UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

Intel Corporation
Petitioner

v.

Qualcomm Incorporated
Patent Owner

U.S. Patent No. 9,535,490
Claims 9, 11-13, 26, 27

Case IPR2018-01346

DECLARATION OF BILL LIN, PH.D. ON BEHALF OF PETITIONER

TABLE OF CONTENTS

I. BACKGROUND 1
II. MATERIALS CONSIDERED 4
III. LEGAL PRINCIPLES 5
A. Claim Construction 5
B. Anticipation 6
C. Obviousness 7
IV. SUMMARY OF OPINIONS 9
V. BRIEF DESCRIPTION OF THE TECHNOLOGY 9
A. Processor-To-Processor Communications 9
B. Communication Bus Power-Saving States 11
C. Data Buffering 12
VI. OVERVIEW OF THE '490 PATENT 14
A. Alleged Problem of the Prior Art 15
B. Purported Solution of the '490 Patent 16
C. Summary Of The Prosecution History of the '490 Patent 22
VII. OVERVIEW OF THE PRIOR ART REFERENCES 25
A. U.S. Patent No. 9,329,671 To Heinrich *et al.* 26
B. U.S. Patent No. 8,160,000 to Balasubramanian 29
VIII. CLAIM CONSTRUCTION 33

A. "triggered by" 34

IX. LEVEL OF ORDINARY SKILL IN THE ART 35

X. SPECIFIC GROUNDS FOR CHALLENGE 36

A. Ground 1: Claims 9, 11-13, 26, and 27 Are Rendered Obvious By Heinrich In View Of Balasubramanian 36

1. Claim 1 36

2. Claim 9 65

3. Claim 12 67

4. Claim 11 69

5. Claim 13 71

6. Claim 26 74

7. Claim 27 77

XI. AVAILABILITY FOR CROSS-EXAMINATION 96

XII. RIGHT TO SUPPLEMENT 97

XIII. JURAT 97

I, Bill Lin, Ph.D declare as follows:

I. BACKGROUND

1. I am currently Professor and Vice Chair of Electrical and Computer Engineering at the University of California, San Diego (UCSD). I am also Adjunct Professor of Computer Science and Engineering at UCSD.

2. My Curriculum Vitae, which states my qualifications more fully, is attached as Appendix A. A list of all cases in which I have testified as an expert at trial or by deposition in the last four years is also included in Appendix A.

3. I received a Bachelor's of Science degree in 1985, a Master's of Science degree in 1988, and a Ph.D. in 1991, all in Electrical Engineering and Computer Sciences from the University of California, Berkeley.

4. I joined UCSD in 1997, and I have been a tenured professor since 1999. My teaching and research has focused on computer architecture and computer network problems, including the design of multiprocessor and multi-core processor architectures, multiprocessor and multi-core processor interconnection buses and networks, network processors, systems-on-chips, and data networks. I regularly teach a senior-level design course on the design of advanced processors, and I have taught graduate courses in hardware/software co-design and advanced special topics in computer architecture.

5. At UCSD, I am a Principal Investigator in the UCSD Center for Networked Systems (CNS). CNS brings together researchers to work on a range of challenges in the design of future networked systems. My contribution to CNS has been expertise in the design of computer architecture solutions for packet processing and computer networking. I am also a Principal Investigator in the UCSD Center for Wireless Communications (CWC). CWC brings together researchers to work on a range of challenges in the design of future wireless communications systems. My contribution to CWC has been expertise in the design of multi-core processor architectures for wireless communications and mobile computing.

6. Prior to joining UCSD, I was the Head of the Systems and Communications Group at IMEC in Leuven, Belgium, where I led a team of researchers who worked on a range of computer design problems, including hardware/software co-design, processor interfaces, multiprocessor and multi-core processor design methodologies, and specialized processors for wireless communications and computer networking.

7. During my career, I have received or worked on research efforts that received millions of dollars in research funding from both government agencies and industry, including funding for research in multi-core processor design,

system-on-chips, hardware/software co-design, packet processing, and computer networks.

8. I have served as an Associate or Guest Editor for several journals published by the Association for Computing Machinery ("ACM") and the Institute of Electrical and Electronics Engineers ("IEEE"). I have also served as General Chair of several ACM/IEEE conferences, and on the Organizing or Steering Committees of many ACM/IEEE conferences, and I have served on the Technical Program Committees of numerous ACM/IEEE conferences. I am the author of over 170 peer-reviewed publications in the field of computer engineering dating to the 1980s, including journal articles, conference papers, book chapters, technical reports, and invited papers. A number of these publications have received best paper awards or distinguished paper citations. I have also given numerous invited and keynote talks around the world. A list of my publications within the last ten years is included in my CV.

9. I am the inventor of five patents: U.S. Patent Nos. 8,443,444, 7,860,004, 7,672,005, 5,870,588, and 5,748,487.

10. I have been retained by counsel for Intel Corporation ("Petitioner") as an independent expert witness for the above captioned Petition for *Inter Partes* Review of claims 9, 11-13, 26, and 27 of U.S. Patent No. 9,535,490 (the "'490

patent") (Ex-1401). I am being compensated at my normal consulting rate of $550 for my work. My compensation is not dependent on and in no way affects the substance of my statements in this Declaration.

11. I have no financial interest in the Petitioner. I similarly have no financial interest in the '490 patent, and have had no contact with the named inventors of the '490 patent.

II. MATERIALS CONSIDERED

12. I have reviewed the specification, claims, and file history of the '490 patent. I understand that 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.

13. I have also reviewed the following references, all of which I understand to be prior art to the '490 patent:

- U.S. Patent No. 9,329,671 to Heinrich et al. ("Heinrich") (Ex-1404)
- U.S. Patent No. 8,160,000 to Balasubramanian ("Balasubramanian") (Ex-1405)

14. In addition to the documents listed above, I have also reviewed the file history of the '490 patent, all of the documents listed in Petitioner's List of

Exhibits in the accompanying petition and all other documents cited in this declaration.

III. LEGAL PRINCIPLES

15. I am not an attorney. For the purposes of this declaration, I have been informed about certain aspects of the law that are relevant to my opinions. My understanding of the law is as follows:

A. Claim Construction

16. I have been informed that claim construction is a matter of law and that the final claim construction will ultimately be determined by the Board. For the purposes of my analysis in this proceeding and with respect to the prior art, I have been informed that patents are currently reviewed in an *inter partes* review (IPR) proceeding under the "broadest reasonable interpretation" standard (hereinafter "BRI standard"). I also have been informed that IPRs may soon be reviewed under what is known as "the Phillips standard."

17. I have been informed that the BRI standard refers to the broadest reasonable interpretation that a person of ordinary skill in the art would give to a claim term in light of the specification.

18. I have been informed that under the Phillips standard, claim terms are generally given their plain and ordinary meaning as understood by a person of

ordinary skill in the art at the time of the invention, with the claim term read not only in the context of the particular claim in which the disputed term appears, but also in the context of the entire patent, including the specification.

19. I have been informed that the patentee can serve as his or her own lexicographer. As such, if a claim term is provided with a specific definition in the specification, that claim term should be interpreted in light of the particular definition provided by the patentee.

B. Anticipation

20. I have been informed and understand that a patent claim is invalid if it is "anticipated" by prior art. For the claim to be invalid because it is anticipated, all of its requirements must have existed in a single device or method that predates the claimed invention, or must have been described in a single publication or patent that predates the claimed invention. A patent claim may be "anticipated" if each element of that claim is present either explicitly, implicitly, or inherently in a single prior art reference. I have also been informed that, to be an inherent disclosure, the prior art reference must necessarily disclose the limitation, and the fact that the reference might possibly practice or contain a claimed limitation is insufficient to establish that the reference inherently teaches the limitation.

C. Obviousness

21. I have been informed and understand that a patent claim is invalid if the claimed invention would have been obvious to a person of ordinary skill in the art at the time the application was filed. This means that, even if all of the requirements of a claim are not found in a single prior art reference, the claim is not patentable if the differences between the subject matter in the prior art and the subject matter in the claim would have been obvious to a person of ordinary skill in the art at the time the application was filed.

22. I have been informed and understand that a determination of whether a claim would have been obvious should be based upon several factors, including, among others:

- the level of ordinary skill in the art at the time the application was filed;
- the scope and content of the prior art; and
- what differences, if any, existed between the claimed invention and the prior art.

23. I have been informed and understand that the teachings of two or more references may be combined in the same way as disclosed in the claims, if such a combination would have been obvious to one having ordinary skill in the

art. In determining whether a combination based on either a single reference or multiple references would have been obvious, it is appropriate to consider, among other factors:

- whether the teachings of the prior art references disclose known concepts combined in familiar ways, which, when combined, would yield predictable results;
- whether a person of ordinary skill in the art could implement a predictable variation, and would see the benefit of doing so;
- whether the claimed elements represent one of a limited number of known design choices, and would have a reasonable expectation of success by those skilled in the art;
- whether a person of ordinary skill would have recognized a reason to combine known elements in the manner described in the claim;
- whether there is some teaching or suggestion in the prior art to make the modification or combination of elements claimed in the patent; and
- whether the innovation applies a known technique that had been used to improve a similar device or method in a similar way.

24. I understand that one of ordinary skill in the art has ordinary creativity, and is not an automaton.

25. I understand that in considering obviousness, it is important not to determine obviousness using the benefit of hindsight derived from the patent being considered.

26. I have been informed and understand that a single reference can alone render a patent claim obvious, if any differences between that reference and the claims would have been obvious to a person of ordinary skill in the art at the time of the alleged invention—that is, if the person of ordinary skill could readily adapt the reference to meet the claims of the patent, by applying known concepts to achieve expected results in the adaptation of the reference.

IV. SUMMARY OF OPINIONS

27. It is my opinion that every limitation of claims 9, 11-13, 26, and 27 of the ’490 patent is disclosed by the prior art, and that claims 9, 11-13, 26, and 27 are rendered obvious by the prior art cited in this declaration.

V. BRIEF DESCRIPTION OF THE TECHNOLOGY

A. Processor-To-Processor Communications

28. 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).

29. 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 the use the data (*e.g.*, a phone application can play received voice data or a web browser application can display received web page data).

30. 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:

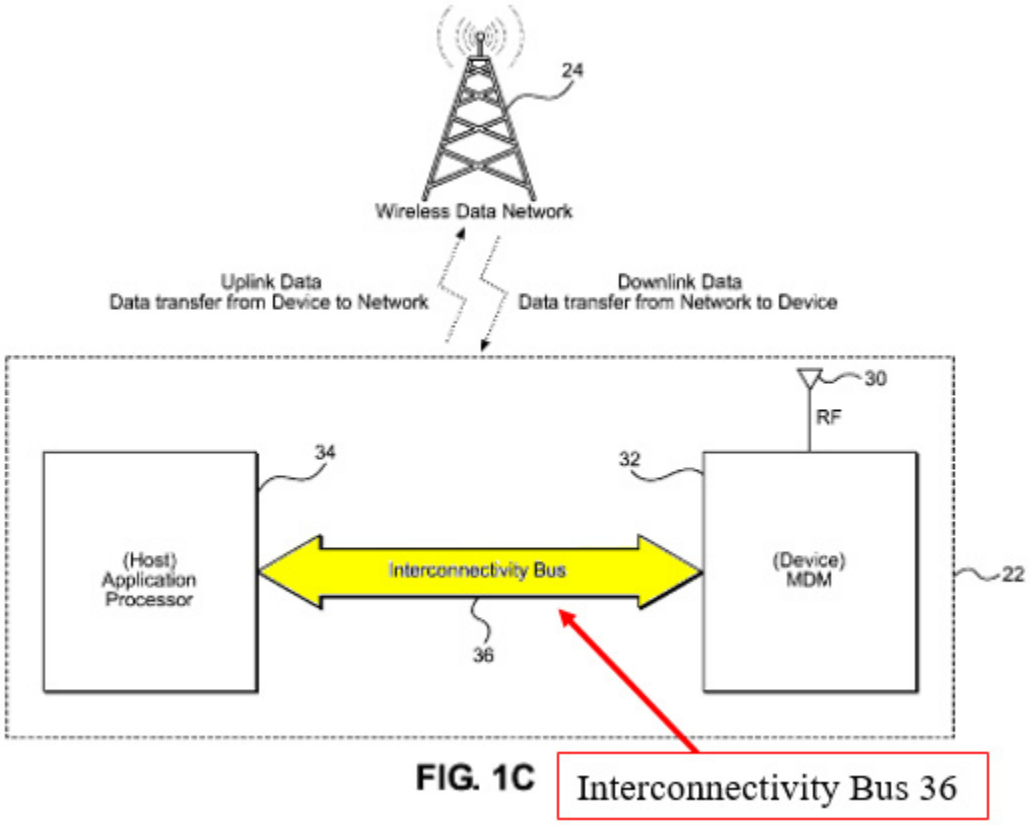


FIG. 1C

Ex-1401, 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.

B. Communication Bus Power-Saving States

31. 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-1401, 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.").

32. 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. *Id.*, 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.").

C. Data Buffering

33. 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.

34. 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. *See, e.g.*, Ex-1405, 5:47-51, 6:40-43, 9:4-7; Ex-1404, 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.

35. 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.

36. 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-1404, 9:22-24; Ex-1405, 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-1405, 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.

VI. OVERVIEW OF THE '490 PATENT

37. The application leading to the '490 patent (Ex-1401) 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]

[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

38. 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

39. The ’490 patent explains 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-1401, 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 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:

---

patent are enabled by or have written description support in the ’498 and/or ’073 provisional applications.

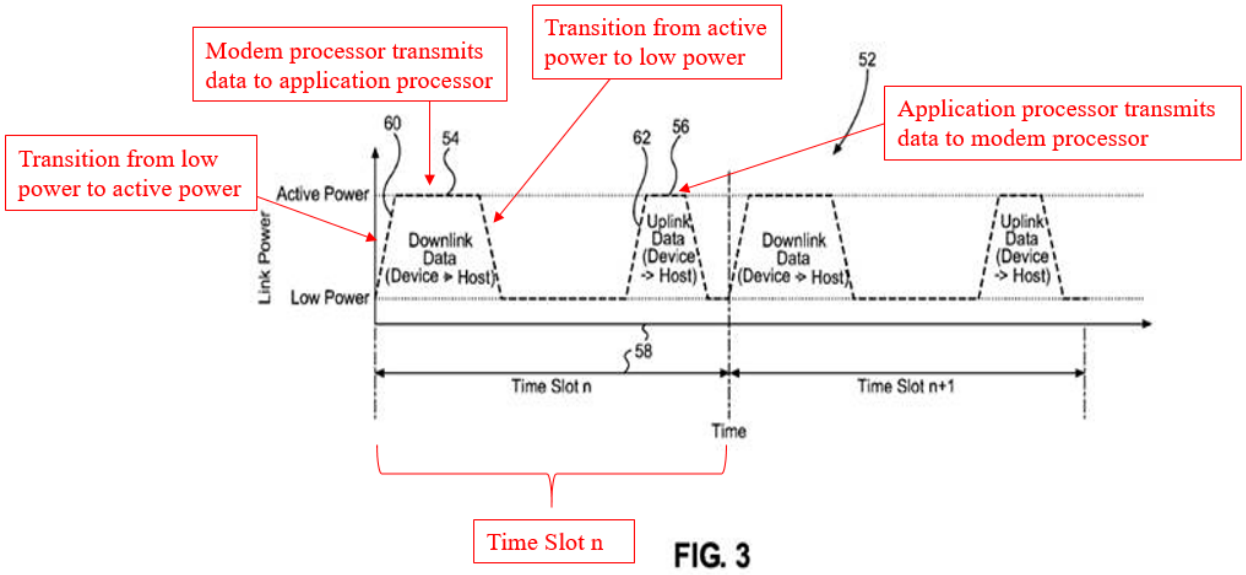


FIG. 3

Ex-1401, Fig. 3 (annotations added), 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"); 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

40. 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 to solve this problem. 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:

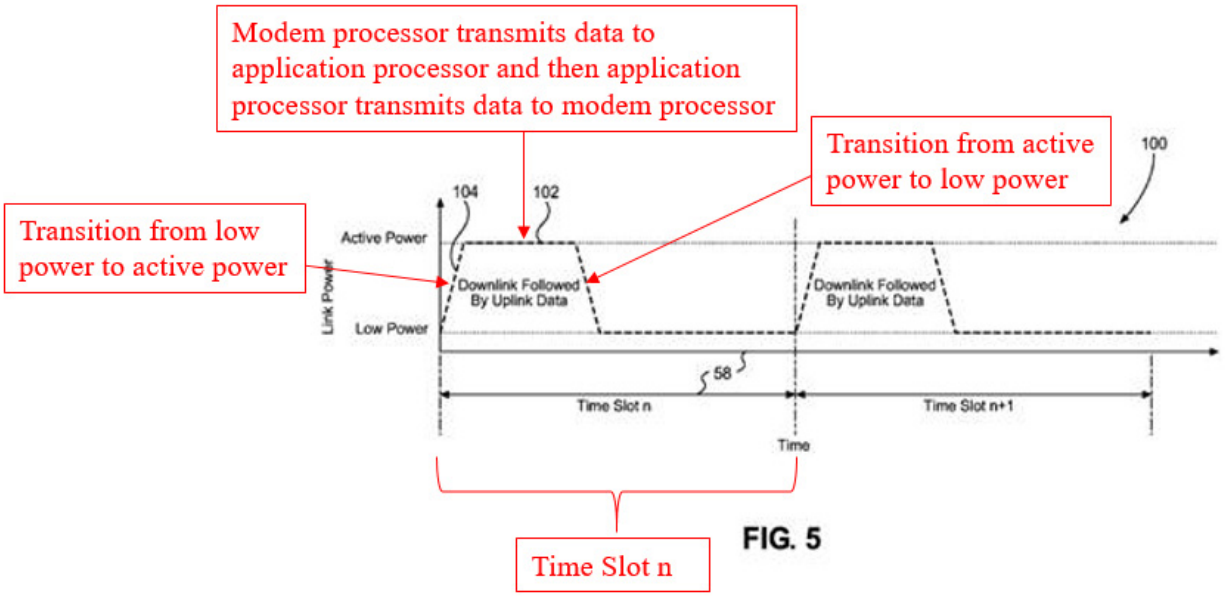


FIG. 5

Ex-1401, 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.").

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

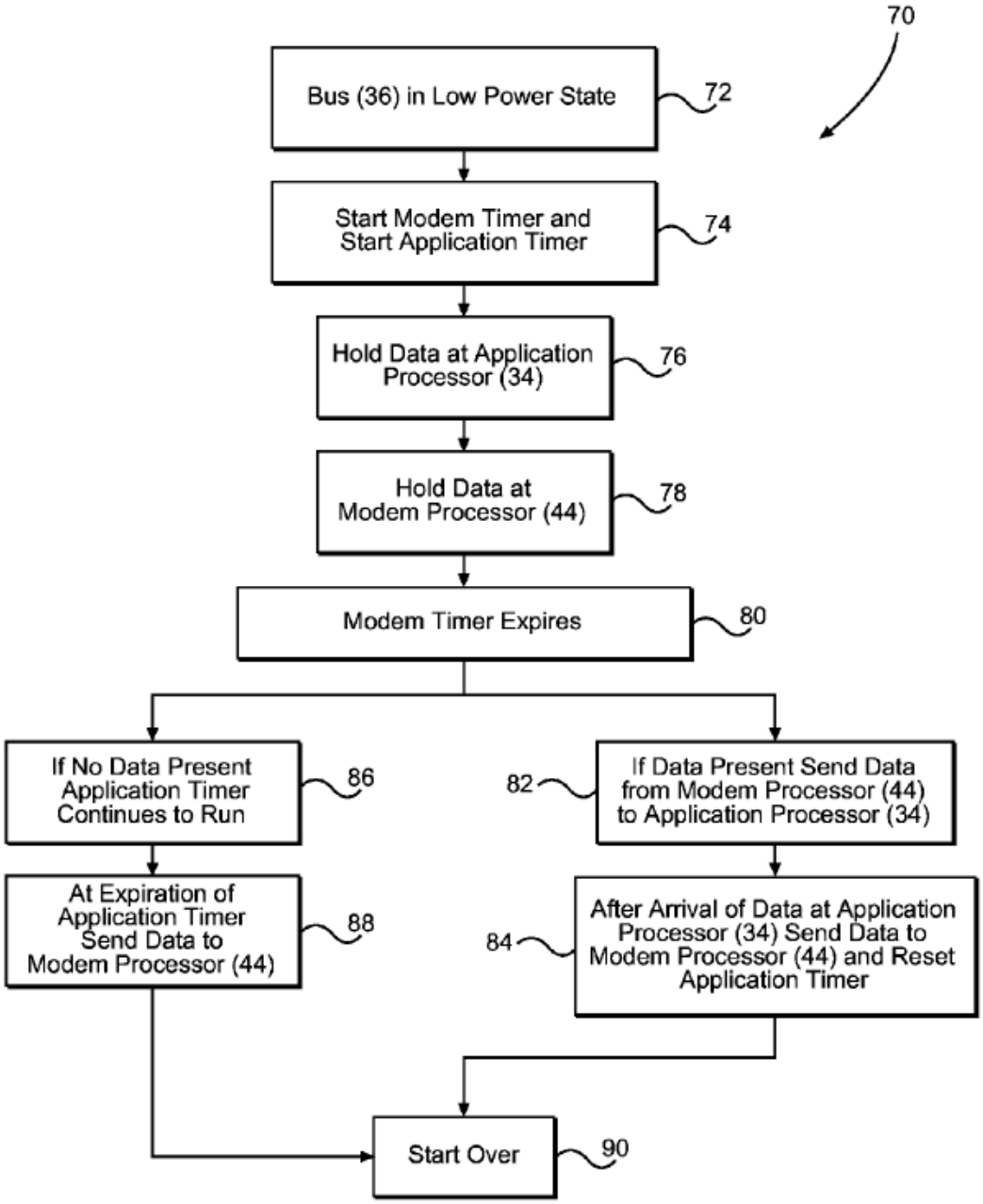


FIG. 4

Ex-1401, 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.").

42. 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.*, 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 processor

43. 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.").

44. 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-1405, 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

---

will send any data that it has buffered to the modem processor when the application timer later expires. Ex-1401, 10:4-10.

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***.") (emphasis added)[3].

45. 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. Ex-1405, 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,

[3] All emphasis and annotations added unless otherwise noted.

the transceiver 110 ***may send a message to the network interface 112 requesting transmission of all queued packets***.")

C. Summary Of The Prosecution History of the '490 Patent

46. I understand that U.S. Application No. 14/568,694 ("the '694 application), which issued as the '490 patent, was filed on December 12, 2014. I also understand that 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-1403, 251 (April 27, 2015 Preliminary Amendment at 8). I summarize below relevant arguments and amendments the patentee made during prosecution to overcome some of the rejections and objections raised by the Examiner.

47. 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-1406) and U.S. Patent No. 6,021,264 ("Morita") (Ex-1407). *See* Ex-1403, 93-99

(June 10, 2016 Office Action at 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-1403, 33(August 24, 2016 Response to Office Action at 2). The Applicant similarly amended original independent claims 15, 20, and 24-28 to include a "trigger" limitation. *Id.*, 45 (Response at 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. . . .").

48. 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 <u>data,</u>

<u>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</u>.

Id., 39-40 (Response at 8-9).

49. 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.*, 44-46 (Response at 13-15).

50. The Examiner subsequently allowed the claims as amended.[4]

VII. OVERVIEW OF THE PRIOR ART REFERENCES

51. In my opinion, there is nothing novel or inventive in claims 9, 11-13, 26, and 27 of the '490 patent. The concepts claimed by the '490 patent were well known long before the '490 patent application was filed. In fact, the primary prior

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

art references discussed in this declaration were not before the Examiner during prosecution and in combination disclose all the elements that the Examiner found to be missing from the prior art of record. Below I provide brief summaries of three key references—Heinrich, Balasubramanian, and Tsai.

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

52. I understand that because U.S. Patent No. 9,329,671 to Heinrich *et al.* (Ex-1404) was filed January 29, 2013 and issued May 3, 2016, it is prior art at least under 35 U.S.C. § 102(a)(2).

53. 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:

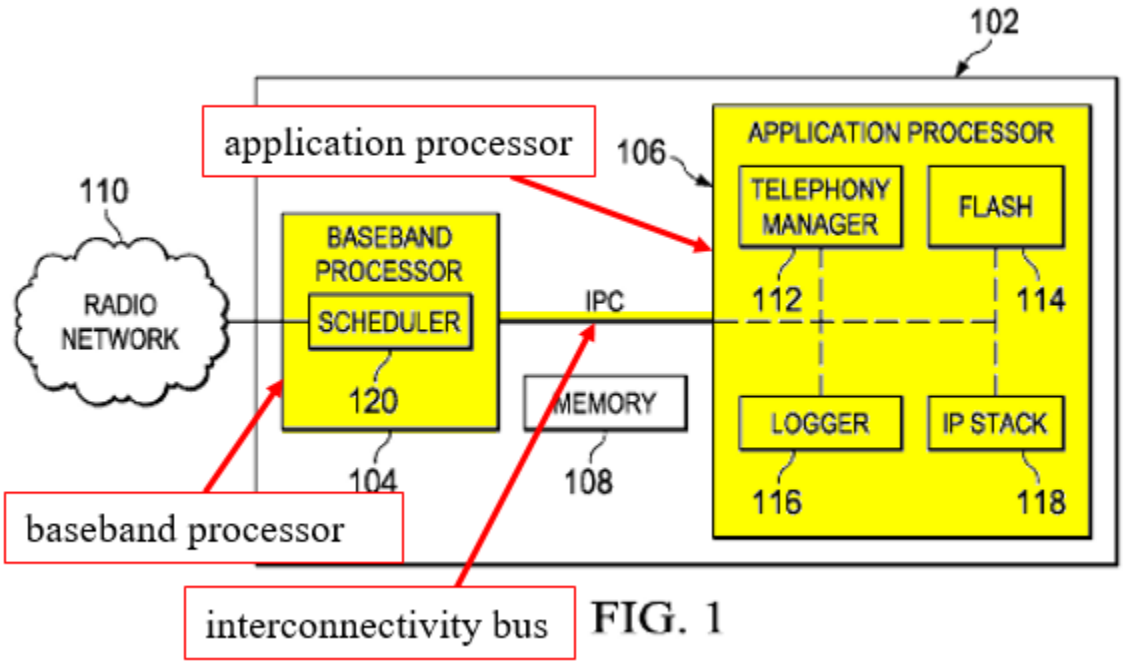


FIG. 1

Ex-1404, 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. . . .").

54. 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. Ex-1404, 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. . . ."); 9:13-16 ("***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. . . .").

55. 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.

56. 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***.").

57. 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

58. I understand that because U.S. Patent No. 8,160,000 to Balasubramanian (Ex-1405) was filed October 3, 2006 and issued April 17, 2012 it is prior art under at least 35 U.S.C. § 102(a)(1).

59. 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:

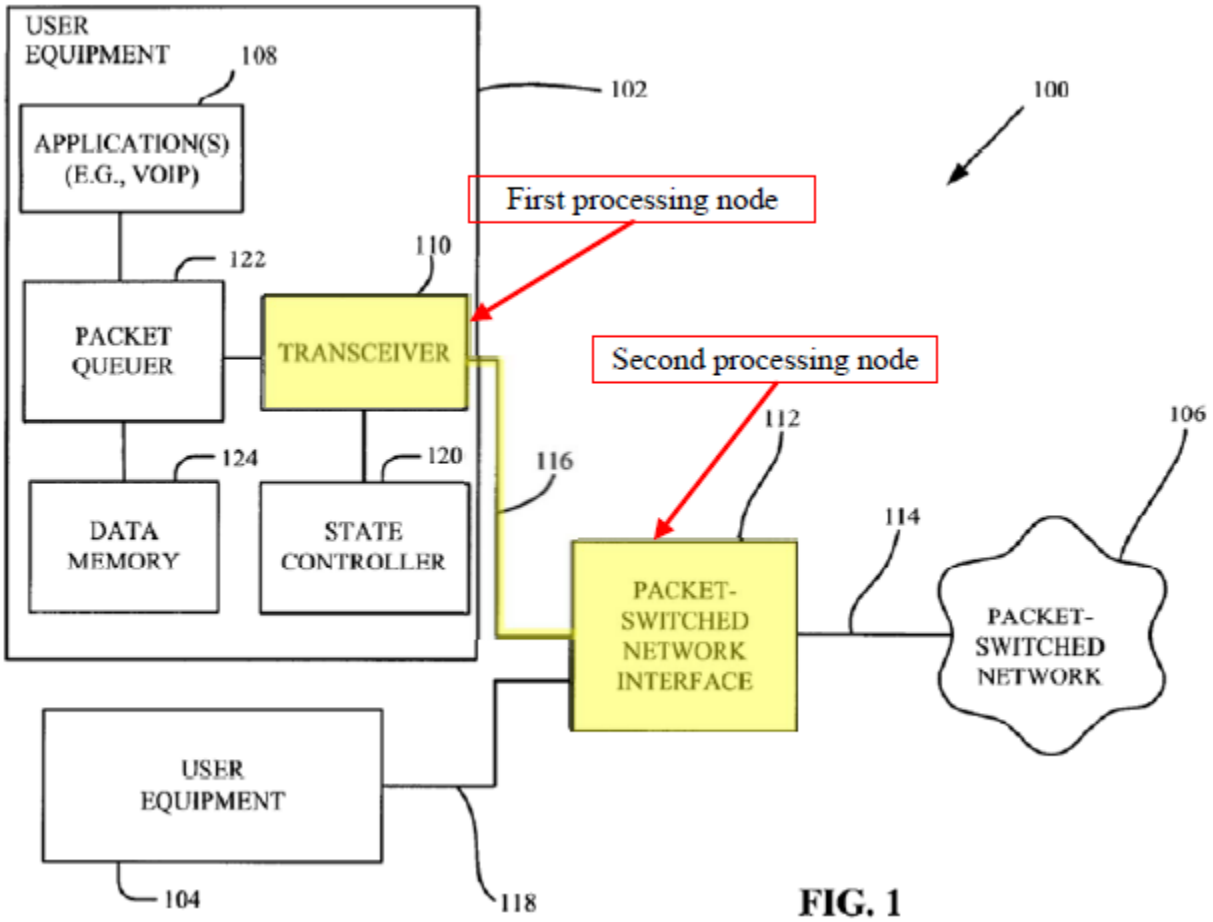


FIG. 1

Ex-1405, Fig. 1.

60. 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.

61. 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.").

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 suspend state into an active state reduces power consumption. Ex-1405, 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.").

63. 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.*, 5:66-6:4 ("[T]he user equipment 102 may include a packet queuer 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.").

64. 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). Ex-1405, 9:33-35 (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 ….").

VIII. CLAIM CONSTRUCTION

65. I have applied the "broadest reasonable interpretation" standard for the claim terms of the challenged claims. However, based on my reading of the '490 patent's specification and the ordinary meanings of the claim terms, the prior art teaches each claim limitation under any reasonable interpretation of the claim terms. My analysis is, therefore, not dependent on application of the "broadest reasonable interpretation" standard.

66. I understand that the Petitioner has set forth its proposed construction of a term of claim 1 of the '490 patent and its support for the construction. Those are copied below. I also understand that the remaining terms of claims 9, 11-13, 26, and 27 are readily understandable and I have given them their broadest reasonable interpretation in light of the specification as commonly understood by those of ordinary skill in the art.

A. "triggered by"

67. As used in the '490 patent, a person of ordinary skill 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-1401, 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* Ex-1401, claim 1. The prosecution history also supports this understanding of "triggered by." *See* Ex-1403, 45 (8/24/2016 Response at 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-1412 (*Heritage Dictionary*), 1444 ("trigger[:] . . . 1. to set off; initiate"); Ex-1413 (*Oxford Dictionary*), 1541

("trigger . . . cause to happen or exist"); Ex-1414 (*Merriam-Webster's Dictionary*), 1337 ("trigger[:] . . . to initiate, actuate, or set off by a trigger").

IX. LEVEL OF ORDINARY SKILL IN THE ART

68. I understand that the level of ordinary skill may be reflected by the prior art of record, and that a person of ordinary skill in the art to which the claimed subject matter pertains would have the capability of understanding the scientific and engineering principles applicable to the pertinent art. I understand that one of ordinary skill in the art has ordinary creativity, and is not a robot.

69. I understand there are multiple factors relevant to determining the level of ordinary skill in the art, including (1) the levels of education and experience of persons working in the field at the time of the invention, (2) the sophistication of the technology, (3) the types of problems encountered in the field; and (4) the prior art solutions to those problems. There are likely a wide range of educational backgrounds in the technology fields pertinent to the '490 patent.

70. A person of ordinary skill in the art at the time of the alleged invention 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. Additional graduate education could substitute for work experience, and additional work experience/training could substitute for formal education. Although my qualifications and experience exceed those of the hypothetical person of ordinary skill in the art defined above, my analysis and opinions regarding the '490 patent have been based on the perspective of a person of ordinary skill in the art at the time of invention.

X. SPECIFIC GROUNDS FOR CHALLENGE

71. The sections below demonstrate in detail how claims 9, 11-13, 26, and 27 of the '490 patent are not patentable over the prior art.

A. Ground 1: Claims 9, 11-13, 26, and 27 Are Rendered Obvious By Heinrich In View Of Balasubramanian

1. Claim 1

a) [1a] Preamble: "[a] mobile terminal"

72. The '490 patent describes the claimed "mobile terminal" as a smart phone, cell phone, tablet, or similar device. Ex-1401, 6:37-41 ("The mobile terminal 22 may be a smart phone, . . . a cellular telephone, a tablet, a laptop, or other mobile computing device.").

73. Heinrich discloses 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-1404, 4:21-23; *see also id.*, Fig. 1 (showing user device 102).

b) [1b] "a modem timer"

74. Claim 1 requires "a modem timer." Heinrich teaches 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-1404, 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. . . . ").

75. 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 determines when data held by the baseband processor 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.").

c) [1c] "a modem processor"

76. Heinrich discloses the "modem processor" of claim 1. For example, as shown in Figure 1, Heinrich discloses a mobile communication system that includes baseband processor 104:

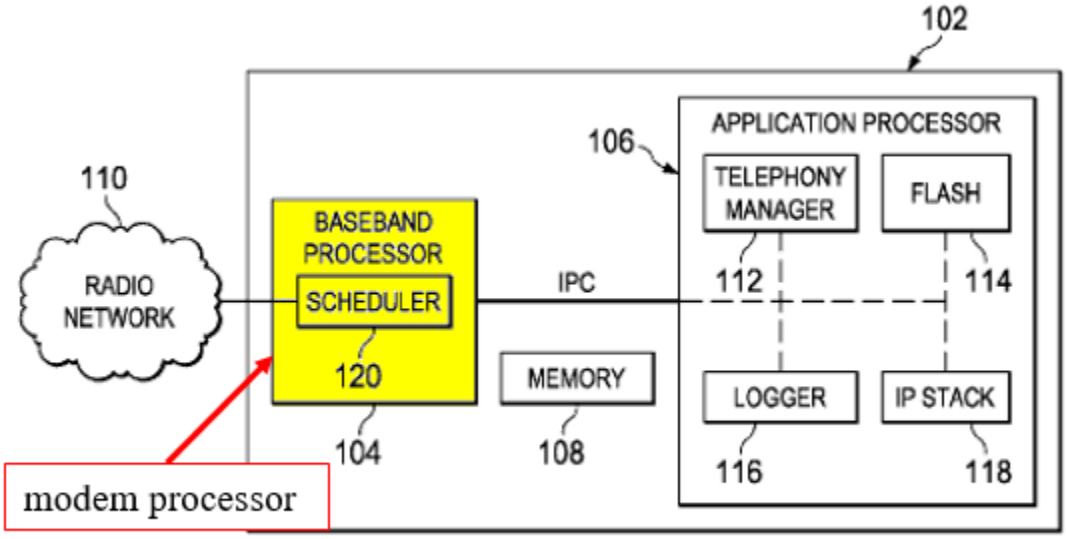


FIG. 1

Ex-1404, Fig. 1. Heinrich discloses that baseband processor 104 is a "modem" processor that communicates with a network. *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. . . . [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).

d) [1d] "the modem processor configured to hold modem processor to application processor data until expiration of the modem timer"

77. 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 modem timer. *Id.*, 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. 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) 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.").

78. Heinrich describes "baseband memory" where baseband data is buffered. Ex-1404, 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.").

79. It would have been obvious to a person of ordinary skill that the baseband processor in Heinrich can hold its 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 Preeti Ranjan Panda et al., *On-Chip vs. Off-Chip Memory: The Data Partitioning Problem in Embedded Processor-Based Systems*, 5 ACM Transactions on Design Automation of Electronic Systems 682, 683 (2000) ("Panda") (Ex-1411).

80. ***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-

1411, 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. . . .") (emphasis in original).

81. ***Second***, the data could be held on a separate or external chip—such as a memory chip. *Id.* ("[A]ccess to 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):

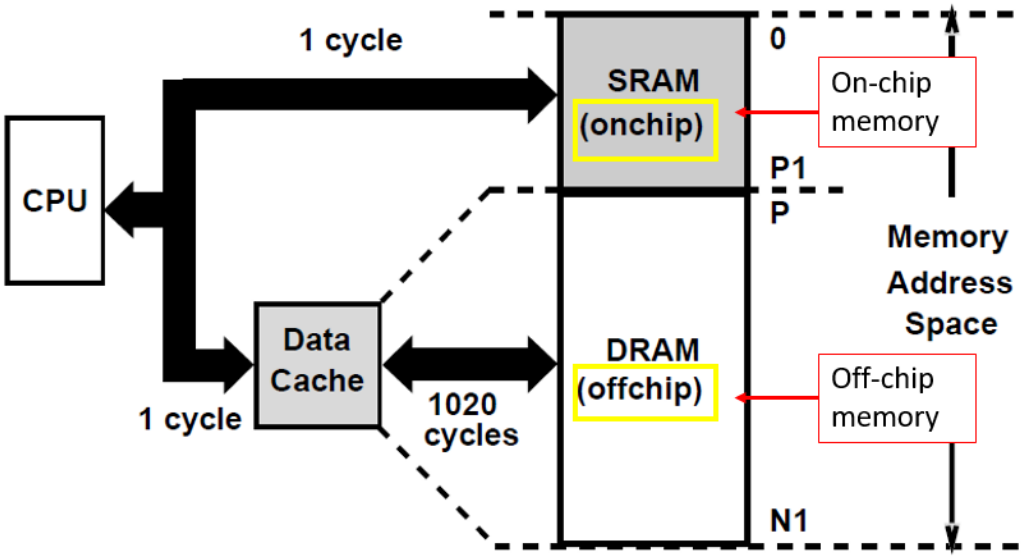


Fig. 2. Division of data address space between SRAM and DRAM.

82. A person of ordinary skill 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 person of ordinary skill would have reasonably expected the design to succeed.

e) [1e] "an application processor"

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

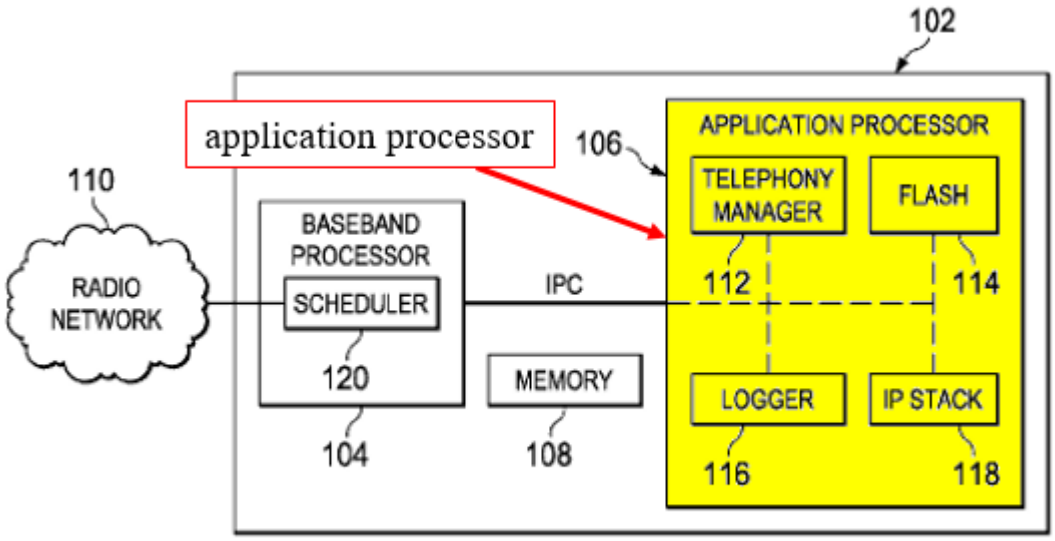


FIG. 1

Ex-1404, 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 Wi-Fi module, a GPS module, etc.").

f) [1f] "an interconnectivity bus communicatively coupling the application processor to the modem processor"

84. Heinrich discloses a bus labeled "IPC" ("inter processor communication") that communicatively couples the baseband processor 104—*i.e.*, modem processor—to the application processor 106:

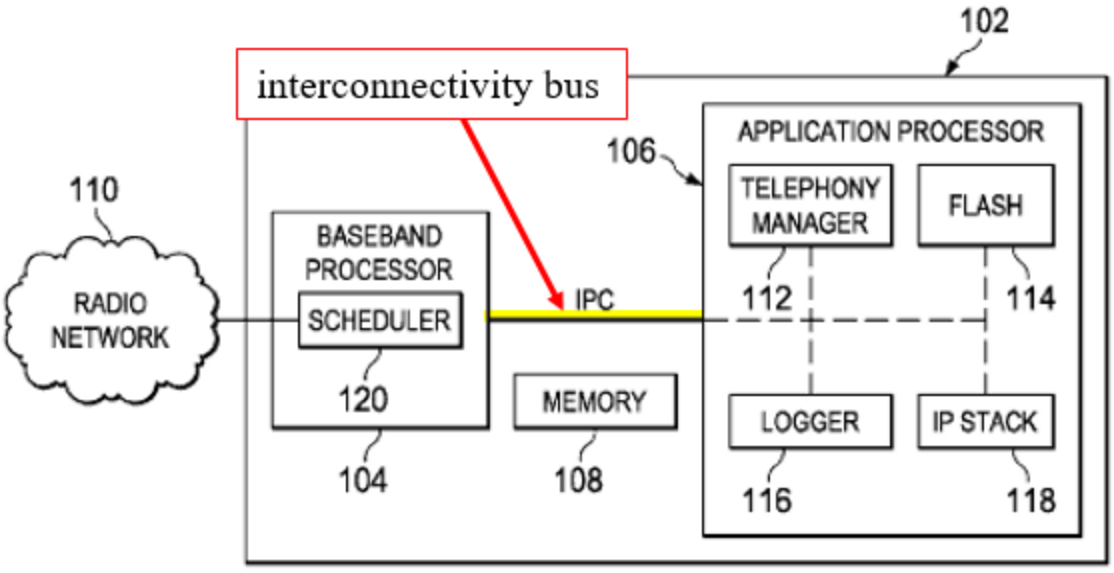


FIG. 1

Ex-1404, 4:44-46 ("There is a physical interface configured for communicating IPC activities between the baseband processor 104 and the application processor 106."), Fig. 1.

85. The IPC bus in Heinrich is a "physical interface" that can be any one of many industry-standard buses. *Id.*, 4:46-50 ("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.").

g) [1g] "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."

86. Claim [1g] requires "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."

87. ***"[T]he application processor configured to hold application processor to modem processor data":*** Heinrich renders obvious that the "application processor" is "configured to hold application processor to modem processor data." As I 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 [1d]. 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-1404, 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***.").

88. It would have been obvious to a person of ordinary skill to construct Heinrich's application processor to hold the data for the same reasons described in claim [1d]. *See supra* claim [1d]. While Heinrich does not expressly specify where such data is held, it would have been obvious to a person of ordinary skill

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.

89. As I explained above in claim limitation [1d], a person of ordinary skill 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-1411, 683, 686 (Fig. 2).

90. As with the baseband processor, a person of ordinary skill 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 [1d].

91. ***"[U]ntil triggered by receipt of the modem processor to application processor data from the modem processor through the interconnectivity bus"***: The combination of Balasubramanian and Heinrich discloses an application processor that holds data "until triggered by receipt of the modem processor to application processor data from the modem processor through the interconnectivity bus."

92. 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-1404, 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.").[5]

93. 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 ("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***

[5] The quoted portion of Heinrich describes the transmission of data from the baseband processor. However, as I explained above, Heinrich discloses that the same techniques can be used to control transmission of data from the application processor to the baseband processor. Ex-1404, 12:52-55.

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.

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):

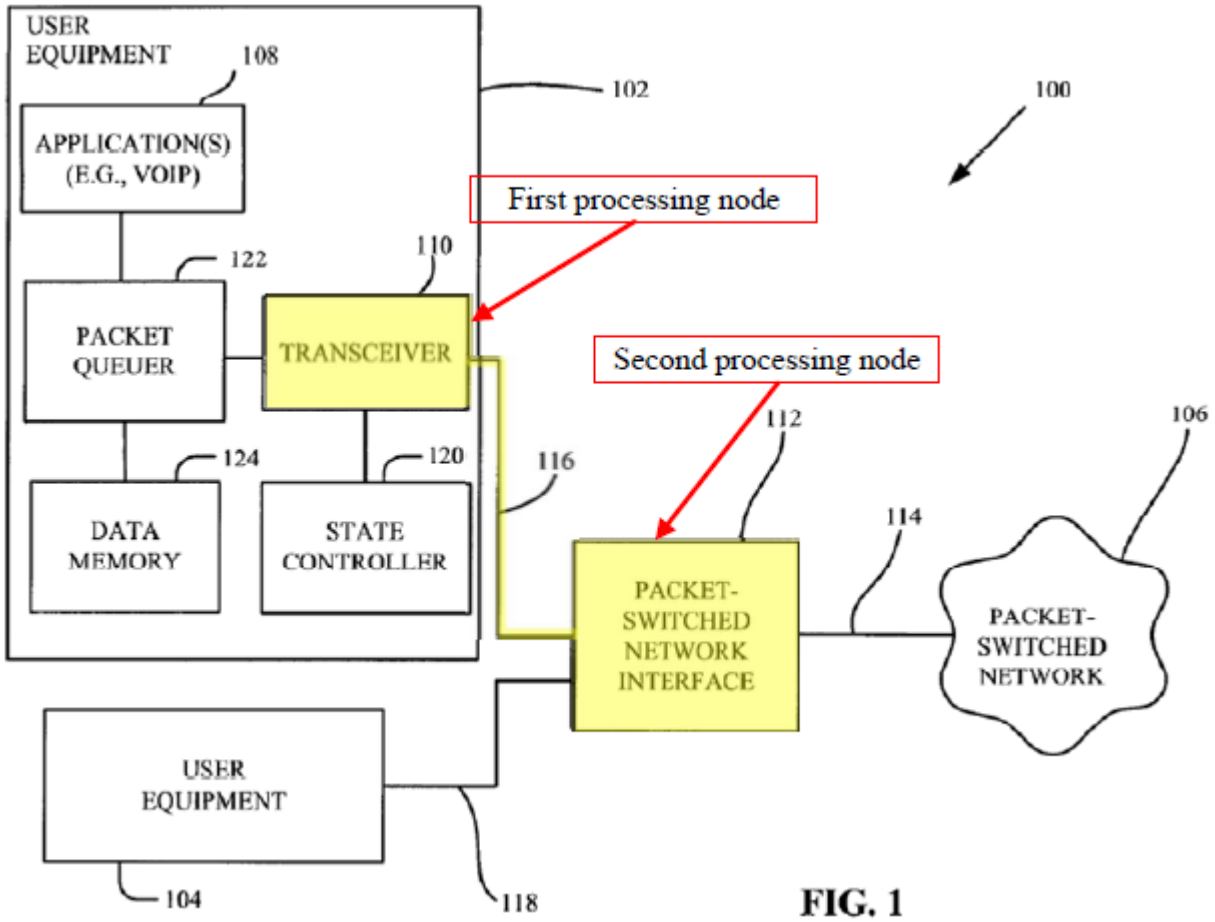


FIG. 1

Ex-1405, Fig. 1, 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.").

95. Figure 2 of Balasubramanian also discloses this feature. Consistent with the disclosure in Balasubramanian (*see* Ex-1405, 6:63-7:8), 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.

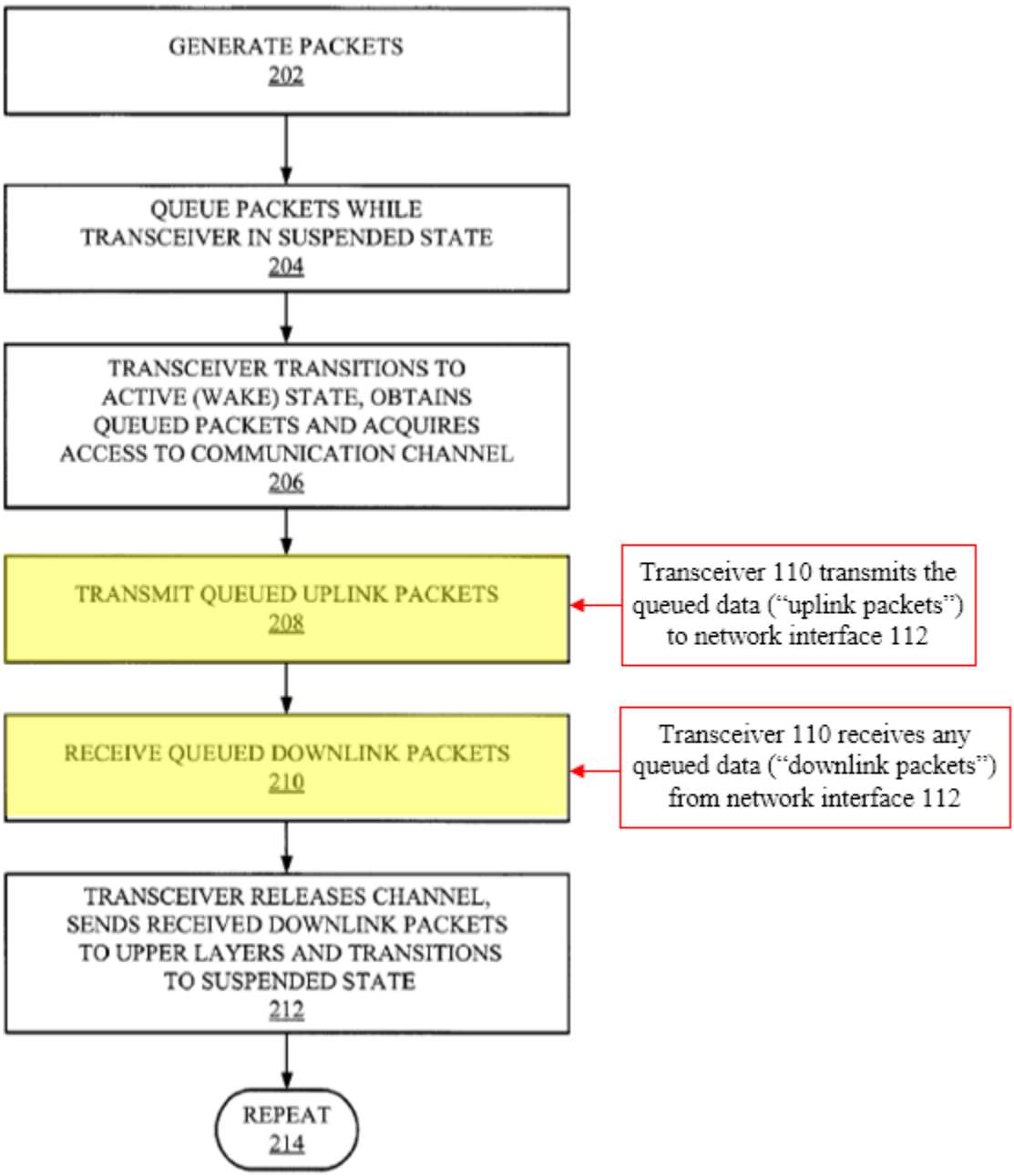


FIG. 2

Ex-1405, Fig. 2, 6:63-7:8.

96. 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 all 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 modem processor to application processor data from the modem processor through the interconnectivity bus."[6]

97. ***"[A]fter 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"***: The combination of Heinrich and Balasubramanian discloses that "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."

98. As I explained above, Heinrich discloses a modem processor that transmits held data over an interconnectivity bus to the application processor upon expiration of a timer. Heinrich also discloses that its application processor can

[6] This Petition sets forth reasons why a person of ordinary skill in the art would combine Heinrich and Balasubramanian in the section immediately below.

transmit held data to the modem processor upon determining that the modem processor is awake. Ex-1404, 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***."); *id.*, 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***."); 9:50-62 ("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***."). Heinrich further discloses transmitting data from the application processor to the modem processor through a bus. *Id.*, 4:44-46. While each data communication ("IPC activity") in Heinrich has its own timer, Heinrich discloses that all the held data in the modem processor is transmitted to the application processor upon expiration of any of the timers. *Id.*, 9:2-13 ("[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. . . . 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.").

99. Balasubramanian adds that receipt of data at a second processing node (network interface 112) from a first processing node (transceiver 110) "trigger[s]" the transmission of held data from the second processing node to the first processing node, meaning that the second processing node sends its data to the first processing node after receiving data from the first processing node and responsive to receiving data from the first processing node. Ex-1405, 7:4-8 ("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."); *id.*, 6:5-10 ("Similarly, the network interface **112** may be adapted to queue packets destined for the user equipment **102** when the transceiver **110** is in a suspended state. In this case, when the transceiver 110 is transitioned to an active state, the network interface 112 may send the queued packets to the transceiver 110.").

100. Although Balasubramanian does not explicitly disclose a "modem processor" coupled to an "application processor" by an interconnectivity bus, it does disclose a first processing node (*e.g.*, transceiver 110) coupled to a second processing node (*e.g.*, network interface 112) by an interconnectivity bus (communication link 116). *See* Ex-1405, Fig. 1. Balasubramanian further discloses transmitting data from a second processing node (*e.g.*, the network interface 112) to a first processing node (*e.g.*, transceiver 110) upon receipt of all the data from the first processing node. *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."), Fig. 2. Finally, Balasubramanian teaches that its first processing node (*e.g.*, transceiver 110) can transmit its held data to the second processing node (*e.g.*, network interface 112) upon the expiration of a timer. Ex-1405, 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***.").

101. In other words, Heinrich teaches a modem processor that transmits all buffered data to an application processor upon expiration of a timer, and an application processor that transmits all buffered data to the modem processor upon determining that the modem processor is awake. Balasubramanian teaches a first processing node (*e.g.*, transceiver 110) that transmits data to a second processing node (*e.g.*, network interface 112) upon the expiration of a timer, and a second processing node that uses the receipt of the data from the first processing node as a trigger to send data back to the first processing node. Thus, Balasubramanian can be viewed as a particular application of the idea in Heinrich, because the trigger taught by Balasubramanian (the receipt of data from the first processing node) is just a specific way of determining that the first processing node is awake. Thus, the combination of Heinrich and Balasubramanian would result in a system in which a modem processor transmits all buffered data to an application processor upon expiration of a timer, and an application processor that transmits all buffered data to the modem processor upon determining, based on the receipt of data from

the modem processor, that the modem processor is awake. In other words, the combination would create a system in which the application processor is triggered to send its held data to the modem processor through an interconnectivity bus, in response to the receipt of data from the modem processor through the interconnectivity bus.

102. ***Combination of Heinrich and Balasubramanian:*** It would have been obvious for a person of ordinary skill to substitute the "trigger" scheme from Balasubramanian into the processor-to-processor communication system of Heinrich (which includes a "modem processor" and an "application processor") in order to form the claimed combination. For example, as set forth above, Heinrich discloses the "modem processor," "modem timer," "application processor," and "interconnectivity bus" limitations from claim 1. Balasubramanian teaches a very similar architecture as Heinrich—including a first processing node (transceiver 110), a timer (Ex-1405, 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 1, 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.

103. ***Motivation to Combine:*** A person of ordinary skill would have been motivated to combine Balasubramanian's "trigger" scheme with the processor-to-processor communication system of Heinrich for at least four reasons.

104. ***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-1404, 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-1405, 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."). 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

to look to Balasubramanian when considering the power consumption issues of Heinrich.

105. ***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 would have been motivated to use the techniques taught by Balasubramanian to address the same issues described in Heinrich.

106. ***Third***, a person of ordinary skill 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-1404, 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 person of ordinary skill 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-1405, 7:24-28.

107. ***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-1404, 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 the 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 person of ordinary skill 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* Ex-1404, 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).

108. ***Reasonable Expectation of Success:*** A person of ordinary skill would have had a reasonable expectation of success in combining

Balasubramanian's "trigger" scheme with the processor-to-processor communications system of Heinrich.

109. 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.[7] Ex-1404, 9:5-14 ("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 aggregated IPC activities

[7] While Heinrich focuses mostly on transmissions from the baseband processor to the application processor, Heinrich discloses that the same techniques can be used for transmissions from the application processor to the baseband processor. Ex-1404, 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."). Thus, where Heinrich discloses a technique where a mobile processor can transmit held data to an application upon the occurrence of a trigger or event, it discloses that same feature for an application processor.

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.*, 10:24-31). Thus, Heinrich discloses various techniques that trigger the transmission of data from one processor to another processor.

110. A person of ordinary skill 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 person of ordinary skill would therefore understand that using Balasubramanian's trigger scheme would work as expected in the processor-to-processor communications system of Heinrich. 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-1404, 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 person of ordinary skill would recognize that one such "means" is the trigger scheme described in Balasubramanian.

2. Claim 9

111. Claim 9 depends from claim 1, which as described *supra* in section X.A.1 is obvious over Heinrich in view of Balasubramanian. Heinrich renders obvious the limitation of claim 9, which recites that "the application processor comprises the modem timer."

112. As described in Section X.A.1.b), the "lazy timer" of Heinrich is a "modem timer" because it determines when data held by the baseband processor is sent to the application processor. Ex-1404, 9:2-6; *see also id.*, Fig. 1, 7:10-11. Heinrich further teaches that the disclosed "lazy timers" may be allocated by a "scheduler." *Id*., 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.").

113. In addition, Heinrich teaches that a single "scheduler" can control the scheduling of data transmissions in both directions between the baseband and application processors. *Id.*, 7:19-21 ("The scheduler 120 may control the scheduling of IPC activities in both directions between the processors 104 and 106."); 12:59-64. And, Heinrich teaches that a scheduler can be implemented on the baseband processor 104 or the application processor 106. *Id.*, 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.").

114. While Heinrich does not explicitly disclose a scheduler on an application processor that controls the scheduling of data transmissions in both directions between processors 104 and 106, such an arrangement would have been obvious to a person of ordinary skill in the art in light of the above disclosures of Heinrich. As explained above, Heinrich teaches that a scheduler could be implemented on an application processor, and that a single scheduler can control the scheduling of data transmissions in both directions. It would have been obvious to a person of ordinary skill in the art to combine both concepts together in order to implement a scheduler on an application processor that controls the scheduling of data transmissions in both directions. Such a scheduler would allocate a "lazy timer" that controls when data is sent from the baseband processor to the application processor (*i.e.*, a modem timer).

115. A person of ordinary skill in the art would have been motivated to use a single scheduler rather than two schedulers so as to simplify the design of the mobile device, and a person of ordinary skill in the art would have been motivated to place that scheduler on the application processor rather than the baseband processor, because there are only a handful of available placement options (on the

baseband processor, on the application processor, or as a standalone chip), and the application processor has ample processing power to devote to such a task. A person of ordinary skill in the art would have had a reasonable expectation of success to modify Heinrich in this way, given that the combination is a combination of two simple concepts already taught in Heinrich.

3. Claim 12

116. Claim 11 depends from claim 1, which as described *supra* in Section X.A.1 is obvious over Heinrich in view of Balasubramanian. Heinrich in view of Balasubramanian renders obvious the limitation of claim 12, which recites "a packet number limit counter associated with the modem processor, the modem processor configured to send data to the application processor if a threshold associated with the packet number limit counter is exceeded."

117. As described *supra* in Sections X.A.1.d) and X.A.1.g), Heinrich teaches a baseband processor sending data to an application processor upon the expiration of a lazy timer associated with the baseband processor. Heinrich arguably does not disclose sending data to the application processor "if a threshold associated with the packet number limit counter is exceeded." Balasubramanian discloses "a packet number limit counter" and sending data "if a threshold associated with the packet number limit counter is exceeded."

118. Balasubramanian discloses a "counter" that tracks a number of packets to queue before sending the packets. Ex-1405, 2:5-8 ("[P]ackets may be queued until the designated number of packets have been queued. At this point, the apparatus may transition to a wake state so that the queued packets may be transmitted as a group."; 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."); 9:8-9 ("[A]configurable number of packets may be queued at a time.").

119. 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 a threshold associated with the packet number limit counter is exceeded." 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. 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. 1404, 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***."). In addition to the reasons provided above for claim [1g],

a person of ordinary skill in the art would have been motivated and had a reasonable expectation of success to modify Heinrich to send data based on reaching a packet count limit instead of or in addition to sending data based on the expiration of a timer given Heinrich's recognition of the overflow risk. *See* Section X.A.1.g). Further, 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. A person of ordinary skill in the art 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.

4. Claim 11

120. Claim 11 depends from claim 1, which as described *supra* in Section X.A.1 is obvious over Heinrich in view of Balasubramanian. Heinrich in view of Balasubramanian renders obvious the limitation of claim 11, which recites "a byte accumulation limit counter associated with the modem processor, the modem processor configured to send data to the application processor if a threshold associated with the byte accumulation limit counter is exceeded."

121. As described above with respect to claim 12, Heinrich in view of Balasubramanian renders obvious the additional limitation of a modem processor that sends data to an application processor if a threshold associated with the packet number limit counter is exceeded.

122. For the same reasons set forth supra in Section IX.A.3, a person of ordinary skill in the art would also have found it obvious for the modem processor to send data to the application processor if a threshold associated with a byte accumulation limit counter is exceeded. Including a byte counter into Heinrich's system, rather than a packet counter, would have provided an alternative or additional guard against overfilling the buffer in a system supporting 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 person of ordinary skill in the art 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. Further, expressing a buffer size in bytes rather than packets 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, or group of bits and bytes) and Heinrich expressly recognized that data could be expressed in megabytes. *See* Ex-1404, 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"). A person of ordinary skill in the art 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.

5. Claim 13

123. Claim 13 depends from claim 1, which as described *supra* in Section X.A.1 is obvious over Heinrich in view of Balasubramanian. Heinrich renders obvious the limitation of claim 13, which recites that the "modem processor is configured to determine if held data comprises a control packet and send such control packet before expiration of the modem timer."

124. Heinrich discloses that a "modem processor is configured to determine if … data comprises a control packet and send such control packet before expiration of the modem timer." As described above in claim [1d], Heinrich teaches accumulating certain data until the expiration of a "lazy timer." See Section IX.A.1.d. More specifically, Heinrich discloses that "non real-time sensitive activities" are accumulated until expiration of a lazy timer, whereas real-time sensitive activities are urgent and should not be delayed. Ex-1404, 8:14-20.

Heinrich also teaches a scheduler in the baseband processor determining whether IPC activities are real time sensitive and sending the real time sensitive activities prior the expiration of a lazy timer. *Id.*, 9:33-40 ("The procrastination property of the scheduler 120 as described in the example given above helps to synchronize non real-time sensitive IPC activities, however it does not synchronize these IPC activities with other IPC activities that are real-time sensitive, i.e. that can't wait before being scheduled. In step S302 some of the IPC activities are determined as being real-time sensitive, and those IPC activities are communicated to the application processor 106 without delay."); *id.*, FIG. 1. Heinrich further teaches that IPC activities include "control information," which Heinrich describes as activities associated with "initiating a voice call between the user device and another node of the radio network." *Id.*, 5:2-7. Heinrich also refers to "IPC activities" as "packets." *See id.*, 13:61-62 ("Time to set IPC (USB) interface to sleep mode after last IPC packet: 1 s"). A person of ordinary skill in the art would have understood that "control information" would be a form of "real-time sensitive" data as voice data is usually processed without delay. A person of ordinary skill in the art would have also understood, given Heinrich's teaching that IPC activities can be in the form of packets, that control information could take the form of one or more control packets. Taken together, Heinrich teaches the

baseband processor determining whether data is a real-time control packet and sending the real-time control packet to the application processor prior to the expiration of a lazy timer.

125. Heinrich arguably does not explicitly disclose that the "control information" is "held." A person of ordinary skill in the art would have understood that even for "real-time" data, the scheduler in the baseband processor in Heinrich would have to buffer the data at least for as long as it took for the application processor to wake up. Heinrich teaches that the application processor must be in the awake mode to receive and process data from the baseband processor. Ex-1404, 1:55-2:8 ("For example, the Application Processor may operate in an ***awake mode*** in which it is ***able to process IPC activities that it receives from the baseband processor***. The Application Processor may alternatively operate in a ***sleep mode*** in which it ***does not process IPC activities*** . . . Every time a quantum of data is sent over the IPC from the baseband processor to the Application Processor, the Application Processor needs to be in, or enter into, a state which allows for that communication to happen. If the Application Processor is in the sleep mode when the IPC activity is initiated then it is "woken up", i.e. switched to operate in an awake mode in order to process the IPC activity."). Heinrich also explains that an application processor can take time to complete a transition from a

low power mode to an awake mode. Ex-1404, 14:11-17 ("Assuming a typical DRX7 (1.28 s paging cycle) let us initially consider that both the application processor (AP) 106 and the baseband processor (BB) 104 are in low power mode. ***T=0: first paging activity; BB wakes up to decode paging block***; BB sends logging data out; AP wakes; both AP and BB are awake; current˜=200 mA T=1 s: ***AP completes exit from low power***"). A person of ordinary skill in the art would have also understood that this temporary holding of data would have been consistent with Heinrich's teaching of sending real-time sensitive data without delay because Heinrich was not concerned with describing the lower level details of data processing—which include the general principle that data must be at least temporarily held in order to be processed.

6. Claim 26

a) [26a] Preamble: "[a] mobile terminal"

126. For the same reasons discussed above for claim [1a], Heinrich discloses the "mobile terminal" of claim 26. *See supra* Sections X.A.1.a), X.A.1.b).

b) [26b] "a modem byte accumulation limit counter"

127. For the same reasons discussed above for claim 11, which recites "a byte accumulation limit counter associated with the modem processor," Heinrich

renders obvious the "modem byte accumulation limit counter" of claim 26. *See supra* Sections X.A.3, X.A.1.b).

c) [26c] "a modem processor"

128. For the same reasons discussed above for claim [1c], Heinrich discloses the "modem processor" of claim 26. *See supra* Sections X.A.1.c), X.A.1.b).

d) [26d] "the modem processor configured to hold modem processor to application processor data until a predefined threshold of bytes has been reached by the modem byte accumulation limit counter"

129. For the same reasons discussed above for claims [1d] and 11, which recite "the modem processor configured to hold modem processor to application processor data until expiration of the modem timer" and "the modem processor configured to send data to the application processor if a threshold associated with the byte accumulation limit counter is exceeded," respectively, Heinrich renders obvious the "the modem processor configured to hold modem processor to application processor data until a predefined threshold of bytes has been reached by the modem byte accumulation limit counter" of claim 26. *See supra* Sections X.A.1.d), X.A.3, X.A.1.b).

e) [26e] "an application processor"

130. For the same reasons discussed above for claim [1e], Heinrich discloses the "application processor" of claim 26. *See supra* Sections X.A.1.e), X.A.1.b).

f) [26f] "an interconnectivity bus communicatively coupling the application processor to the modem processor"

131. For the same reasons discussed above for claim [1f], Heinrich discloses the "interconnectivity bus communicatively coupling the application processor to the modem processor" of claim 26. *See supra* Sections X.A.1.f), X.A.1.b).

g) [26g] "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.

132. For the same reasons discussed above for claim [1g], Heinrich in view of Balasubramanian renders obvious 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" of claim 26. *See supra* Sections X.A.1.g), X.A.1.b).

7. Claim 27

a) [27a] Preamble: "[a] mobile terminal"

133. For the same reasons discussed above for claim [1a], Heinrich discloses the "mobile terminal" of claim 26. *See supra* Sections X.A.1.a), X.A.1.b).

b) [27b] "a modem packet counter"

134. For the same reasons discussed above for claim 12, which recites "packet number limit counter associated with the modem processor," Heinrich in view of Balasubramanian renders obvious the "a modem packet counter" of claim 27. *See supra* Sections X.A.4, X.A.1.b).

c) [27c] "a modem processor"

135. For the same reasons discussed above for claim [1c], Heinrich discloses the "modem processor" of claim 27. *See supra* Sections X.A.1.c), X.A.1.b).

d) [27d] "the modem processor configured to hold modem processor to application processor data until a predefined threshold of packets has been reached by the modem packet counter"

136. For the same reasons discussed above for claims [1d] and 12, which recite "the modem processor configured to hold modem processor to application processor data until expiration of the modem timer" and "the modem processor configured to send data to the application processor if a threshold associated with the packet number limit counter is exceeded," respectively, Heinrich in view of Balasubramanian renders obvious the "the modem processor configured to hold modem processor to application processor data until a predefined threshold of packets has been reached by the modem packet counter" of claim 27. *See supra* Sections X.A.1.d), X.A.4, X.A.1.b).

e) [27e] "an application processor"

137. For the same reasons discussed above for claim [1e], Heinrich discloses the "application processor" of claim 27. *See supra* Sections X.A.1.e), X.A.1.b).

f) [27f] "an interconnectivity bus communicatively coupling the application processor to the modem processor"

138. For the same reasons discussed above for claim [1f], Heinrich discloses the "interconnectivity bus communicatively coupling the application

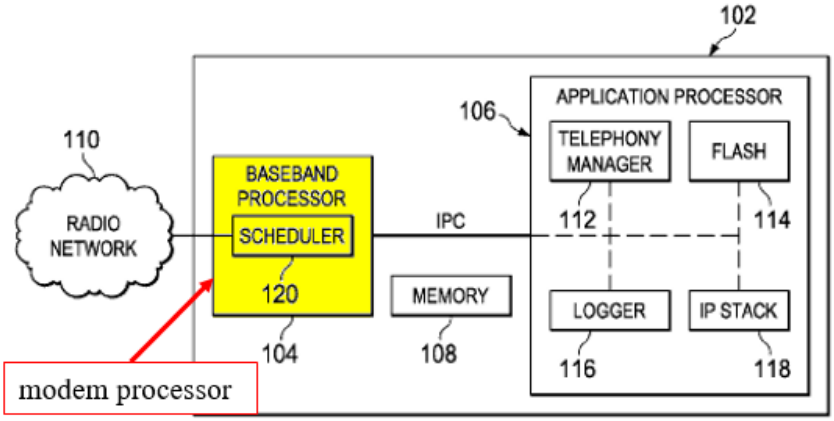
processor to the modem processor" of claim 27. *See supra* Sections X.A.1.f), X.A.1.b).

g) [27g] "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."

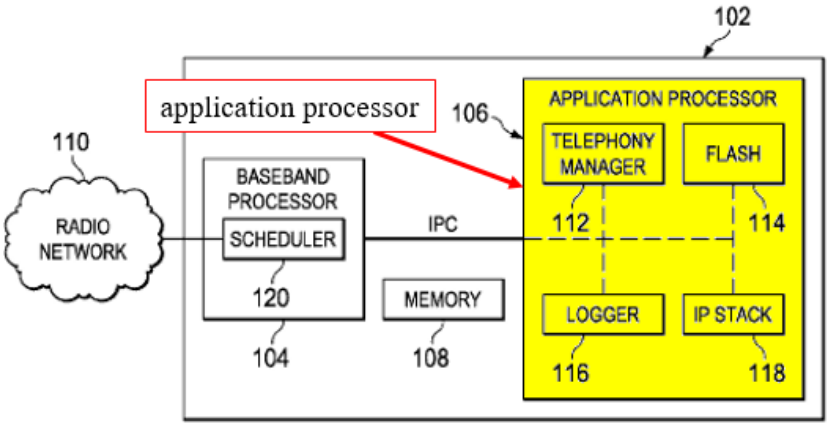
139. For the same reasons discussed above for claim [1g], Heinrich in view of Balasubramanian renders obvious the "application processor configured to hold application processor to modem processor data until triggered by receipt of the modem processor to application processor data from the modem processor 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" of claim 27. *See supra* Sections X.A.1.g), X.A.1.b).

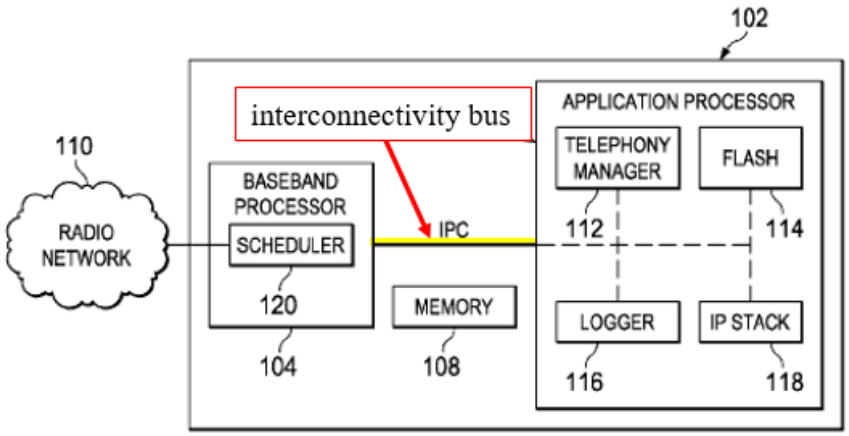
140. The chart below provides a summary of the disclosure of the references in the grounds discussed above and presented in the form of a claim chart.

Ground 1: Claims 9, 11-13, 26, 27 Are Rendered Obvious By Heinrich In View Of Balasubramanian	
Limitation	**Heinrich And Balasubramanian**
[1a] Preamble: A mobile terminal	Heinrich Ex-1404, 4:21-23 ("The user device 102 may be, for example, a ***mobile phone***, a tablet, a laptop computer or other embedded device able to connect to the network 110."). Ex-1404, Fig. 1.
[1b] a modem timer	Heinrich Ex-1404, 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. . . ."). Ex-1404, 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."). Ex-1404, Fig. 1.
[1c] a modem processor	Heinrich

Ground 1: Claims 9, 11-13, 26, 27 Are Rendered Obvious By Heinrich In View Of Balasubramanian	
Limitation	**Heinrich And Balasubramanian**
	 FIG. 1 Ex-1404, Fig. 1. Ex-1404, 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.").
[1d] the modem processor configured to hold modem processor to application processor data until expiration of the modem timer	<u>Heinrich</u> Ex-1404, 9:5-21 ("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. . . . ***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

Ground 1: Claims 9, 11-13, 26, 27 Are Rendered Obvious By Heinrich In View Of Balasubramanian	
Limitation	**Heinrich And Balasubramanian**
	processor 106 because the scheduler 120 can deduce that the application processor 106 is in the awake mode."). Ex-1404, 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."). Ex-1404, 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."). Ex-1404, 11:57-58 ("In order to delay file write accesses they are stored in a cache memory of the baseband subsystem."). Ex-1404, 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."). Ex-1404, Fig. 1. <u>Other Background References</u> Ex-1411, 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 . . ..

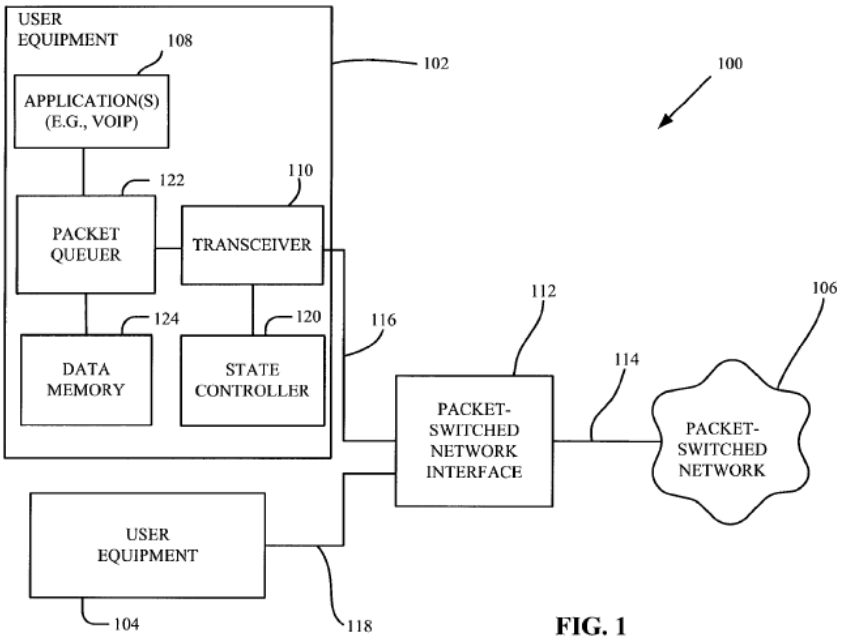
Ground 1: Claims 9, 11-13, 26, 27 Are Rendered Obvious By Heinrich In View Of Balasubramanian	
Limitation	**Heinrich And Balasubramanian**
	[I]t is possible to incorporate embedded DRAMs along with a processor core in the same chip . . ..”). Ex-1411, 683 (“[A]ccess to an off-chip memory (usually DRAM) requires relatively longer access times.”).
[1e] an application processor	Heinrich  FIG. 1 Ex-1404, Fig. 1. Ex-1404, 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.”).
[1f] an interconnectivity bus communicatively coupling the application	Heinrich

Ground 1: Claims 9, 11-13, 26, 27 Are Rendered Obvious By Heinrich In View Of Balasubramanian	
Limitation	**Heinrich And Balasubramanian**
processor to the modem processor	 FIG. 1 Ex-1404, Fig. 1. Ex-1404, 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."). Ex-1404, 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 (HIS), (iii) a Serial Peripheral Interface (SPI), or (iv) a shared memory.").
[1g] the application processor configured to hold application processor to modem processor data until	<u>Heinrich</u> Disclosure of Limitation: Ex-1404, 12:52-55 ("A scheduler may implement the ***same scheduling techniques*** as those described above, but

Ground 1: Claims 9, 11-13, 26, 27 Are Rendered Obvious By Heinrich In View Of Balasubramanian	
Limitation	**Heinrich And Balasubramanian**
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	configured to schedule IPC activities ***from the application processor 106 to the baseband processor 104***."). Ex-1404, 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***."). Ex-1404, 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.* at 7:7-27. Ex-1404, 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.* at 9:43-49. Ex-1404, 4:44-46 ("There is a physical interface configured for communicating IPC activities between the baseband processor 104 and the application processor 106."). Ex-1404, 9:2-14 ("[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. . . . 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.").

Ground 1: Claims 9, 11-13, 26, 27 Are Rendered Obvious By Heinrich In View Of Balasubramanian	
Limitation	**Heinrich And Balasubramanian**
	Ex-1404, Fig. 1. Support for Motivation to Combine: Ex-1404, 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-1404, 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."). Ex-1404, 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."). Ex-1404, 9:50-62 ("9:50-62 ("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

Ground 1: Claims 9, 11-13, 26, 27 Are Rendered Obvious By Heinrich In View Of Balasubramanian	
Limitation	**Heinrich And Balasubramanian**
	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."). Ex-1404, 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.* at 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 will be in the awake mode by determining that real-time sensitive IPC activities are being sent to the application processor, 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."). Balasubramanian

Ground 1: Claims 9, 11-13, 26, 27 Are Rendered Obvious By Heinrich In View Of Balasubramanian	
Limitation	**Heinrich And Balasubramanian**
	 FIG. 1 Ex-1405, Fig. 1. Ex-1405, 7:4-13 ("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. Alternatively, the transceiver 110 may send a message to the network interface 112 requesting transmission of all queued packets. . . ."). Ex-1405, 6:5-10 ("In this case, when the transceiver 110 is transitioned to an active state, the network interface 112 may send the queued packets to the transceiver 110."). Ex-1405, 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

Ground 1: Claims 9, 11-13, 26, 27 Are Rendered Obvious By Heinrich In View Of Balasubramanian	
Limitation	**Heinrich And Balasubramanian**
	achieved by rescheduling the packet traffic into groups of traffic." (emphasis added))
9. The mobile terminal of claim 1, wherein the application processor comprises the modem timer.	Heinrich Ex-1404, 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."); 12:59-64.
11. The mobile terminal of claim 1, further comprising a byte accumulation limit counter associated with the modem processor, the modem processor configured to send data to the application processor if a threshold associated with the byte accumulation limit counter is exceeded.	*See* Claim [1d], [1g], [12]. Heinrich Ex-1404, 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.") Ex-1404, 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.")
12. The mobile terminal of claim 1, further comprising a	*See* Claim [1d], [1g]. Heinrich

Ground 1: Claims 9, 11-13, 26, 27 Are Rendered Obvious By Heinrich In View Of Balasubramanian	
Limitation	**Heinrich And Balasubramanian**
packet number limit counter associated with the modem processor, the modem processor configured to send data to the application processor if a threshold associated with the packet number limit counter is exceeded.	*See* claim 1 (disclosing a baseband processor sending data to an application processor upon the expiration of a lazy timer associated with the baseband processor). Ex-1404, 13:61-62 ("Time to set IPC (USB) interface to sleep mode after last IPC packet: 1 s") Ex-1404, 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.") Balasubramanian Ex-1405, 12:3-13 ("Packets may be queued until a certain criterion is reached. As discussed above, the criterion may be based on a configuration variable such as a configurable amount of time, ***a configurable number of packets*** or some other suitable parameter. Again, the configuration variable may be provided via a signal 524 to a timer/counter 526 that may be used to determine when the criterion has been reached.") Ex-1405, 9:29-36 ("The device 402 may include an upper layer control module 430 that may provide functionality similar to the packet queuer 122 discussed above in conjunction with FIG. 1. As represented by block 308 in FIG. 3, the 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 to (e.g., send or provide access to) a component of the lower layers 410 (e.g., the media access controller 418)."). Ex-1405, 6:55-65 ("[O]nce the configurable amount of packets have been queued, the transceiver 110 transitions to an active (e.g., wake) state. . . . [T]he transceiver 110

Ground 1: Claims 9, 11-13, 26, 27 Are Rendered Obvious By Heinrich In View Of Balasubramanian	
Limitation	**Heinrich And Balasubramanian**
	then transmits the queued uplink packets over the communication link 116."); 9:8-9 ("[A]configurable number of packets may be queued at a time.").
13. The mobile terminal of claim 1, wherein the modem processor is configured to determine if held data comprises a control packet and send such control packet before expiration of the modem timer.	*See* claim [1d] Heinrich Ex-1404, 1:54-2:8 ("For example, the Application Processor may operate in an awake mode in which it is able to process IPC activities that it receives from the baseband processor. The Application Processor may alternatively operate in a sleep mode in which it does not process IPC activities…. Every time a quantum of data is sent over the IPC from the baseband processor to the Application Processor, the Application Processor needs to be in, or enter into, a state which allows for that communication to happen. If the Application Processor is in the sleep mode when the IPC activity is initiated then it is "woken up", i.e. switched to operate in an awake mode in order to process the IPC activity.") Ex-1404, 5:2-7 (control information (e.g. for initiating a voice call between the user device and another node of the radio network)) Ex-1404, 8:14-20 ("non real-time sensitive activities" are accumulated until expiration of a lazy timer, whereas real-time sensitive activities are urgent and should not be delayed) Ex-1404, 9:33-40 ("The procrastination property of the scheduler 120 as described in the example given above helps to synchronize non real-time sensitive IPC activities, however it does not synchronize these IPC activities with other IPC activities that are real-time sensitive, i.e. that can't wait before being scheduled. In step S302 ***some of***

Ground 1: Claims 9, 11-13, 26, 27 Are Rendered Obvious By Heinrich In View Of Balasubramanian	
Limitation	**Heinrich And Balasubramanian**
	the IPC activities are determined as being real-time sensitive, and those IPC activities are communicated to the application processor 106 without delay.") Ex-1404, 13:61-62 ("Time to set IPC (USB) interface to sleep mode after last IPC packet: 1 s") Ex-1404, 14:11-17 ("Assuming a typical DRX7 (1.28 s paging cycle) let us initially consider that both the application processor (AP) 106 and the baseband processor (BB) 104 are in low power mode. ***T=0: first paging activity; BB wakes up to decode paging block***; BB sends logging data out; AP wakes; both AP and BB are awake; current˜=200 mA T=1 s: ***AP completes exit from low power***") Ex-1404, FIG. 1.
[26a] A mobile terminal comprising:	*See* claim [1a].
[26b] a modem byte accumulation limit counter;	*See* claim 11.
[26c] a modem processor,	*See* claim [1c].
[26d] the modem processor configured to hold modem processor to application processor data until a predefined threshold of bytes	*See* claim [1d] and 11.

Ground 1: Claims 9, 11-13, 26, 27 Are Rendered Obvious By Heinrich In View Of Balasubramanian	
Limitation	**Heinrich And Balasubramanian**
has been reached by the modem byte accumulation limit counter;	
[26e] an application processor;	*See* claim [1e].
[26f] an interconnectivity bus communicatively coupling the application processor to the modem processor; and	*See* claim [1f].
[26g] 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	*See* claim [1g].

Ground 1: Claims 9, 11-13, 26, 27 Are Rendered Obvious By Heinrich In View Of Balasubramanian	
Limitation	**Heinrich And Balasubramanian**
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.	
[27a] A mobile terminal comprising:	*See* claim [1a].
[27b] a modem packet counter;	*See* claim 12.
[27c] a modem processor,	*See* claim [1c].
[27d] the modem processor configured to hold modem processor to application processor data until a predefined threshold of packets has been reached by	*See* claim [1d] and 12.

Ground 1: Claims 9, 11-13, 26, 27 Are Rendered Obvious By Heinrich In View Of Balasubramanian	
Limitation	**Heinrich And Balasubramanian**
the modem packet counter;	
[27e] an application processor;	*See* claim [1e].
[27f] an interconnectivity bus communicatively coupling the application processor to the modem processor; and	*See* claim [1f].
[27g] 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	*See* claim [1g].

Ground 1: Claims 9, 11-13, 26, 27 Are Rendered Obvious By Heinrich In View Of Balasubramanian	
Limitation	**Heinrich And Balasubramanian**
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.	

XI. AVAILABILITY FOR CROSS-EXAMINATION

141. In signing this declaration, I recognize that the declaration will be filed as evidence in a contested case before the Patent Trial and Appeal Board of the United States Patent and Trademark Office. I also recognize that I may be subject to cross-examination in the case and that cross-examination will take place within the United States. If cross-examination is required of me, I will appear for cross-examination within the United States during the time allotted for cross-examination.

XII. RIGHT TO SUPPLEMENT

142. I reserve the right to supplement my opinions in the future to respond to any arguments that the Patent Owner raises and to take into account new information as it becomes available to me.

XIII. JURAT

143. I declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the full knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1201 of Title 18 of the United States code.

Dated: 7/5/18

Bill Lin, Ph.D.

Exhibit A

BILL LIN
Curriculum Vitae
Tel: 858.531.2441
Email: billmbox@gmail.com

EDUCATION

- **Ph.D**, Electrical Engineering and Computer Sciences
  University of California, Berkeley, May 1991
- **Masters of Science**, Electrical Engineering and Computer Sciences
  University of California, Berkeley, May 1988
- **Bachelor of Science**, Electrical Engineering and Computer Sciences
  University of California, Berkeley, May 1985

PROFESSIONAL EXPERIENCE

1/1997 – present	**University of California, San Diego** Professor, Electrical and Computer Engineering Adjunct Professor, Computer Science and Engineering
2/1992 – 12/1996	**IMEC, Leuven, Belgium** Group Head, Systems Control and Communications Group

AREAS OF EXPERTISE

- All aspects of computer architecture and computer network problems, including the design of heterogeneous multicore processors, systems-on-chips, data networks, wireless communications and mobile computing systems, and multimedia and graphics systems.

PROFESSIONAL ACTIVITIES

- Served or serving as Editor on 3 ACM or IEEE journals, as General Chair on 4 ACM or IEEE conferences, on the Organizing or Steering Committees for 5 ACM or IEEE conferences, and on the Technical Program Committees of 44 ACM or IEEE conferences.

SELECTED PROFESSIONAL ACTIVITIES

- Associate Editor, ACM Transactions on Design Automation and Electronics Systems (TODAES), 2010-2015
- Editorial Board, International Journal of Embedded Systems, 2003-present
- General Chair, ACM/IEEE Symposium on Networks-on-Chips (NOCS), 2009
- General Chair, ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), 2010
- Chair of Steering Committee, ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), 2012-2017
- Guest Editor, IEEE Transactions on Network and Service Management (TNSM), 2012-2013
- General Co-Chair, ACM SIGMETRICS, 2015

PUBLICATIONS

- Published over 170 journal articles and conference papers in top-tier venues and publications.

PATENTS

- Mitigating Low-Rate Denial-of-Service Attacks in Packet-Switched Networks. United States Patent 8,443,444. Issued in May 14, 2013.
- Systems and Methods for Proactive Surge Protection. United States Patent 7,860,004. Issued in December 28, 2010.
- Method and Apparatus for Scan Block Caching. United States Patent 7,672,005. Issued in March 2, 2010.
- Design Environment and a Design Method for Hardware/Software Co-Design. United States Patent 5,870,588. Issued in February 9, 1999.
- System and Method for Generating a Hazard-Free Asynchronous Circuit. United States Patent 5,748,487. Issued in May 5, 1998.

EXPERT WITNESS HISTORY

I have served as an expert witness on 21 cases (most cases involved multiple patents in suit), of which I wrote expert reports or declarations for 14 cases, analyzed source code in 10 cases, was deposed 11 times, and testified at trial 2 times. The specific cases and services provided are as follows:

1. Vizio, Inc. v. LG Electronics, Inc. and LG Electronics USA Inc., Civil Action No. 09-cv-1481-BEL
    - Expert witness on behalf of Vizio, Inc.
    - Law firm: Jones Day.
    - Services provided: Provided expert report, analyzed source code.
    - Technologies: Digital TV.

2. Vizio Inc. v. LG Electronics, Inc. and LG Electronics USA Inc., In the Matter of CERTAIN FLAT PANEL DIGITAL TELEVISIONS AND COMPONENTS THEREOF, U.S. International Trade Commission Inv. No. 337-TA-733
    - Expert witness on behalf of Vizio, Inc.
    - Law firm: Jones Day.
    - Services provided: Provided expert report, analyzed source code.
    - Technologies: Digital TV.

3. Vizio Inc. v. Coby Electronics Corp., Curtis International Ltd., E&S International Enterprises, Inc., MStar Semiconductor, Inc., ON Corp US, Inc., Renesas Electronics Corporation, Japan, Renesas Electronics America, Inc., Sceptre, Inc., and Westinghouse Digital, LLC, In the Matter of CERTAIN DIGITAL TELEVISIONS AND COMPONENTS THEREOF, U.S. International Trade Commission Inv. No. 337-TA-789
    - Expert witness on behalf of Vizio, Inc.
    - Law firm: Jones Day.
    - Services provided: Provided expert reports, was deposed, analyzed source code.
    - Technologies: Digital TV.

4. Linex Technologies Inc. v. Hewlett-Packard Company, Apple Inc., Aruba Networks, Inc., Meru Networks, and Ruckus Wireless, In the Matter of CERTAIN WIRELESS COMMUNICATION DEVICES AND SYSTEMS, COMPONENTS THEREOF, AND PRODUCTS CONTAINING SAME, U.S. International Trade Commission Inv. No. 337-TA-775
    - Expert witness on behalf of Hewlett-Packard Company, Apple Inc., Aruba Networks, Inc., Meru Networks, and Ruckus Wireless
    - Law firms: Covington & Burling, K&L Gates, Morrison & Foerster, Lewis Roca & Rothgerber.
    - Services provided: Provided expert reports, was deposed, analyzed source code.
    - Technologies: WiFi networks.

5. MOSAID Technologies, Inc. v. Dell, Inc., Qualcomm Atheros, Inc., et al., Case No. 2:11-cv-00179-TJW
    - Expert witness on behalf of Qualcomm Atheros, Inc.
    - Law Firm: McDermott Will & Emory.
    - Services provided: Provided expert analysis.
    - Technologies: WiFi networks.

6. Hitachi Consumer Electronics Co. Ltd., et al. v. Top Victory Electronics (Taiwan) Co. Ltd., et al., Civil Action No. 2:10-cv-260-JRG.
    - Expert witness on behalf of Vizio, Inc.
    - Law Firm: Akin Gump Strauss Hauer & Feld
    - Services provided: Provided expert reports, analyzed source code.
    - Technologies: Digital TV.

7. United States International Trade Commission (USITC) Investigation Number: 337-TA-853 and Technology Properties Limited, LLC v. Kyocera Corporation and Kyocera Communciations, Inc. in the Northern District of California, Case No. 3:12-cv-03860-JSC
    - Expert witness on behalf of Kyocera Corporation and Kyocera Communciations.
    - Law firm: Morrison & Foerster.
    - Services provided: Provided expert analysis.
    - Technologies: Cell phones, microprocessors.

8. Netgear, Inc. v. Ruckus Wireless, Inc., Case No. 1:10-cv-00999-SLR / D. Del.
    - Expert witness on behalf of Ruckus Wireless, Inc.
    - Law firms: Lewis Roca & Rothgerber, Orrick.
    - Services provided: Provided expert report, analyzed source code, was deposed, and testified at trial.
    - Technologies: WiFi networks.

9. Sonics, Inc. v. Arteris, Inc., Case No. 11 CV-05311 SBA, and Arteris S.A.S. v. Sonics, Inc., Case No. C 12-00434 (WHA)
    - Expert witness on behalf of Arteris, Inc. and Arteris S.A.S.
    - Law firm: DLA Piper.
    - Services provided: Retained as an expert.
    - Technologies: Systems-on-Chips (SoCs).

10. Linex Technologies Inc. v. Hewlett-Packard Company, Apple Inc., Aruba Networks, Inc., Meru Networks, and Ruckus Wireless, U.S. District Court, Northern District of California, C 13-00159 CW
    - Expert witness on behalf of Hewlett-Packard Company, Apple Inc., Aruba Networks, Inc., Meru Networks, and Ruckus Wireless
    - Law firms: Covington & Burling, Milbank, K&L Gates, Morrison & Foerster, Lewis Roca & Rothgerber.
    - Services provided: Provided expert reports, analyzed source code.
    - Technologies: WiFi networks.

11. Frontier Communications Corp. v. Google, C.A. No. 10-545 (D. Del)
    - Expert witness on behalf of Frontier Communications Corp.
    - Law firm: Steese, Evans & Frankel
    - Services provided: Provided expert analysis, wrote declarations, analyzed source code.
    - Technologies: Voice over IP.

12. U.S. Ethernet Innovations LLC v. Acer, Inc. et al., Case No. 10-cv-3724 (N.D. Cal.)
    - Expert witness on behalf of Qualcomm Atheros, Inc., Sigma Designs, Inc., and AT&T Services, Inc.
    - Law firms: Reed Smith LLP, Pillsbury Winthrop Shaw Pittman LLP, and Vinson & Elkins LLP.
    - Services provided: Provided expert reports, was deposed, analyzed source code.
    - Technologies: Intelligent network interfaces, data center networks.

13. Inter Partes Review of a U.S. Patent.
    - Expert witness on behalf of Netgear, Inc. and Belkin International, Inc.
    - Law firm: Reed Smith LLP.
    - Services provided: Provided declaration for IPR petition.
    - Technologies: Wireless networks.

14. Inter Partes Review of U.S. Patents.
    - Expert witness on behalf of Arista Networks.
    - Law firm: Fish & Richardson.
    - Services provided: Provided declarations for IPR petitions, deposed 3 times.
    - Technologies: Network routers, data center networks.

15. Invalidity Expert Witness.
    - Expert witness on behalf of Intel Corp.
    - Law firm: Wilmer Cutler Pickering Hale and Dorr LLP.
    - Services provided: Provided expert analysis.
    - Technologies: WiFi networks, bluetooth networks.

16. Parthenon Unified Memory Architecture LLC v. HTC Corporation et al., Case No. 2:14-CV-690-JRG-RSP (E. D. Texas).
    - Expert witness on behalf of HTC Corporation and HTC America, Inc.
    - Law firm: Sidley Austin LLP.
    - Services provided: Retained as an expert.
    - Technologies: Memory architectures.

17. System Architecture Information Technology dba SAI Technology, Inc. v. Qualcomm Inc.
    - Expert witness on behalf of Qualcomm Inc.
    - Law firm: McDermott Will & Emory.
    - Services provided: Retained as an expert, analyzed source code, was deposed, and testified at trial.
    - Technologies: WiFi networks.

18. Invalidity Expert Witness.
    - Expert witness on behalf of Sandvine Corp.
    - Law firm: Erise IP.
    - Services provided: Provided declarations.
    - Technologies: Network monitoring and measurements.

19. Inter Partes Review of U.S. Patents.
    - Expert witness on behalf of Intel Corp., Cavium Inc., and Dell Inc.
    - Law firms: Weil, Gotshal & Manges LLP, Duane Morris LLP, Alston & Bird LLP.
    - Services provided: Provided declarations for IPR petitions, deposed 2 times.
    - Technologies: Intelligent network interfaces, data center networks.

20. Expert Witness.
    - Law firm: Covington & Burling.
    - Services provided: Retained as an expert.
    - Technologies: WiFi networks.

21. Qualcomm Inc. v. Apple Inc., Inv. No. 337-TA-1065 (ITC), and Civ. Action No. 3:17-cv-01375-JAH-MDD (S.D. Cal.)
    - Expert witness on behalf of Apple Inc. and Intel Corp.
    - Law firm: Wilmer Cutler Pickering Hale and Dorr LLP.
    - Services provided: Provided expert reports/declarations, was deposed.
    - Technologies: Wireless systems, computer architecture.