalf of Intel Corp.
By: Jason Kipnis, Reg. No. 40,680
David L. 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 IPR2018-01261

PETITION FOR *INTER PARTES* REVIEW OF U.S. PATENT NO. 9,535,490 CHALLENGING CLAIM 31

TABLE OF CONTENTS

I. INTRODUCTION 1

II. MANDATORY NOTICES 1

A. Real Party-in-Interest 1

B. Related Matters 1

C. Counsel 2

D. Service Information 2

III. CERTIFICATION OF GROUNDS FOR STANDING 3

IV. OVERVIEW OF CHALLENGE AND RELIEF REQUESTED 3

A. Prior Art Patents and Printed Publications 3

B. Grounds for Challenge 3

V. TECHNOLOGY BACKGROUND 4

A. Processor-To-Processor Communications 4

B. Communication Bus Power-Saving States 6

C. Data Buffering 7

VI. THE '490 PATENT 8

A. Alleged Problem of the Prior Art 9

B. Purported Solution of the '490 Patent 10

C. Prosecution History of the '490 Patent 16

VII. CLAIM CONSTRUCTION 19

A. "pull" 20

VIII. OVERVIEW OF PRINCIPAL PRIOR ART REFERENCES 22

A. U.S. Patent No. 9,329,671 To Heinrich et al. 22

B. U.S. Patent No. 8,160,000 to Balasubramanian 25

IX. SPECIFIC GROUNDS FOR PETITION 30

A. Ground 1: Claim 31 is Rendered Obvious By Heinrich In View Of Balasubramanian 30

1. [31a] Preamble: "[a] mobile terminal" 30

2. [31b] "a modem timer" 30

3. [31c] "a modem processor" 31

4. [31d] "the modem processor configured to hold modem processor to application processor data until expiration of the modem timer" 32

5. [31e] "an application processor" 37

6. [31f] "an interconnectivity bus communicatively coupling the application processor to the modem processor" 38

7. [31g] "the application processor configured to hold application processor to modem processor data until the modem processor pulls data from the application processor after transmission of the modem processor to application processor data" 39

8. [31h] "wherein the modem processor is further configured pull data from the application processor after transmission of the modem processor to application processor data and before the

interconnectivity bus transitions from an active power state to a low power state." 51

X. CONCLUSION 56

I. INTRODUCTION

Intel Corporation ("Petitioner") respectfully submits this Petition for *Inter Partes* Review ("Petition") of claim 31 of U.S. Patent No. 9,535,490 (the '490 patent") (Ex-1001). The '490 patent discloses a device comprising two processors that communicate over a bus using a push-pull protocol. *See* Ex-1001, claim 31. 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, claim 31 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 a certain claim of the '490 patent. 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 1-6 and 8 of the '490 patent (IPR2018-01293) 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 claim 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 claim 31 of the '490 patent (Ex-1001).

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-1004), 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-1005), 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 claim 31 as unpatentable under 35 U.S.C. § 103. This Petition, which is supported by the Declaration of Dr. Bill Lin ("Lin") (Ex-1002), 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 Claim**
1.	§ 103	Combination of Heinrich and Balasubramanian	31

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:

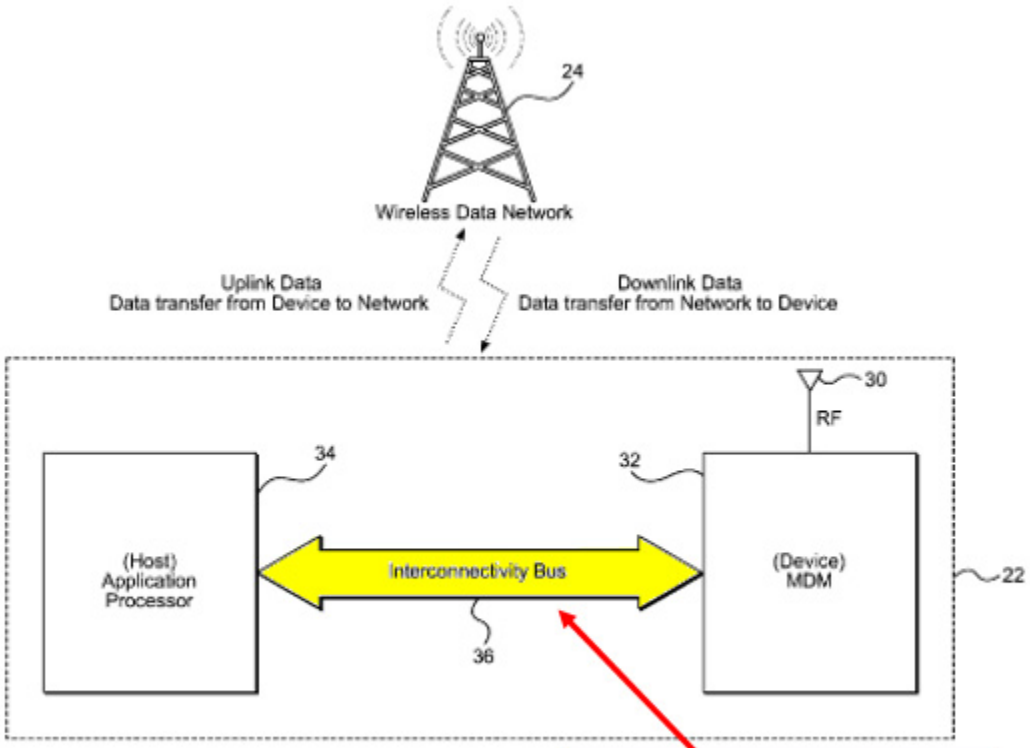

FIG. 1C

Ex-1001, 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-1001, 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-1001, 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-1005, 5:47-51; 6:40-43; 9:4-7; Ex-1004, 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-1004, 9:22-24; Ex-1005,1:66-2:2; 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.*, *id.*, 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-1001) 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.

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-1001, 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: (1) transitioning the bus from a low

[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.

power state to an active state to transmit downlink data, and (2) separately performing the same bus transition again at a later time to transmit uplink data—as shown in Figure 3 below:

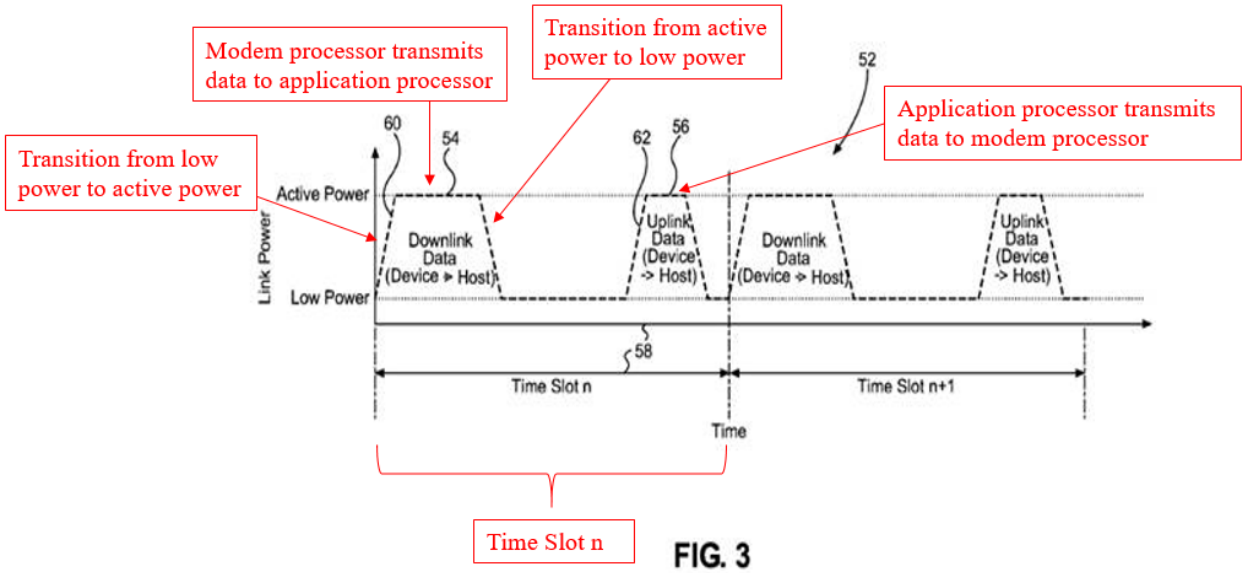

FIG. 3

Ex-1001, Fig. 3 (annotations added); *id.*, 8:35-40 (stating that "two transitions (i.e., 60, 62) from low power to active power [] every 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:

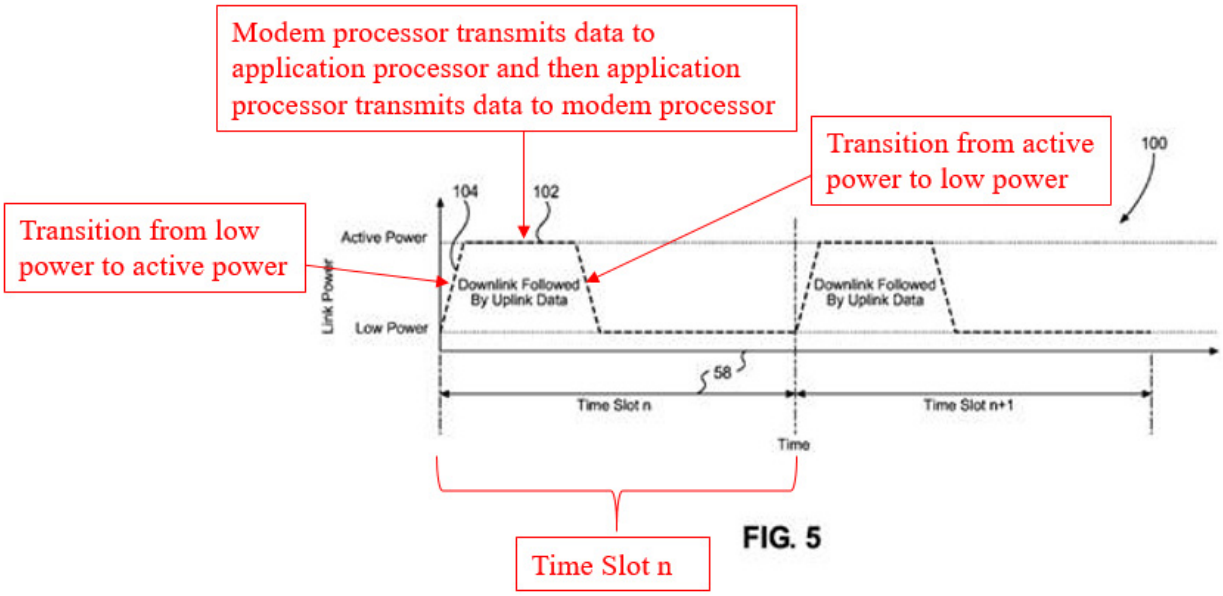

FIG. 5

Ex-1001, Fig. 5 (annotations added), 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 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:

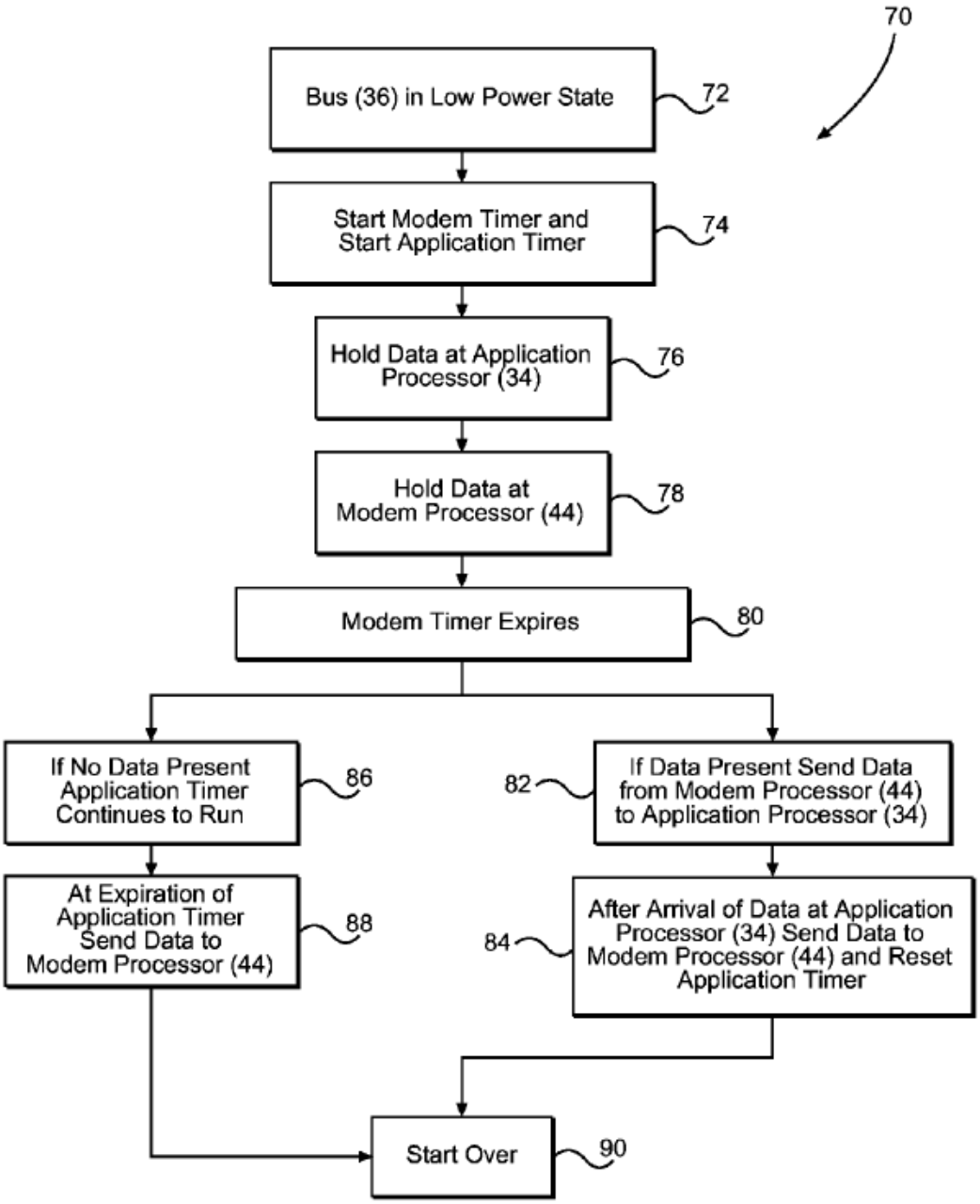

FIG. 4

Ex-1001, 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 (step 76), and the modem data is held at the modem processor 44 (step 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 step 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 step 84). *Id.*[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

The specification also discloses an alternative embodiment for how to transmit buffered downlink and uplink data during the same active state. Unlike the Figure 4 scheme where both processors "push" their data to the other processor (*i.e.*, each processor sends its data at its own initiative), in this embodiment, the modem processor (1) first pushes its data by sending the buffered downlink data to the application processor, and (2) then pulls the application processor's buffered uplink data by receiving that data in response to a request for it. *Id.*, 4:38-42 ("The application processor is configured to hold application processor to modem processor data until the modem processor pulls data from the application processor after transmission of the modem processor to application processor data.").

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-1005, 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***

---

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

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. 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 were the specific schemes disclosed in the '490 patent for how to transmit all buffered downlink and uplink data during the same active state: namely, (1) using the receipt of buffered data from a first processing node as a "trigger" for a second processing node to send its buffered data, or (2) having the first processing node send all its buffered data to a second processing node and then pull any buffered data from the second processing node. For example, nearly a decade before the '490 patent was filed, Balasubramanian disclosed a system in which receipt of data from a first processing node "trigger[ed]" the transmission of data from a second processing node, and in which the first processing node could push its data to a second processing node and then immediately pull any data from the second processing node. Ex-1005, 7:6-11 ("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. Alternatively, the transceiver 110 ***may send a message to the network interface 112 requesting transmission of all queued packets***.").

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-1003 [Preliminary Amendment], 46.

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-1006) and U.S. Patent No. 6,021,264 ("Morita") (Ex-1007). *See* Ex-1003 [June 10, 2016 Office Action], 3-9. In response to this rejection, the Applicant amended claim 1 to include the "trigger" limitation as shown below:

1. A mobile terminal comprising:

a modem timer;

a modem processor, the modem processor configured to hold modem processor to application processor data until expiration of the modem timer;

an application processor;

an interconnectivity bus communicatively coupling the application processor to the modem processor; and

the application processor configured to hold application processor to modem processor data until triggered by receipt of the modem processor to application processor data from the modem processor through the interconnectivity bus after which the application processor to modem processor data is sent to the modem processor through the interconnectivity bus responsive to the receipt of the modem processor to application processor data from the modem processor through the interconnectivity bus.

Ex-1003 [August 24, 2016 Response to Office Action], 2. The Applicant similarly amended original independent claims 15, 20, and 24-28 to include a "trigger" limitation. *Id.*, 14 ("Independent 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. . . ."). Original independent claim 29, which issued as claim 31, included a "pull" limitation that the Applicant further amended:

29. A mobile terminal comprising:

a modem timer;

a modem processor, the modem processor configured to hold modem processor to application processor data until expiration of the modem timer;

an application processor;

an interconnectivity bus communicatively coupling the application processor to the modem processor; and

the application processor configured to hold application processor to modem processor data until the modem processor pulls data from the application processor after transmission of the modem processor to application processor data,

wherein the modem processor is further configured pull data from the application processor after transmission of the modem processor to application processor data and before the interconnectivity bus transitions from an active power state to a low power state.

Id., 8-9.

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*., 14. 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 construction of a term of claim 31 of the '490 patent and its support for the construction. 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 BRI to the 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.

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

A person of ordinary skill in the art 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. "pull"

As used in the '490 patent, a person of ordinary skill would have understood the term "pull" to mean receiving data in response to a request for the data. For example, claim 31 refers to "pull[ing] data from the application processor," which refers to the modem processor receiving data from the application processor. The '490 specification is consistent with this meaning of the term "pull." For instance, the '490 specification explains that data can be "pulled or pushed" across a bus, indicating that the two are different. *See* Ex-1001, 9:61-10:1 ("After arrival of the modem data at the application processor 34, the application processor 34 ***releases any application data*** that has been held at the application processor 34 and resets the application timer (block 84).… ***As an alternative***, the modem processor 44 may continue to ***pull the uplink data*** 56 from the application processor 34 …."); 16:34-37 ("Likewise, once a timer has expired, data can be ***pulled or pushed*** across the interconnectivity bus 36 based on polling, setting doorbell registers, or

other technique."). These quotations further demonstrate that there are a variety of ways to perform a "pull" or "push" (*i.e.*, "polling, setting doorbell registers, or other techniques"). These quotations also make clear that one distinction between a push and pull data transfer is the event that triggers the transfer of data. In a pull data transfer, the data is transferred in response to a request for data by the intended ***recipient*** of the data, whereas in a push data transfer, data is transferred when the ***sender*** determines it is time to transfer. Lin-¶68. *See also, e.g.*, Kaplan, *Wiley Electrical and Electronics Engineering Dictionary*, IEEE Press, John Wiley & Sons, 2004 ("**pull technology**[:] Data distribution, such as that over the Internet, in which users receive information by requesting it.") (Ex-1008); Downing, et al., *Dictionary of Computer and Internet Terms*, Barron's Educational Series, Inc., 11th ed. 2013 ("**pull**[:] the process whereby the user retrieves information from a network at the user's request, as in traditional web browsing . . . .") (Ex-1009); and Duan et al., *Push vs. Pull: Implications of Protocol Design on Controlling Unwanted Traffic*, SRUTI, July 7, 2005, §2.2 ("In the receiver-pull model, it is the receiver who initiates the message transfer by explicitly contacting the sender. The sender passively waits for the receiver and delivers the entire content upon receiving a request.") (Ex-1010).

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.* (Ex-1004) was filed January 29, 2013 and issued May 3, 2016. Heinrich is therefore prior art at least under 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:

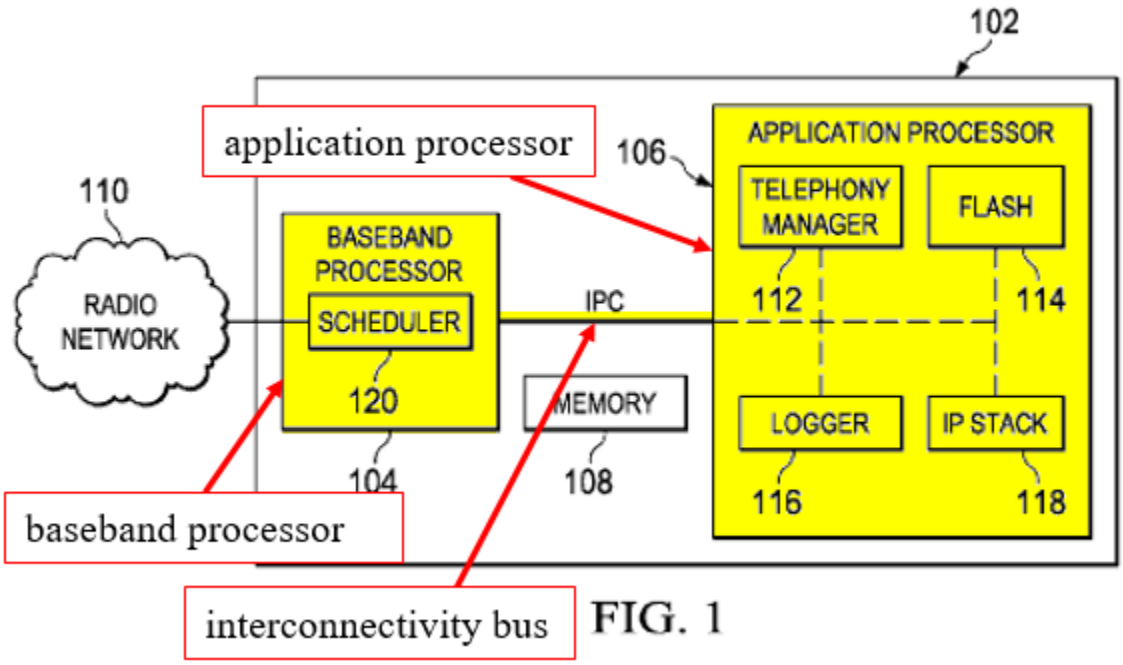

FIG. 1

Ex-1004, Fig. 1, 4:26-46 ("FIG. 1 shows the computer system of the user device 102 including a first sub-system implementing a ***baseband processor 104*** and a second sub-system implementing an ***Application Processor (AP) 106***. . . . 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 one of . . . a Universal Serial Bus (USB) interface. . . .").

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 ("***[W]hen 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. 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.").

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

U.S. Patent No. 8,160,000 to Balasubramanian (Ex-1005) 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:

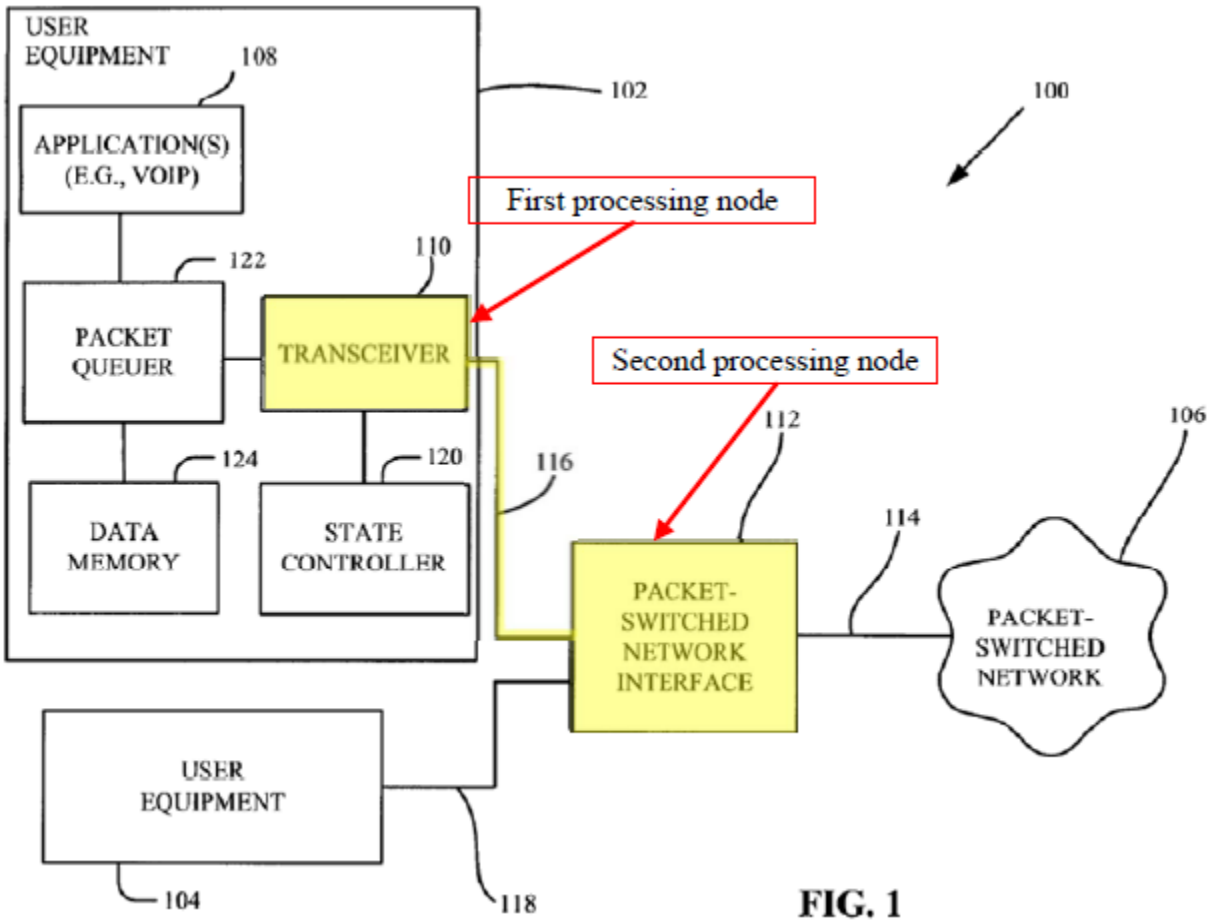

FIG. 1

Ex-1005, Fig. 1.

Balasubramanian discloses two schemes to synchronize transfers over the communication link 116. In a first embodiment, 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."); 2:23-24 ("Advantageously, these uplink and downlink packets may be transmitted during a single wake state of the apparatus.").

In an alternative embodiment, Balasubramanian discloses a scheme in which the transceiver 110 sends its buffered packets to the network interface 112 and then ***pulls*** buffered packets from the network interface 112 during the same awake state. *Id.*, 6:63-7:11 ("[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***…. [T]he transceiver 110 may send a ***message to the network interface 112 requesting transmission of all queued packets***."); *see also id.*, 6:15-19.

Balasubramanian discloses two processing nodes communicating over a wired link, like the bus in Heinrich. *Id.*, 4:50-54 ("[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-¶62.

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 suspended state into an active state reduces power consumption. Ex-1005, 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 by the user equipment 102 or every time it is expected that the user equipment 102 will receive a packet. 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 ("For example, the 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)."); 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.*, 9:33-35 ("[T]he control module 430 may use an output of the ***timer***/counter 426 to determine when to make the packets in the packet queue 422 available. . . ."); 6:46-49 ("As is discussed in more detail below, the packets may be queued for configurable amount of time."); 6:55-60 ("As represented by block 206, once 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. The transceiver 110 may thus obtain queued packets from the upper layers and ***establish communications with the network interface 112***."); 9:7-9 ("For example, packets 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 claim of the '490 patent, and how the prior art renders this claim obvious. *See also* Lin-¶¶72–111.

A. Ground 1: Claim 31 is Rendered Obvious By Heinrich In View Of Balasubramanian

1. [31a] Preamble: "[a] mobile terminal"

The '490 patent describes the claimed "mobile terminal" as a smart phone, cell phone, tablet, or similar device. Ex-1001, 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 102" that can be "a mobile phone, a tablet, a laptop computer or other embedded device able to connect to the network 110." Ex-1004, 4:21-23; *see also id.*, Fig. 1 (showing user device 102).

2. [31b] "a modem timer"

Claim 31 requires "a modem timer." Heinrich teaches that limitation by disclosing a "scheduler" and, within the scheduler, a "timer." The scheduler can be "associated with the baseband processor"—*i.e.*, the claimed "modem processor"—and controls communications between the baseband processor and the

application processor. Ex-1004, 7:8-21 ("[T]here is a centralized scheduler 120 which is associated with the baseband processor 104…. The scheduler 120 may control the scheduling of IPC activities [*i.e.*, processor-to-processor communications] in both directions between the processors 104 and 106….").

The scheduler associated with the baseband processor includes a timer called a "lazy timer." *Id.*, 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. Each IPC activity is sent on the IPC interface when its respective lazy timer fires."). The "lazy timer" of Heinrich is a "modem timer" because it is within the baseband processor and determines when held data (*i.e.*, IPC activities) is sent to the application processor. *Id.; see also id.*, Fig. 1; 7:10-11 ("In the example shown in FIG. 1, the scheduler 120 is implemented as a software module on the baseband processor 104."); Lin-¶76.

3. [31c] "a modem processor"

Heinrich discloses the "modem processor" of claim 31. For example, as shown in Figure 1, Heinrich discloses user device 102 that includes baseband processor 104:

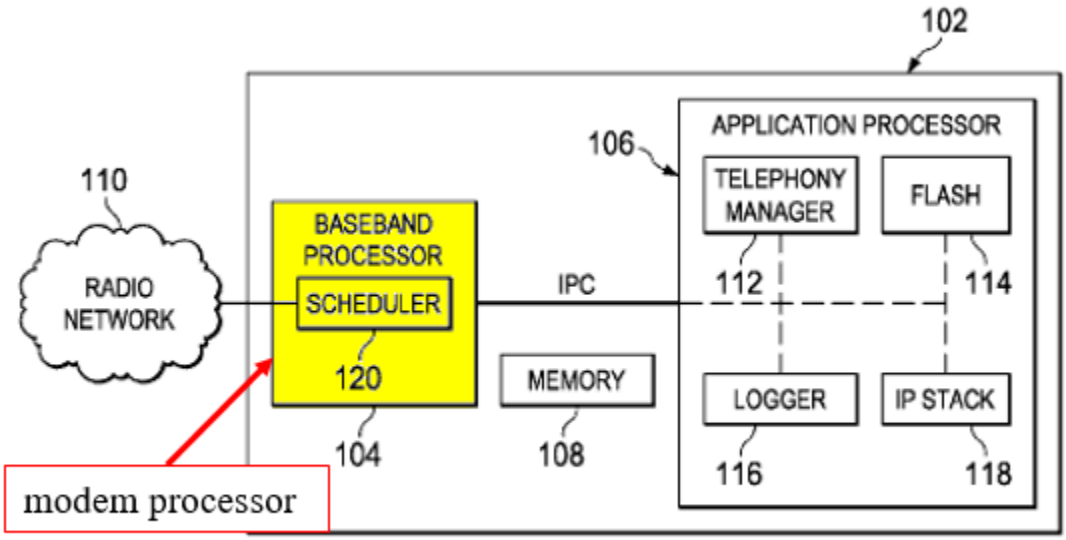

FIG. 1

Ex-1004, Fig. 1. Heinrich discloses that baseband processor 104 is a "modem processor." *Id.*, 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. FIG. 1 shows a link between the baseband processor 104 and the radio network 110 to indicate that the baseband processor 104 is implemented at the user device 102 for communicating with the radio network 110."); Lin-¶77.

4. [31d] "the modem processor configured to hold modem processor to application processor data until expiration of the modem timer"

Claim 31 requires "the modem processor configured to hold modem processor to application processor data until expiration of the modem timer." 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 the modem timer. Ex-1004, 7:65-8:1 ("[T]he scheduler 120 identifies IPC activities that are not real-time sensitive and which 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 sleep mode-to-awake mode transitions and thereby reduces power consumption. *Id.*, 10:44-51 ("Furthermore, ***delaying*** some of the IPC activities to thereby group them together not only results in ***fewer transitions*** between the sleep and awake modes on the application processor 106, but it may also reduce the overall number of IPC activities that are communicated and therefore processed by the application processor 106. This can further ***reduce the power consumed*** by the computer system, in particular the power consumed by the application processor 106."). One of ordinary skill in the art would understand that such delayed and "aggregated IPC activities" would be "modem processor to application processor data" that would have to be held (i.e., buffered) by the baseband processor or a memory connected to the baseband processor until the baseband processor determines that it is time to

transmit the held data to the application processor, such as when the timer expires. Lin-¶78; *see also* Ex-1004, 11:57-58 ("In order to delay file write accesses they are stored in a cache memory of the baseband subsystem.").

Heinrich describes "baseband memory" where baseband data is buffered. *Id.*, 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 person of ordinary skill in the art that the baseband processor in Heinrich can hold data—*i.e.*, the "modem processor to application processor data"—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*, ACM Transactions on Design Automation of Electronic Systems, Vol. 5, No. 3, July 2000, 682-704 ("Panda") (Ex-1011).

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. Ex-1011, 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, showing off-chip DRAM storage and on-chip SRAM and data cache storage):

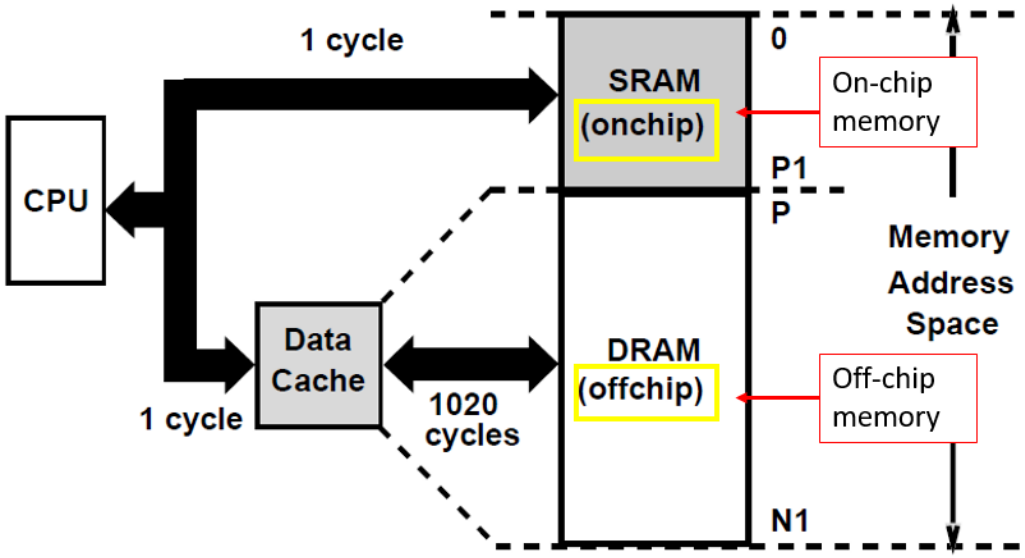

Fig. 2. Division of data address space between SRAM and DRAM.

A person of ordinary skill in the art 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. Lin-¶82; *see also* Ex-1011, 682-683 ("Modern embedded systems are characterized by a trend towards increasing levels of chip-level integration. System design is gradually

changing its focus from the integration of chips on a board to the integration of complex electronic components on the same chip."). This allows for fewer chips in a device, reducing cost. *Id*., 683 ("This shift in focus, which is possibly due to increasing chip capacities, is driven by the goal of cost reduction that stems from reduced chip count."). ***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.* ("[D]ata stored in embedded [*i.e.*, on-chip] DRAM can be accessed much faster than that in off-chip DRAM. . . ."); Lin-¶82. Because using on-chip memory to hold data was well-known in systems like Heinrich, a person of ordinary skill in the art would have reasonably expected the design to succeed. Lin-¶82. "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).

Heinrich thus discloses or renders obvious a "modem processor configured to hold modem processor to application processor data until expiration of the modem timer."

5. [31e] "an application processor"

Heinrich discloses an "application processor." As shown in Figure 1, Heinrich discloses a mobile communication system that includes an "application processor 106":

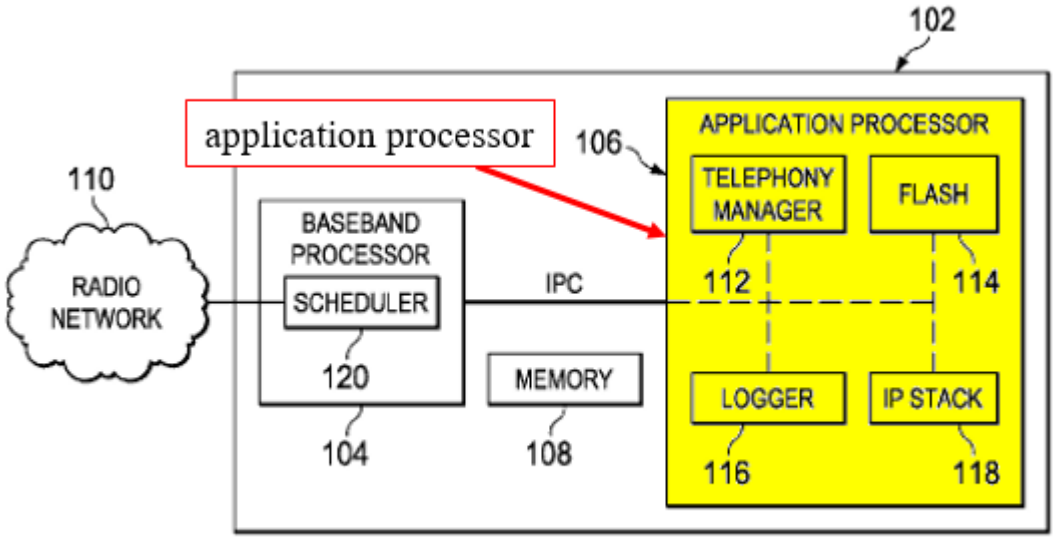

FIG. 1

Ex-1004, Fig. 1, 4:36-42 ("The ***application processor 106*** executes an operating system of the user device 102 and handles other multimedia features on the user device 102. For example, the application processor 106 processes data relating to peripherals (not shown in FIG. 1) of the user device 102 such as a display, a WiFi module, a GPS module, etc."). Lin-¶84.

6. [31f] "an interconnectivity bus communicatively coupling the application processor to the modem processor"

Heinrich discloses "an interconnectivity bus communicatively coupling the application processor to the modem processor." As shown in Figure 1, a bus labeled "IPC" ("inter processor communication") couples the baseband processor 104—*i.e.*, modem processor—to the application processor 106:

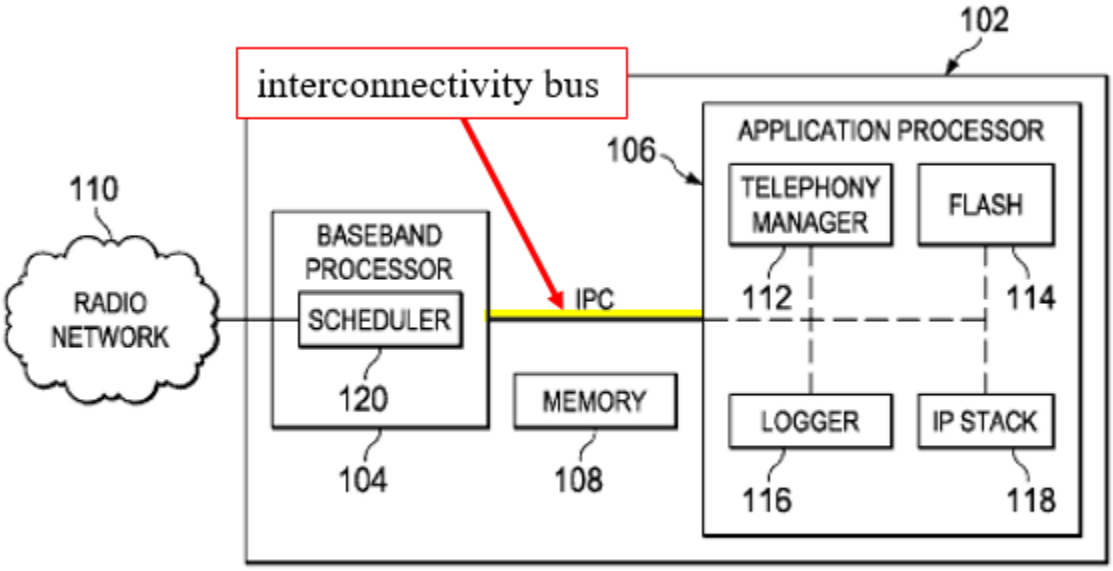

FIG. 1

Ex-1004, Fig. 1; Lin-¶85.

The baseband processor and application processor communicate over the IPC bus. Ex-1004, 4:65-5:1 ("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."). Lin-¶86.

The IPC bus in Heinrich can be any one of many industry-standard buses. Ex-1004, 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 one of: (i) a Universal Serial Bus (USB) interface, (ii) a Mobile Industry Processor Interface (MIPI), such as a High-Speed Synchronous Interface (HSI), (iii) a Serial Peripheral Interface (SPI), or (iv) a shared memory."). Heinrich thus discloses "an interconnectivity bus communicatively coupling the application processor to the modem processor." Lin-¶87.

7. [31g] "the application processor configured to hold application processor to modem processor data until the modem processor pulls data from the application processor after transmission of the modem processor to application processor data"

The combination of Heinrich and Balasubramanian discloses "the application processor configured to hold application processor to modem processor data until the modem processor pulls data from the application processor after transmission of the modem processor to application processor data."

"the application processor configured to hold application processor to modem processor data": Heinrich discloses or renders obvious that the "application processor" is "configured to hold application processor to modem processor 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* claim limitation [31d]. Heinrich further explains that the same methods used to transmit data from the baseband 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-1004, 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***."). Lin-¶89.

To implement this transmission scheme, the application processor in Heinrich includes a separate "scheduler" that controls transmission of data from the application processor to the baseband processor. Ex-1004, 7:19-27 ("The scheduler 120 may control the scheduling of IPC activities in both directions between the processors 104 and 106. Alternatively, … 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***."). Like the scheduler in the baseband processor, the scheduler in the application processor identifies "IPC activities"—*i.e.,* data—that are not "real-time sensitive" and can therefore be delayed and stored prior to transmission to the baseband processor. *Id*., 7:65-8:1 ("[T]he scheduler 120 identifies IPC activities

that are not real-time sensitive and which can be delayed until it is deemed profitable to run them."). *See also id.*, 7:7-27; Lin-¶90.

The claim requires that the "application processor" "hold[s]" the data. Heinrich does not expressly specify where such data is held. But it would have been obvious to a person of ordinary skill in the art that the application processor in Heinrich can hold its data—*i.e.*, the "application processor to baseband processor data"—in its own memory on the same chip as the processor circuitry.

As explained above in claim limitation [31d], a person of ordinary skill in the art 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 and (2) on a separate or external chip. Lin-¶92. A person of ordinary skill in the art 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 [31d].

Moreover, Balasubramanian discloses a first processing node (*i.e.*, transceiver 110) coupled to a second processing node (*i.e.*, network interface 112) by a communication link 116, which can be a wired link.

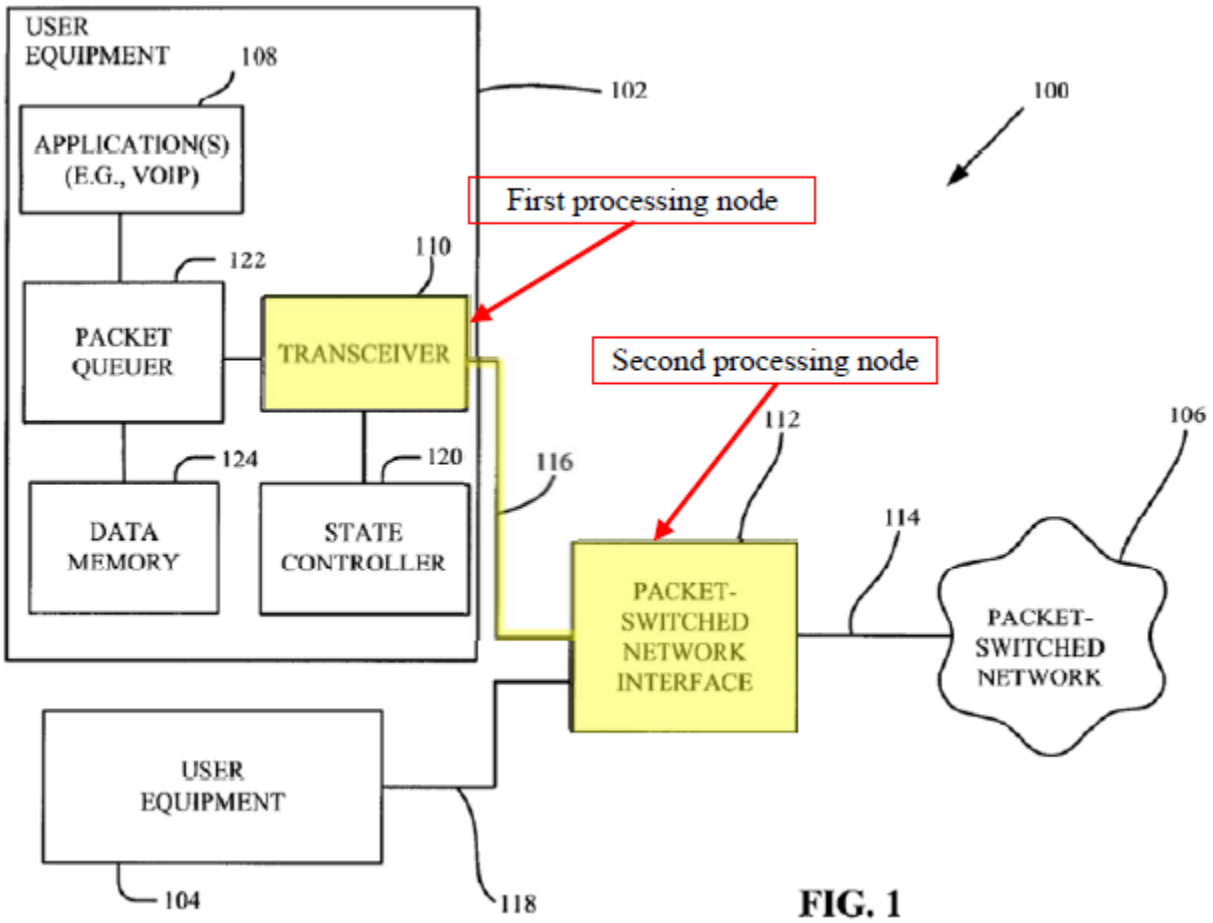

FIG. 1

Ex-1005, Fig. 1, 4:50-54 ("[T]he sub-network may communicate via some other protocol (e.g., a wire-based protocol or a wireless-based protocol) over communication links 116 and 118."). Balasubramanian further discloses that data may be "queue[d]" (*i.e.*, held) on both sides of the link when a processing node is in a low power state—*i.e.*, user equipment 102 with transceiver 110 (a first processing node) could hold data intended for later transmission to network interface 112 (a second processing node) over the link, and the second processing node could hold data intended for later transmission to the first processing node.

See id., 5:47-61 ("To increase the amount of time the transceiver 110 is in the suspended state (and thereby conserve more power), the ***user equipment 102 and/or the network interface 112 may be adapted to queue packets*** while the transceiver 110 is in the suspended state. The ***equipment 102 and the interface 112 may be adapted to group (e.g., consolidate, bundle, combine, etc.) the queued packets for transmission over the communication link 116 when the transceiver 110 is in the active state***. 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 by the user equipment 102 or every time it is expected that the user equipment 102 will receive a packet. Accordingly, power savings may be achieved by rescheduling the packet traffic into groups of traffic.").

From this disclosure of Balasubramanian, a person of ordinary skill in the art would have found it obvious that a modem processor and an application processor both could each hold data intended for the other while the link between them is not active. Lin-¶94.

"until the modem processor pulls data from the application processor after transmission of the modem processor to application processor data": The combination of Heinrich and Balasubramanian discloses an application processor that holds data "until the modem processor pulls data from the application

processor after transmission of the modem processor to application processor data."

There are two common ways of initiating a data transmission from a first processor to a second processor: (1) a "push" and (2) a "pull." *See, e.g.*, Duan et al., *Push vs. Pull: Implications of Protocol Design on Controlling Unwanted Traffic*, SRUTI, July 7, 2005, §2.1 (Ex-1010) ("In the sender-push model, the sender knows the identity of a receiver in advance and pushes the message in an asynchronous manner to the receiver. . . . The biggest disadvantage of the sender-push model is that it is the sender who completely controls *what* message is delivered and *when* it is delivered."); §2.2 ("In the receiver-pull model, it is the receiver who initiates the message transfer by explicitly contacting the sender. The sender passively waits for the receiver and delivers the entire content upon receiving a request."). When performing a "push," a first processor initiates a data transmission on its own, often upon the expiration of a timer or when a counter threshold is reached. *See id.* ("In the sender-push model, a sender can deliver traffic at will to a receiver, who can only passively accept the traffic, such as in the SMTP-based email delivery system."). When performing a "pull," the second processor receives data from the first processor in response to a request for the data from the second processor. *See id.* ("In contrast, in the receiver-pull model, receivers can regulate if and when they wish to retrieve data, such as the HTTP-

based web access system."); §2.2 ("In the receiver-pull model, it is the receiver who initiates the message transfer by explicitly contacting the sender. The sender passively waits for the receiver and delivers the entire content upon receiving a request."). Lin-¶96.

Balasubramanian discloses a second processing node (*i.e.*, network interface 112) that holds its data until a first processing node (*i.e.*, transceiver 110) pulls data from the second processing node after transmission of data to the second processing node. Lin-¶97. Balasubramanian discloses that after transmission of data from a first processing node (*e.g.*, the transceiver 110) to a second processing node (*e.g.*, the network interface 112), the first processing node can then pull data from the second processing node. Ex-1005, 6:63-7:11 ("[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***…. [T]he transceiver 110 may send a message to the network interface 112 ***requesting transmission of all queued packets***."); *see also id.*, 6:15-19 ("Conversely, in response to a ***request*** by the transceiver 110 or some other indication, the network interface 112 may send any of its queued downlink packets to the transceiver 110

in succession over the communication link 116."). Lin-¶97. Balasubramanian's disclosure of the transceiver 110 receiving all queued packets from the network interface 112 in response to a message from the transceiver 110 requesting the queued data is a "pull" of data by the transceiver 110, because the transceiver 110 initiates and controls the transfer of the data from the network interface 112 to the transceiver 110. Lin-¶97.

Balasubramanian thus teaches the requirement of claim limitation [31g] that requires an application processor to hold data "until the modem processor pulls data from the application processor after transmission of the modem processor to application processor data"—except Balasubramanian does not explicitly disclose data transmissions between an application processor and a modem processor. It instead teaches transmissions between a transceiver 110 and a network interface 112 over a link. Heinrich, however, teaches transmission of data between an application processor and a modem processor over a link, as well as the aggregation of such data (*i.e.*, buffering the data and sending it all at once). The combination of the Heinrich and Balasubramanian therefore discloses claim limitation [31g]. Lin-¶98.

Combination of Heinrich and Balasubramanian: A person of ordinary skill would have been motivated to incorporate Balasubramanian's pulling-after-transmission techniques—where a first processing node pulls data from a second

processing node after transmission of all held data from the first processing node—with Heinrich's interprocessor communication system and scheduling techniques to form the claimed combination of claim 31 of the '490 patent. Lin-¶99. A person of ordinary skill in the art would have been motivated to combine Heinrich and Balasubramanian for at least four reasons (and would have reasonably expected the combination to succeed for its intended purpose).

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-1004, 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-1005, 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 person of ordinary skill in the art to look to Balasubramanian when considering the power consumption issues of Heinrich. Lin-¶100.

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 person of ordinary skill in the art would have been motivated to use the techniques taught by Balasubramanian to address the same issues described in Heinrich. Lin-¶101.

Third, Balasubramanian's teaching of sending data from a first processing node (the transceiver 110) to a second processing node (the network interface 112) and then pulling data from the second processing node would fit naturally into Heinrich's disclosure of having a scheduler control data transfers in both directions over an IPC link. Indeed, Heinrich teaches that a "good time" to schedule communications over the IPC link is when the application processor is in the awake state, and one way to know that a processor is in an awake state is when data is being transmitted to it. Ex-1004, 7:19-21 ("The scheduler 120 may control the scheduling of IPC activities in both directions between the processors 104 and 106."); *id.*, 9:13-21 ("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, thereby waking 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 scheduler 120 can deduce that the application processor 106 is in the awake mode***."). Lin-¶102.

Fourth, a person of ordinary skill in the art would realize that the same advantages taught by Balasubramanian would be gained by applying the same approach to the system in Heinrich. Lin-¶103. Balasubramanian explains that combining the transmission and receipt of packets into a single active state of the transceiver 110 can result in power savings, because it increases the amount of time that the transceiver 110 is in the suspended state. Ex-1005, 5:55-61 ("[P]ower 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 by the user equipment 102 or every time it is expected that the user equipment 102 will receive a packet. Accordingly, power savings may be achieved by rescheduling the packet traffic into groups of traffic."); 1:52-62 ("In some aspects power savings are achieved in an apparatus by grouping packets. For example, packets may be queued while an apparatus is in a suspended state. . . . The apparatus may then transition from the suspended state to a wake state (*e.g.*, a normal operating state) to transmit and/or receive a group of queued packets. Advantageously, the group of queued packets may be transmitted and/or received in relatively close

succession during a single wake state."); 14:49-63 ("In view of the above it should be appreciated that numerous advantages may be achieved using the teachings herein. . . . By "waking" less often to transmit and/or receive packets, the number of transitions between active and suspended states (*e.g.*, turning the transceiver on and off) will be reduced. Accordingly, savings may be achieved through avoidance of the lag time associated with turning the transceiver on and off. Here, the transceiver may use less power since the power consumption associated with some of the lag time may be saved."). Applying the power-saving approach in Balasubramanian to Heinrich would result in a system in which the application processor buffers its data until it is pulled by the modem processor, which occurs after the modem processor transmits data to the application processor and during the same active state of the apparatus. Lin-¶103.

Reasonable Expectation of Success: A person of ordinary skill in the art would have had a reasonable expectation of success in combining Balasubramanian's "pull" scheme with the processor-to-processor communications system of Heinrich. Both "push" and "pull" schemes were very well-known in the art, and a person of ordinary skill in the art would have had no reason to doubt that a pull scheme could have been implemented in the system of Heinrich. Lin-¶104. Further, scheduling the pull to occur during the same active state would not have required any alterations to the bus protocol — it would merely have involved

routine scheduling, which would also have been considered trivial to a person of skill in the art. Lin-¶104. Thus, a person of skill in the art would have reasonably expected that the combination of Balasubramanian's "pull" scheme with the system of Heinrich would work as described herein.

8. [31h] "wherein the modem processor is further configured pull data from the application processor after transmission of the modem processor to application processor data and before the interconnectivity bus transitions from an active power state to a low power state."

As explained for claim limitation [31g] above, the combination of Heinrich and Balasubramanian discloses that "the modem processor is … configured [to] pull data from the application processor after transmission of the modem processor to application processor data." Heinrich in combination with Balasubramanian discloses the further requirement of claim limitation [31h], which requires that the modem processor transmits data to and pulls data from the application processor "before the interconnectivity bus transitions from an active power state to a low power state."[4]

[4] Claim 31 does not require that the interconnectivity bus ever enter a low power state, and states merely that the modem processor transmit data to and pull data from the application processor *before* the interconnectivity bus enters a low power state (*i.e.*, during the same active state).

Balasubramanian teaches that the communication link 116 between the transceiver 110 and the network interface 112 can be a wired connection, which is an interconnectivity bus. *See* Ex-1005, 4:50-53 ("[T]he sub-network may communicate via . . . a wire-based protocol . . . over communication links 116 and 118."); *id.*, 7:26-28 ("[T]he teachings herein may be incorporated into a wire-based or wireless communication system.").

Balasubramanian also discloses pushing and pulling data between two processing nodes (the transceiver 110 and network interface 112) before the interconnectivity bus between the two nodes transitions from an active power state to a low power state. ***First***, Balasubramanian teaches performing push and pull transactions during a single active state of the transceiver 110. *Id.*, 2:23-24 ("Advantageously, these uplink and downlink packets may be transmitted during a ***single wake state*** of the apparatus."); 6:63-7:11 ("[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***…. [T]he transceiver 110 may send a message to the network interface 112 requesting transmission of all queued packets."); Lin-¶107.

Balasubramanian therefore discloses performing push and pull transactions during a single wake state of the communication link.

Second, the single wake state of the transceiver 110 in Balasubramanian defines a single wake state of the communication link 116 because the transceiver 110 is the circuit that drives data onto and receives data from the communication link 116. Ex-1005, 4:39-42 ("[A] transceiver 110 may provide a physical layer interface (and, optionally, a data link layer interface) that handles the physical transmission of packets from and to the user equipment 102."); 5:2-4 ("Alternatively, transmit components and receive components may be independently transitioned between active and power save modes."); 5:10-29 (disclosing placing the transceiver 110 into a low-power state while other components that generate and queue data remain in an active state); Lin-¶108.

Because Heinrich similarly discloses a bus with low and high power states and performing a data transfer between two processors during a single wake state of a bus, a person of ordinary skill in the art would have been motivated to apply Balasubramanian's teaching of performing push and pull transactions during a single wake state of communication link 116, to the system of Heinrich, to create a system in which a modem processor pushes data to, and pulls data from, an application processor during a single active state of the communication link between the modem processor and application processor. Lin-¶109.

In particular, Heinrich discloses that the bus over which the processors communicate can be a USB bus. Ex-1204, 4:44-48 ("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 explains that the bus can transition from a sleep mode to an active mode. Specifically, each processor in Heinrich includes a "USB interface" that can transition from sleep mode to active mode to place the bus into sleep (*i.e.*, low power) mode or active (*i.e.*, high power) mode. *Id.*, 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")."); 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 further 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"); *see also 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-¶110.

A person of ordinary skill in the art would also have been motivated to apply Balasubramanian's teaching of performing push and pull transactions during a single wake state of communication link 116, to the system of Heinrich, to further advance Heinrich's goal of achieving power savings by aggregating multiple data transactions for transmission over a physical IPC interface (i.e., the bus) between modem and application processors. Ex-1004, 8:37-43 ("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-¶109. Further, Heinrich teaches a structure designed to do just this—a scheduler that detects that the physical IPC interface (*i.e.*, the bus) is in an active state, and the application processor is awake, when the baseband processor transmits data to the application processor. Ex-1004, 9:54-58 ("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."); Lin-¶111.

A person of ordinary skill in the art would have been further motivated to combine Heinrich and Balasubramanian and would have reasonably expected the combination to succeed for its intended purpose for the same reasons described above for claim limitation [31g]. *See* Section IX.A.7 *supra.* Thus, Heinrich in combination with Balasubramanian discloses this limitation. Lin-¶112.

X. CONCLUSION

Based on the foregoing, claim 31 of the '490 patent recites subject matter that is obvious. The Petitioner requests institution of an *inter partes* review to cancel that claim.

Respectfully submitted,

Intel Corp.,
Petitioner

By: ___/Jason D. Kipnis/_____
Jason D. Kipnis
Registration No. 40,680
Wilmer Cutler Pickering
Hale and Dorr LLP

CERTIFICATE OF SERVICE

I hereby certify that on June 29, 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 1001-1017
- List of Exhibits for Petition for *Inter Partes* Review of U.S. Patent No. 9,535,490 (Exhibits 1001-1017)
- 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

/Christopher R. O'Brien/

Christopher R. O'Brien
Registration No. 63,208

CERTIFICATE OF COMPLIANCE

I hereby certify that the foregoing, Petition for *Inter Partes Review* of U.S. Patent No. 9,535,490, contains 10,679 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: June 29, 2018

/Christopher R. O'Brien/
Christopher R. O'Brien
Registration No. 63,208

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

Exhibit	Description
1001	U.S. Patent No. 9,535,490
1002	Declaration of Dr. Bill Lin
1003	File History for U.S. Patent No. 9,535,490
1004	U.S. Patent No. 9,329,671 to Heinrich et al. ("Heinrich")
1005	U.S. Patent No. 8,160,000 to Balasubramanian ("Balasubramanian")
1006	PCT Publication No. WO 2009/039034 to Tsai ("the Intel PCT")
1007	U.S. Patent No. 6,021,264 to Morita ("Morita")
1008	Kaplan, Wiley Electrical and Electronics Engineering Dictionary, IEEE Press, John Wiley & Sons, 2004 ("Kaplan")
1009	Downing, et al, Dictionary of Computer and Internet Terms, Barron's Educational Series, Inc., 2013 ("Downing")
1010	Duan et al., "Push vs. Pull: Implications of Protocol Design on Controlling Unwanted Traffic," SRUTI. July 7, 2005 ("Duan")
1011	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")
1012	Houghton Mifflin Co., The American Heritage College Dictionary, 2000 ("Heritage Dictionary")
1013	Soanes et al., Concise Oxford English Dictionary, 11th Ed., Oxford University Press, 2004 ("Oxford Dictionary")
1014	Merriam-Webster, Inc., Merriam-Webster's Collegiate Dictionary, 11th Ed., 2000 ("Merriam-Webster's Dictionary")
1015	Gerber, et al., "P2P the gorilla in the cable," National Cable & Telecommunications Association (NCTA) 2003 National Show, June 2000 ("Gerber")

Exhibit	Description
1016	Kolbehdari, et al., "The Emergence of PCI Express in the Next Generation of Mobile Platforms," Intel Technology Journal, Vol. 9, No. 1, 2005 ("Kolbehdari")
1017	U.S. Patent No. 8,112,646 to Tsai ("Tsai")