DOCKET NO.: 0107131-00645US1
Filed on behalf of Intel Corporation
By: John V. Hobgood, Reg. No. 61,540
Richard Goldenberg, Reg. No. 38,895
Dominic Massa, Reg. No. 44,905
Wilmer Cutler Pickering Hale and Dorr LLP
60 State Street
Boston, MA 02109
Email: John.Hobgood@wilmerhale.com
Richard.Goldenberg@wilmerhale.com

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

Intel Corporation
Petitioner

v.

VLSI Technology LLC
Patent Owner

Case IPR2019-01192

DECLARATION OF DR. CARL SECHEN
U.S. PATENT NO. 7,523,331
CLAIM 7

TABLE OF CONTENTS

Page

I declare as follows:

I. INTRODUCTION

1. My name is Dr. Carl Sechen.

2. I earned a B.E.E. in Electrical Engineering from the University of Minnesota in 1975, followed by a M.S. in Electrical Engineering from the Massachusetts Institute of Technology in 1977. I earned a Ph.D. in Electrical Engineering from the University of California, Berkeley in 1986.

3. I have been a Professor of Electrical Engineering for 33 years. Since August 15, 2005, I have been a Professor of Electrical and Computer Engineering at the University of Texas at Dallas. From July 1992 to August 14, 2005, I served as a Professor of Electrical Engineering at the University of Washington. From July 1986 through June 1992, I served as an Assistant Professor and then Associate Professor of Electrical Engineering at Yale University.

4. Over these years my research has focused on the design and computer-aided design of digital integrated circuits, including computer architecture and the design of dynamic random-access memory ("DRAM") and static random-access memory (SRAM) modules. I have taught numerous students how to design DRAM and SRAM memories as part of the design courses that I have offered.

5. As a professor, I have developed and taught numerous courses, in particular, several courses that teach digital integrated circuit design and memory design in great detail. I have taught these courses continuously for the past 24 years. I have taught digital integrated circuit design and memory design to undergraduate and graduate students at the University of Washington and at the University of Texas at Dallas since 1995.

6. I have also been involved in numerous research projects on digital integrated circuit design and memory design. I have taught numerous graduate researchers how to design digital integrated circuits, including memories. Most of these had applications in computer architecture. I have also taught several Ph.D. graduate students how to design digital integrated circuits, including the design of various types of computer circuits and memories. I have authored or coauthored over 190 papers and one book, the majority of which concern digital integrated circuit design and memory design.

7. I was elected a Fellow of the IEEE in 2002 for contributions to placement and routing of ASICs. IEEE stands for Institute of Electrical and Electronics Engineers, which is the leading professional association for electrical engineers. The Board of Directors of the IEEE awards the rank of "Fellow" to individuals with an extraordinary record of accomplishments in any of the IEEE fields of interest. The total number of IEEE members who can be named Fellows

in any one year cannot exceed one-tenth of one percent of the total voting IEEE membership.

8. I received several research and teaching awards during my career. I received the Semiconductor Research Corporation's Inventor's Recognition Award in 1988 and in 2001. I also received the Technical Excellence Award from the Semiconductor Research Corporation in 1994. While serving as Professor at the University of Washington, I received the Outstanding Research Advisor award from the Department of Electrical Engineering in 2002. In 2008, I received the Distinguished Teacher of the Year Award from the Department of Electrical Engineering at the University of Texas at Dallas. I also received the Distinguished Teaching Award for the Erik Johnson School of Engineering and Computer Science in 2014.

9. Over the years, I have also received funding to conduct research in computer circuits and memory designs, including area-efficient and reliable embedded DRAM and SRAM design. Together with my graduate students, I have designed and fabricated various types of DRAM and SRAM chips.

10. I am a co-inventor on a provisional patent application and an issued patent directed to transistor and computational logic technologies.

11. A copy of my CV is attached as Appendix A.

12. I have reviewed the specification, file history and claims of U.S. Patent No. 7,523,331 to Gerardus Wilhelmus Theodorus Van Der Heijden (the "'331 patent") (Ex. 1001).

13. I have reviewed and understand the following references that form the grounds of the petition:

- Japanese Patent Application Publication JP 2000-105639 (English Translation) ("Irie") (Ex. 1002)
- Applicant Admitted Prior Art from U.S. Patent 7,523,331 ("AAPA") (Ex. 1001)
- U.S. Patent 5,694,567 ("Bourekas") (Ex. 1005)

14. I have also reviewed and understand the following documents cited in the petition: Ex. 1006, Ex. 1007, and Ex. 1009 – Ex. 1025.

15. I have been retained by Petitioner, Intel Corporation, to provide my opinion concerning the validity of the '331 patent in support of its petition for *inter partes* review of the '331 patent.

16. I am being compensated at my normal consulting rate of $375 per hour for my work. My compensation is not in any way dependent on the outcome of any *inter partes* review, and in no way affects the substance of my statements in

this declaration, nor have I any financial or personal interest in the outcome of this proceeding.

17. To the best of my knowledge, I have no financial interest in Petitioner, Intel Corporation, or in the '331 patent. To the extent any mutual funds or other investments I own have a financial interest in the Petitioner or the '331 patent, I do not knowingly have any financial interest that would affect or bias my judgment.

II. BACKGROUND TECHNOLOGY

A. Computer Memory

18. Computers have memory that stores information such as data and program instructions. K.R. Kaplan and R.O. Winder, "Cache-based Computer Systems,", March 1973 ("Kaplan") (Ex. 1018), 30. Such memory is typically divided into main memory and cache memory. Ex. 1002, ¶¶ 0011-0012; Ex. 1018, 30. Main memory is commonly used at computer start up because it stores programs that execute when the computer is turned on. U.S. Patent Publication 2002/0026601 ("Shiraga") (Ex. 1020), ¶¶ 0008, 0009. ROM (Read Only Memory) and SRAM (Static Random Access Memory) are examples of main memory. *Id*., ¶ 0008. For instance, ROM stores start-up programs that execute when a computer is powered on and SRAM stores programs during the operation of the computer. *Id.*, ¶ 0009. Dynamic RAM, also known as "DRAM," is a common type of main memory RAM. *Id.*, Abstract, ¶ 0005. In contrast, cache memory is used to store information that is

most recently accessed from main memory and provide fast access to this information if it is subsequently needed again. Ex. 1018, 30. SRAM can also be used for cache memory; SRAM is "static" in that it does ***not*** need to periodically refresh, which is different from dynamic RAM, and because of this difference SRAM cache consumes much less power than DRAM. John L. Hennessy and David A. Patterson, *Computer Organization and Design: The Hardware/Software Interface* 541 (2nd ed. 1998) ("Hennessy 2nd") (Ex. 1019), 541.

19. As I explain above, cache memory stores information so it can be accessed easily when the information is needed again without recourse to main memory. Ex. 1018, 30. An exemplary process where cache memory is used, which shows cache memory functionality, is as follows:

- Information is needed by a processor. *Id.*
- The main memory address associated with the information is used to try retrieving the information from cache. *Id.*, 30-32.
- A "cache hit" occurs if the information corresponding to the main memory address is found in cache. *Id.*, 32. The information is then obtained from the cache without accessing the slower main memory. *Id.*
- If the information is not found in the cache, a "cache miss" occurs; here, the main memory must be accessed for the information, and once it is obtained, the information is placed in cache. *Id.*

20. A difference between main memory and cache memory is access speed; cache memory is accessed much faster than main memory because cache stores information retrieved from main memory that was recently used by a processor. Ex. 1018, 30. Moreover, the typical access time is less for cache memory SRAM compared to main memory DRAM. Ex. 1019, 541; see also John L. Hennessy and David A. Patterson, *Computer Architecture: A Quantitative Approach*, at ix, 13, 277-278 (3rd ed. 2003) ("Hennessy 3rd") (Ex. 1023), 394, Fig. 5.3 (reporting typical access times in 2001 of 0.50-25 ns for CMOS SRAM cache and 80-250 ns for CMOS DRAM main memory). Hennessy 2nd explains that "[t]o save pins and reduce the package cost, the same address lines are used for both the row and column address; a pair of signals called RAS (Row Access Strobe) and CAS (Column Access Strobe) are used to signal the DRAM that either a row or column address is being supplied." Ex. 1019, B-32. Hennessy 2nd continues, "The two-level addressing scheme, combined with the internal circuitry, make DRAM access times much longer (by a factor of 5 to 10) than SRAM access times." *Id*. This means that cache memory can quickly provide this information if it is subsequently needed–much faster than if the information was obtained again from main memory. *Id.*

21. Computer memory is usually organized at different hierarchical levels determined by access speed. Ex. 1019, 541. At the higher level, which is closer to the computer processor, sits cache memory because it operates at a faster speed

compared to main memory. *Id.* Programs are stored in these various memory levels of memory, and when memory at one level (*e.g.*, main memory) is deactivated, programs can be stored in memory at a different level (*e.g.*, cache memory). Ex. 1002, ¶¶ 0028, 0032.

22. Information in a memory is accessed for reading/writing using a memory address formed from a "tag" and "index" that identifies a memory location. Ex. 1005, 1:18-30; Ex. 1019, 552-553. Addresses are uniquely identified and differentiated by the tag and index. Ex. 1005, 1:27-54.

23. Cache memory information can be organized in various ways, and a relationship can exist between main memory addresses and where information of these addresses are stored in cache memory. The relationship can be determined at the time of caching or ahead of time. Mapping the main memory and cache memory to each other ahead of time can provide that program instructions stored in a certain portion of main memory are cached in a certain portion of cache memory. *See* Ex. 1005, 4:64-5:3.

B. Operating Modes and Interrupts

24. Operating modes define the state of a computer. For instance, in a "normal" mode, a computer may have full functionality and execute all instructed programs. Ex. 1002, ¶¶ 0022-0023. In contrast, in a "low power" mode, which is

also known as a "standby mode," all or part of the computer may be powered down, and only part of the computer may be available for use. *Id.*, ¶¶ 0034-0035.

25. "Interrupts" can be used to make a computer move from one operating mode to another. Ex. 1002, ¶ 0037. For instance, an interrupt signal can make a processor interrupt its current state and perform a particular operation. *Id.* I provide an example of this situation: a computer may not be in use, and may place itself in a low power mode (standy mode) where its display and certain processors, circuitry, and/or memory are powered down to conserve power. *Id.*, ¶¶ 0034-0036. An interrupt signal may instruct the computer to wake up from this low power mode and return to a normal operating mode. *Id.*, ¶ 0037. For instance, the interrupt signal may be generated by pressing the computer's power button. *Id.*

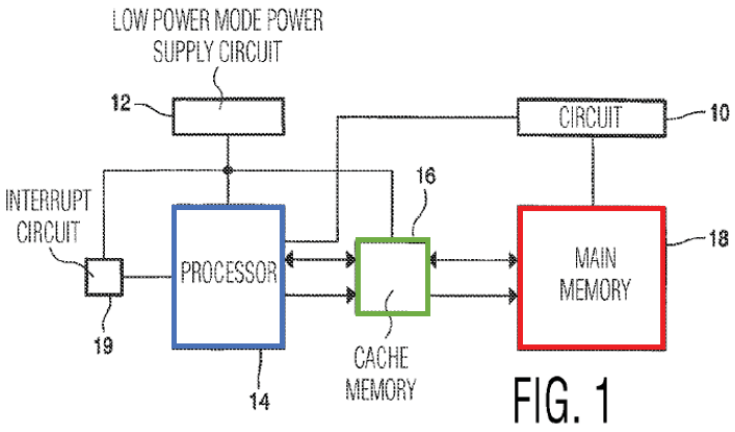
26. In a low power mode, computers usually power down most of their memory. Thus, interrupt programs can be stored in memory parts that are not powered down, such as cache memory, and executed from the cache memory. Ex. 1002, ¶¶ 0032, 0037.

III. OVERVIEW OF THE '331 PATENT

27. The '331 patent states that prior art computing devices could not perform "certain basic functions" like processing interrupts "in the low power mode without switching to the normal operating mode." Ex. 1001, 1:16-20. The patent says these prior art devices were problematic: "[s]witching back the apparatus to a

normal operating mode to perform such basic functions … considerably increase[s] power consumption" when "performed frequently while the apparatus does not have to return to full operation for other reasons." *Id.*, 1:21-25.

28. Fig. 1 of the '331 patent is shown below:



Ex. 1001, Fig. 1[1]

29. The Fig. 1 apparatus includes a processor 14, cache memory 16, and main memory 18 and says that in a "normal operating mode cache memory 16 functions as a conventional cache memory 16." Ex. 1001, 2:35-36.

30. The apparatus switches from the normal operating mode to a "low power operating mode" and the switch can occur when the apparatus detects "it is no longer necessary to operate in the normal mode." Ex. 1001, 3:7-11. Part of the switch includes that instructions of an interrupt program stored in main memory 18

[1] I have added color annotations to figures unless otherwise noted.

may be loaded into cache memory 16. *Id.*, 3:16-19. For instance, the '331 patent says that "[t]he interrupt program is stored at addresses in main memory that have been selected so that all instructions of the interrupt program can be stored together in cache memory 16." *Id.*, 3:19-22.

31. Loading can be performed in various ways. A "switchover program" may be used to "switch [the apparatus] over to the low power mode" and can be provided with the start address and end address of the interrupt program stored in main memory. Ex. 1001, 3:12-13, 23-25. The information at these addresses can be loaded from the main memory 18 into cache memory 16 using this program. *Id.*, 3:25-28. A "conventional locking mechanism" can be used by cache memory 16, which "enables processor 14 to signal that (and optionally from which addresses) instructions and/or data must be kept in cache." *Id.*, 3:31-36. Alternatively, the patent says that the interrupt program "may be stored at addresses that are selected so that it is ensured in advance that the switchover program can load the entire interrupt program into cache memory 16 without being subsequently discarded." *Id.*, 3:38-43. The patent also says that processor 14 may "contain[] an instruction to cause specified instructions, or specifically the interrupt routine, to be loaded into cache memory 16." *Id.*, 3:43-46.

32. The processor can execute the interrupt program without accessing main memory after the instructions have been loaded into cache memory 16 and the

apparatus is in the low power mode. Ex. 1001, 1:53-62. The main memory can be deactivated to reduce power consumption since it is not used. *Id.*, 1:63-2:6. For instance, the patent says that "all instructions of the interrupt program can be stored together in cache memory," and that "[t]he interrupt program is stored at addresses in main memory that have been selected so that all instructions of the interrupt program can be stored together in cache memory 16." *Id.*, 3:19-22. Storing all the instructions together in the cache (as opposed to having some instructions reside in main memory) allows the processor to "execute the interrupt program without recourse to main memory 18." *Id.*, 3:65-67.

IV. LEVEL OF ORDINARY SKILL IN THE ART

33. A person of ordinary skill in the art ("POSA") at the time of the alleged invention would have had at least: (1) an undergraduate degree in electrical engineering (or an equivalent subject), together with at least two years of post-graduate experience designing cache systems; or (2) a master's degree in electrical engineering (or equivalent subject) together with at least one year of post-graduate experience in designing cache systems.

V. UNDERSTANDING OF THE LAW

34. I am not an attorney. For the purposes of this declaration, Petitioner's counsel has informed me about certain aspects of the law that are relevant to my opinions.

35. Petitioner's counsel has informed me that a patent claim may be "anticipated" if each element of that claim is present either explicitly or inherently in a single prior art reference.

36. Petitioner's counsel has informed me that a patent claim can be considered to have been obvious to a POSA 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 POSA at the time the application was filed.

37. Petitioner's counsel has informed me 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;
- what differences, if any, existed between the claimed invention and the prior art.

38. Petitioner's counsel has informed me that a single reference can 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. Alternatively, 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, and when combined, would yield predictable results;
- whether a POSA 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.

39. Petitioner's counsel has informed me that one of ordinary skill in the art has ordinary creativity, and is not an automaton. Petitioner's counsel has

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

VI. CLAIM CONSTRUCTION

40. I have been informed that a claim in *inter partes* review is given its plain and ordinary meaning as understood by a POSA at the time of the invention in light of the claim language, specification, and prosecution history.

41. I understand that claim 7 requires an interrupt program be stored at "***addresses in main memory that have been selected so that all instructions of the interrupt program can be stored together in the cache memory.***"[2] In my opinion, the '331 patent's specification does not describe the meaning of "***stored together***." The patent just says that the instructions are "stored together," repeating the claim language. Ex. 1001, 3:19-22. In my opinion, the file history does not provide explanation of the term either. *See* Amendment (03/18/08) (Ex. 1013). This recitation of claim 7 should therefore have its plain and ordinary meaning.

42. Indeed, a POSA would have understood that the plain meaning of "***addresses in main memory that have been selected so that all instructions of the interrupt program can be stored together in the cache memory***" would at least

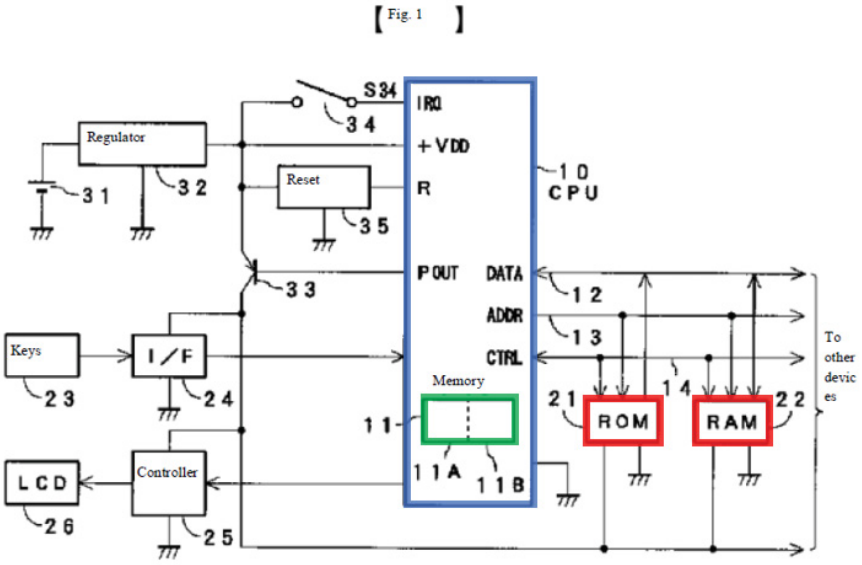
[2] Bolding, underline, and/or italics have been added to quotes, unless otherwise noted.

encompass addresses in main memory that have been selected so that all instructions of the interrupt program can be stored in a **single cache memory or a single part of cache memory at the same time**, and this interpretation is taught by the reference Irie. The '331 patent supports this understanding that all instructions of the interrupt program must be present in the cache memory during the low power operating mode so the interrupt program can be executed from the cache memory. *See* Ex. 1001, 3:48-67. For instance, the '331 patent says that "Once the interrupt program has been loaded into cache memory 16, the switchover program … cause[s] processor 14 to signal to main power supply circuit 10 to deactivate itself," and this is known as, "the low power operating mode." *Id*., 3:48-54. The '331 patent also says that "During operation in low power operating mode," processor 14 may "execute the interrupt program," and "During execution of the interrupt program main memory 18 remains deactivated." *Id*., 3:55-61. Thus, "processor 14 … is able to execute the interrupt program without recourse to main memory 18." *Id*., 3:64-67. Thus, these sections at least show that all instructions of the interrupt program must be present in the cache memory during the low power operating mode so the interrupt program can be executed from the cache memory.

VII. SUMMARY OF THE PRIOR ART REFERENCES

A. Irie

43. Irie discloses a mobile information terminal shown in Fig. 1:



Ex. 1002, Fig. 1

44. The mobile information terminal includes a CPU 10,[3] ROM 21, RAM 22, and cache memory 11. Ex. 1002, ¶¶ 0010-0013. CPU 10 is a "circuit" that "executes … process[es]" and routines such as routine 100 and routine 300, and provides "system control." *Id.*, ¶¶ 0009, 0011, 0024, 0028. ROM 21 stores "programs for controlling the mobile information terminal." *Id.*, ¶ 0013, Fig. 1. RAM 22 "provides a work area and a stack area for the CPU 10." *Id.*, ¶¶ 0013-0014.

45. The terminal is operational in a "normal operating mode" when power switch 34 is in an "on" position, and in the normal operating mode, cache memory 11 operates according to a "first mode" where an "entire area (all addresses) of memory 11 can be used as cache memory." Ex. 1002, ¶¶ 0009, 0011, 0012, 0022-

[3] CPU is a term known in the art to stand for "central processing unit."

0023, 0025, 0037, 0041, claim 1, Abstract. The terminal is placed in an "energy-saving mode" when power switch 34 is moved from "on to off." Ex. 1002, ¶¶ 0013-0014, 0024, 0028, 0033, Figs. 2-4.

46. Irie explains that instructions for an interrupt processing program are loaded from ROM 21 to RAM 22, and then from RAM 22 to cache memory 11 when the terminal switches to the energy-saving mode.[4] Ex. 1002, ¶¶ 0024-0032, Figs. 2-4. This process (governed by routine 300) is performed as follows:

- CPU 10 determines a starting address location in ROM 21 storing the program instructions. *Id.*, ¶¶ 0028, 0032, Fig. 4.
- CPU 10 then copies instructions at that address to a temporary location in RAM 22. *Id.*
- Instructions are next moved from the temporary location in RAM 22 to cache memory 11. *Id.*, ¶¶ 0027-0031.

[4] Irie describes instructions for an interrupt processing program because programs are formed by instructions; thus, the content of Irie's interrupt processing program includes instructions. *See* Ex. 1019, 14 ("The processor is the active part of the board, following the **instructions of a program** to the letter."); *see id.*, 61 ("the machine had to execute the instruction to run the program, the execution time must depend on the **number of instructions in a program**.").

- CPU 10 then determines a subsequent address location in ROM 21 where further program instructions are stored, reads the instruction content stored at this subsequent address location, copies it to the same temporary location in RAM 22, and the instructions are moved from the temporary location in RAM 22 to cache memory 11. *Id.*, ¶¶ 0028-0032.
- This process is continued until a final address location in ROM 21 is reached. *Id.*, ¶¶ 0029-0030.

47. The terminal may then be in an energy-saving mode. Ex. 1002, ¶¶ 0032-0037. To switch CPU 10 back to the normal operating mode, the interrupt processing program copied to cache memory 11 is executed from the cache memory (in particular, cache area 11A of cache memory 11). *Id.*

B. '331 Patent AAPA

48. The '331 patent admits that many concepts related to computer memory are known, conventional, and therefore in the prior art. Likewise, the claims of the '331 patent include recitations that the patent admits were known. For instance, the '331 patent's applicant admitted prior art ("AAPA") admits the following: "**Known apparatuses** that contain a **computer processor and a main memory** with **data and/or instructions for use** by the processor **are often provided** with a **cache memory** in order to speed up execution." Ex. 1001, 1:34-37. Thus, the '331 patent admits that systems including (1) a computer processor,

(2) a cache memory, and (3) a main memory were known, and that storing data and/or instructions in main memory for use by the processor was known. *Id.* , 1:34-49.

49. The AAPA also admits the following about these **known** systems: "[t]he cache memory temporarily stores copies of part of the data and/or instructions that the processor has addressed in main memory, so that it can be retrieved given its main memory address" and "**[w]hen the processor addresses such data and/or instructions again, the cache memory substitutes the cached data and/or instructions for the data and/or instructions from main memory**." Ex. 1001, 1:37-43. I note that this admittedly known disclosure from AAPA is the equivalent of language in claim 7, including limitation 7[c], as explained below.

50. AAPA further states: "cache memory is **conventionally** one of the circuits that are deactivated when the apparatus is switched to the low power mode, because it merely stores redundant copies of part of the data and/or instructions that were used during previous processing." Ex. 1001, 1:45-49. Thus, AAPA admits to the conventional behavior that its known apparatuses (which include a computer processor, cache memory, and main memory) provide. *Id.*

51. There are additional areas where '331 AAPA admits what is in the prior art:

- "[i]n the normal operating mode cache memory 16 functions as a **conventional cache memory** 16;" AAPA explains that in this conventional operation, "[w]hen processor 14 needs an instruction and/or data, processor 14 outputs the address with which the instruction and/or data is addressed in main memory on its address/data interface" and "[c]ache memory 16 receives the address and tests whether an instruction and/or data corresponding to the address is stored in cache memory 16." Ex. 1001, 2:35-42.
- AAPA further explains this conventionality when it notes "[t]echniques for this type of testing **are known per se**" and in the conventional process, "[i]f the addressed instruction and/or data is available in cache memory 16, cache memory 16 returns the instruction and/or data from cache memory 16" but "[i]f the addressed instruction and/or data is not available in cache memory 16, cache memory forwards the address to main memory 18, which returns the instruction and/or data to cache memory 16." *Id.*, 2:42-48.
- AAPA also admits that in this conventional process, "[c]ache memory 16 then supplies the instruction and/or data from main memory 18 to processor 14" and "[p]referably, cache memory 16 also stores a copy

of this instruction and/or data, for later use, when processor 14 uses the address again." *Id.*, 2:48-52.

C. Bourekas

52. Bourekas' system has a cache memory, main memory, and CPU–just like Irie and AAPA. Ex. 1005, 3:46-64, 4:21-24. Bourekas discloses how main memory and cache memory are mapped to each other, and how saving information in one part of main memory means that the information will be cached in a certain part of cache memory. *Id.*, 3:65 - 4:2, Fig. 3. For instance, Bourekas describes a main memory having a physical address range 0000 to 0111 and 1000 to 1111 and a cache memory having a cache index range 000 to 011 and 100 to 111. *Id.* Information can be locked in certain addresses of the cache memory so they are not deleted. For example, Bourekas explains that a program can be locked in the "lower-order half of the cache RAM," which refers to cache index addresses 000 to 011. *Id.*, 4:2-5, Fig. 3.

53. The physical address range 0000 to 0111 of main memory is mapped to the cache index range 000 to 011. Ex. 1005, 3:65 - 4:2, Fig. 3. The physical address range 1000 to 1111 of main memory is mapped to the cache index range 100 to 111. *Id.* Thus, a user can lock a "4 word program" stored at main memory physical address range 0000 to 0011 in the "lower-order half of the cache RAM" cache address at 000 to 011. *Id.*, 4:2-5, Fig. 3.

VIII. SUMMARY OF OPINIONS

54. It is my opinion that every limitation recited in claim 7 of the '331 patent is disclosed by the prior art, and is anticipated and/or rendered obvious by the prior art.

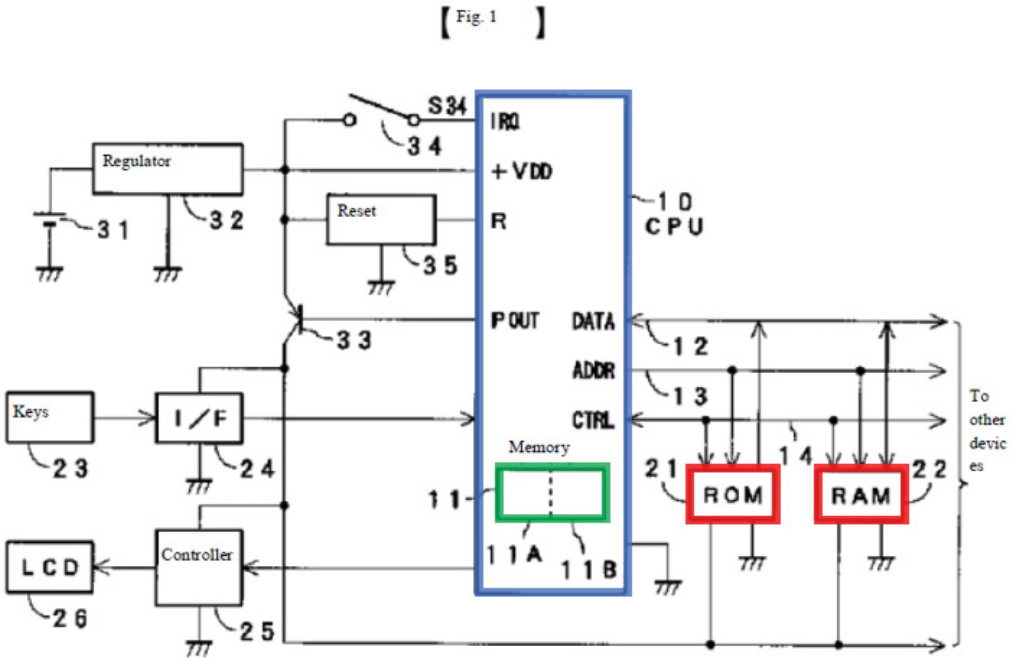
IX. INVALIDITY OF THE CHALLENGED CLAIMS

A. Ground I: Claim 7 is Rendered Obvious by Irie in View of AAPA

1. Claim 7

a) 7[a]: "A method of operating an apparatus that contains an instruction processing circuit, a main memory addressable by the instruction processing circuit and a cache memory, the method comprising"

55. Irie discloses the preamble 7[a]. Irie discloses a mobile information terminal (shown in figure 1 below) which is "*an apparatus*" as claimed. Ex. 1002, ¶ 0010.



Ex. 1002, Fig. 1

56. The mobile information terminal "*contains an instruction processing circuit, a main memory addressable by the instruction processing circuit and a cache memory*" as claimed. First, the terminal includes CPU 10 which is "*an instruction processing circuit*" because it is a "circuit" that "executes…process[es]" such as routine 100 and routine 300 and provides "system control." Ex. 1002, ¶¶ 0009-0011, 0024, 0028, Fig. 1; *see also id.*, claim 1 ("[a]n energy-saving circuit, comprising a CPU which controls operation"). Irie therefore discloses "*an apparatus that contains an instruction processing circuit*" as claimed.

57. Second, Irie discloses that its terminal includes a "*main memory*." The terminal contains a ROM 21 and a RAM 22, which are shown above in reproduced

Figure 1. Ex. 1002, ¶ 0013, Fig. 1. Both ROM 21 and RAM 22 can be considered "*a main memory*" as claimed. Indeed, the '331 patent acknowledges that ROM and RAM are both types of "*main memory*." Ex. 1001, 2:65-3:2.

58. Furthermore, in my opinion, ROM 21 is "*a main memory*" because it "has programs for controlling the mobile information terminal," (Ex. 1002, ¶ 0013, Fig. 1) just as the main memory of the '331 patent holds instructions executed by its processor 14. *See* Ex. 1001, 2:27-34. Specifically, the '331 patent explains that "processor 14 ***executes programs of instructions that are retrieved from main memory 18***." *Id*. Likewise, Irie discloses "a CPU which controls operation of predetermined circuits, [and] a ***ROM in which first and second programs which are executed by the CPU are written***." Ex. 1002, ¶ 0009. My understanding is also supported by Shiraga (Ex. 1020) that describes an information processing apparatus that has a power saving method. Ex. 1020, ¶ 0002. Ex. 1020 states "In the information processing apparatus, it is generally necessary for CPU to fetch a command at a reset vector address immediately after the power is turned on. From this reason, the apparatus is provided with an inexpensive ROM (boot ROM) as a portion of the main memory." *Id*., ¶ 0009.

59. In addition to the '331 patent's express inclusion of ROM as a type of main memory, as I explain above, the RAM 22 of Irie also qualifies as "*a main memory*" because it "provides a work area and a stack area for the CPU 10." Ex.

1002, ¶ 0014. A POSA would have understood this is a typical characteristic of a "*main memory*." My understanding is supported by U.S. Patent 3,713,107 (Ex. 1025), which describes a "main memory" having a "work area" (Ex. 1025, 6:38-40, Fig. 7). Moreover, the '331 patent explains the main memory supplies the instructions and/or data copied to the cache memory. The '331 patent states, "If the addressed instruction and/or data is not available in cache memory 16, cache memory forwards the address to ***main memory 18, which returns the instruction and/or data to cache memory 16***. Cache memory 16 then supplies the instruction and/or data from main memory 18 to processor 14. Preferably, cache memory 16 also ***stores a copy of this instruction and/or data***, for later use, when processor 14 uses the address again." Ex. 1001, 2:45-52. Irie also uses RAM 22 to supply program instructions to its cache memory 11. Irie reads "programs and tables … into the memory 11 … provided in the ROM 21." Ex. 1002, ¶ 0027. Irie does so by reading content from ROM 21 to "a temporary location," which "is any open space in RAM 22." *Id*., ¶ 0028. When "the program 200 … is copied to the same temporary address [in RAM 22]…the area 11A in the memory 11 is used as cache memory." *Id*., ¶ 0031. In this way, "program 200 … is sequentially copied to the cache area 11A." *Id*. Thus, Irie's RAM 22 supplies program instruction to its cache memory 11 just as the '331 patent's main memory 18 supplies instructions and/or data to its cache memory 16.

60. Hennessy 3rd also describes that a RAM is a type of main memory. For instance, Hennessy 3rd explains that "[t]he main memory of virtually every desktop or server computer sold since 1975 is composed of semiconductor DRAMs." Ex. 1023, 454. A DRAM is a dynamic RAM. *Id.*, 455; Ex. 1019, 541. Indeed, Hennessy 2nd explains that "Main memory is implemented from DRAM (dynamic random access memory), while levels closer to the CPU (caches) use SRAM (static random access memory)." Ex. 1019, 541. Therefore, both ROM 21 and RAM 22 are typical examples of *main memory*, and Irie discloses "*an apparatus*" that contains "*a main memory*" as claimed.

61. Irie also discloses that ROM 21 (the claimed "*main memory*") is "*addressable*" by CPU 10 (the claimed "*instruction processing circuit*") as claimed. This is because CPU 10 executes a program routine 300, which copies a program from ROM 21 to cache memory 11. Ex. 1002, ¶ 0028. By executing routine 300, CPU 10 determines a starting address location in ROM 21 storing instruction content for a program, reads this content, and copies it to a temporary location in RAM 22. *Id.*. This content is then moved from the temporary location in RAM 22 to cache memory 11. *Id.*, ¶¶ 0027-0031. Then, CPU 10 determines a subsequent address location in ROM 21 where further program content is stored, reads the content stored at this subsequent address location, copies it to the same temporary location in RAM 22, and this content is moved from the temporary location in RAM 22 to cache

memory 11. *Id.*, ¶¶ 0028-0029. These steps continue for each subsequent address until a final address location holding content in ROM 21 is reached. *Id.*, ¶¶ 0029-0030. Therefore, because CPU 10 determines address locations of ROM 21 to read content stored at these address locations, ROM 21 (the claimed "*main memory*") is "*addressable*" by CPU 10 (the claimed "*instruction processing circuit*"). *Id.*, ¶¶ 0027-0031, Fig. 4.

62. RAM 22, which can also be considered the claimed "*main memory*," is "*addressable*" by CPU 10 (the claimed "*instruction processing circuit*"). RAM 22 "provides a work area and a stack area for CPU 10," and a POSA would have understood that a "work area" refers to a range of memory addresses in RAM 22 to which CPU 10 can read and write data directly. Ex. 1002, ¶ 0014; *id.*, ¶ 0011. My understanding is supported by U.S. Patent No. 3,713,107 (Ex. 1025), which describes a "main memory" having a "work area…used to operate as an input buffer for storage of raw data to be processed" (Ex. 1025, 6:38-40, Fig. 7).

63. Thus, because CPU 10 reads and writes to these memory addresses of RAM 22 directly, RAM 22 (which can be considered the claimed "*main memory*") is "*addressable*" by CPU 10 (the claimed "*instruction processing circuit*").

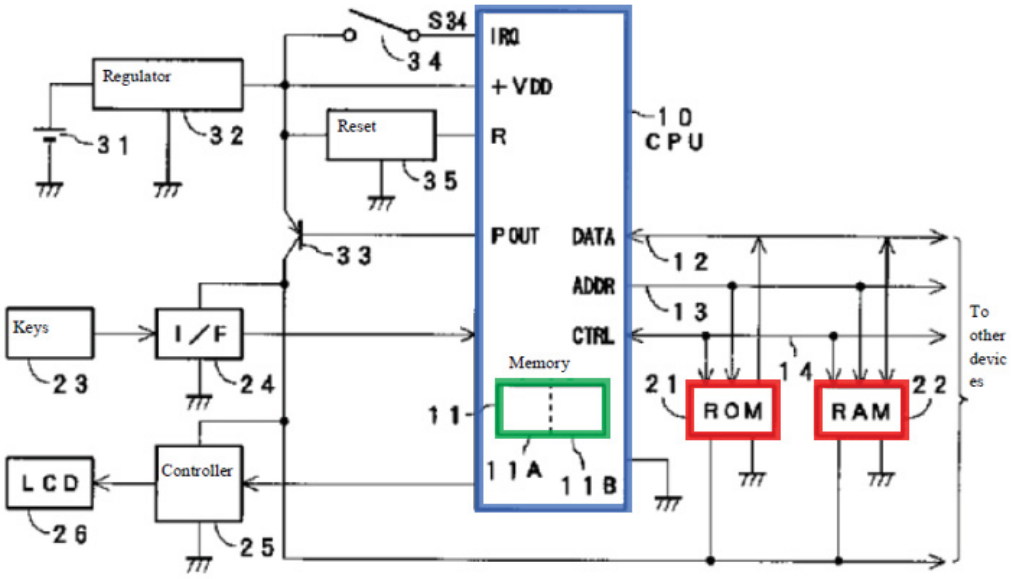
64. I also note that Hennessy 2nd states that "[o]ccasionally, people call the processor the CPU." Ex. 1019, 14. Figure 5.1 shows the main components of a million instructions per second (MIPS) implemented processor and that the

processor directly supplies addresses to both data memory and instruction memory. *Id*., 338-340, Fig. 5.1, 344-345, G-7. This disclosure confirms the understanding a POSA would have had regarding how processors and memory, including main memory, interact, and how main memory is addressable by a processor (CPU). Thus, when viewing Irie, a POSA would have understood that the ROM 21 and RAM 22 are addressable by CPU 10.

65. Thus, Irie discloses "*a main memory addressable by the instruction processing circuit*" as claimed.

66. Irie's mobile information terminal (the claimed "*apparatus*") also contains cache memory 11, which is "*a cache memory*" as claimed. Ex. 1002, ¶ 0012. I provide figure 1 below as reference.

[Fig. 1]



Ex. 1002, Fig. 1

67. Irie explains that "CPU 10" of the mobile information terminal "has a cache memory 11." Ex. 1002, ¶ 0012; *see also id.*, claim 3. Thus, Irie discloses "*an apparatus*" (Irie's mobile information terminal) contains "*a cache memory*" as claimed. Irie also discloses "*[a] method of operating*" its terminal (the claimed "*apparatus*") as claimed because the reference explains that when a power switch 34 is in an "on" position, the terminal (claimed "*apparatus*") is operated in a "normal operating mode." Ex. 1002, ¶¶ 0022-0023. When power switch 34 is moved from "on to off," the terminal (claimed "*apparatus*") operates by executing routines. *Id.*, ¶¶ 0013-0014, 0024, 0028, 0033, Figs. 2-4. Irie therefore discloses "*[a] method of operating an apparatus*" as claimed.

68. Accordingly, Irie discloses the preamble 7[a]

b) 7[b]: "using the cache memory and the main memory in a normal operating mode . . ."

69. Irie discloses limitation 7[b]. To begin with, Irie describes a CPU 10 (having an external memory and cache) that can operate in a "*normal operating mode*." Ex. 1002, ¶ 0011 ("CPU 10…[has] an energy-saving mechanism which allows **operation in normal mode**..."), ¶¶ 0012-13; *see also id.*, claim 1, Abstract.

70. In that normal operating mode , Irie discloses "*using*" cache memory 11 (the claimed "*cache memory*") in the normal operating mode (the claimed "*normal operating mode*"). Cache memory 11 is used in the normal operating mode

according to a "first mode" where the "entire area (all addresses) of memory 11 can be used as cache memory." Ex. 1002, ¶¶ 0012, 0022-0025, 0037, 0041. Specifically, Irie explains that when "the mobile information terminal is in **normal operating mode**… the cache memory 11 is set to a first operating mode" and "both areas 11A and 11B operate as cache memory." *Id.*, ¶ 0023. Irie thus discloses "*using*" cache memory 11 in a "*normal operating mode*."

71. Irie also discloses "*using*" external memories ROM 21 and RAM 22 (either of which can be the claimed "*main memory*" as I explain above) in a normal operating mode. Irie states that "in normal operating mode…the output voltage of the regulator circuit 32 is supplied to…memories 21 and 22," such that ROM 21 "has programs for controlling the mobile information terminal" (and "operates as a general purpose mask ROM"), while RAM 22 "provides a work area and a stack area for the CPU 10." *Id.*, ¶¶ 0013-0014, 0022. Irie further discloses that both memories are operational when CPU 10 is in a "normal mode," because "operating

voltage" is supplied to memories ROM 21 and RAM 22[5]; ROM 21 then stores program 100, which is executed, while RAM 22 is "enabled." *Id.*, ¶¶ 0037-0039; *see also id.*, claim 1 ("the CPU, when operating in the **normal operating mode**, has the switch element turned on by the CPU, **supplying voltage** from the power source through the switch element **to the ROM** and the predetermined circuits **as operating voltage** . . . ."). Irie thus discloses "*using*" ROM 21 and RAM 22 (the claimed "*main memory*") in a normal operating mode.

72. Accordingly, Irie discloses limitation 7[b].

> ***c) 7[c]: ". . . to cache in cache memory a part of data and/or instructions that the instruction processing circuit addresses in the main memory during execution and to substitute cached data and/or instructions when the instruction processing circuit addresses the data and/or instructions in the main memory;"***

73. Irie in view of AAPA teaches limitation 7[c].

[5] Although Irie states "operating voltage is supplied to the peripheral devices such as the circuits 24 and 25 **in** the memories 21 and 22, etc.," Figure 1 shows that circuits 24 and 25 are separate circuits not located "**in**" memories 21 and 22. Ex. 1002, ¶ 0038, Fig. 1. The word "**in**" should thus be read as "**and**," as confirmed by paragraph 0022 of Irie, which states that "voltage…is supplied to the circuits 24 and 25 **and** the memories 21 and 22, etc…." Ex. 1002, ¶ 0022.

74. Irie discloses using its cache memory 11, ROM 21, and RAM 22 in a normal operating mode (claimed "*using the cache memory and the main memory in a normal operating mode*"). I explain this above in section IX.A.1.b (limitation 7[b]).

75. Also, "cache memory 11 is set to a first operating mode, in which both areas 11A and 11B operate as cache memory." Ex. 1002, ¶¶ 0022-0023. This setting improves processing speed of CPU 10. *Id.*

76. A conventional and well known implementation of cache memory functionality is described by AAPA. In my opinion, a POSA would have implemented this cache functionality of AAPA into Irie's cache memory 11. AAPA states that "**known apparatuses** that contain a computer processor and a main memory with data and/or instructions for use by the processor are often provided with a **cache memory in order to speed up execution**" and that in this known apparatus, "[t]he **cache memory temporarily stores copies of part of the data and/or instructions that the processor has addressed in main memory, so that it can be retrieved given its main memory address**." Ex. 1001, 1:34-40; *see also id.*, 2:35-52. Thus, AAPA discloses using known cache memory and main memory "*to cache in cache memory a part of data and/or instructions that the instruction processing circuit addresses in the main memory during execution*" as claimed because AAPA describes how known caches store copies of part of data and/or

instructions that a processor addressed in main memory to speed up execution. *Id.*, 1:34-40.

77. AAPA also explains "**[w]hen the processor addresses such data and/or instructions again, the cache memory substitutes the cached data and/or instructions for the data and/or instructions from main memory**." Ex. 1001, 1:37-43. This disclosure therefore means that AAPA discloses "*to substitute cached data and/or instructions*" as claimed, and that the substituting of AAPA occurs when the processor addresses data and/or instructions in main memory that were previously addressed and temporarily stored in the cache (claimed "*when the instruction processing circuit addresses the data and/or instructions in the main memory*"). *Id.*. Thus, AAPA discloses "*to substitute cached data and/or instructions when the instruction processing circuit addresses the data and/or instructions in the main memory*" as claimed. *Id.*

1. **Constructions from District Court**

78. I understand that in the corresponding district court litigation, the parties have proposed constructions of "data and/or instructions that the instruction processing circuit addresses in the main memory" and "when the instruction processing circuit addresses the data and/or instructions in the main memory." In my opinion, Irie and the AAPA satisfy each party's constructions proposed for these terms in the related district court litigation.

79. I understand that Petitioner proposed that "data and/or instructions that the instruction processing circuit addresses in the main memory" means "data and/or instructions whose main memory address the instruction processing circuit outputs on its address/data interface" and "when the instruction processing circuit addresses the data and/or instructions in the main memory" means "when the instruction processing circuit outputs the main memory address of the data and/or instructions on its address/data interface" in the district court litigation. Ex. 1006, Exhibit A, 3. The constructions proposed by the Petitioner in the district court are satisfied by Irie in view of AAPA.

80. Irie's CPU 10 uses each "address in the ROM 21" ("*main memory*") that stores the content of program 200 ("*data and/or instructions*") when reading program 200 from ROM 21 ("*main memory*") and copying the program into cache memory 11. Ex. 1002, ¶ 0028. I note that paragraph 0028 of Irie refers to a "Program 20," but I understood this to be a typographical error for "program 200" which is used in Irie. Ex. 1002, ¶¶ 0027, 0029-0031, 0033, 0039.

81. Moreover, Irie explains that "CPU 10 ["*instruction processing circuit*"] is connected to … ROM 21 … ["*main memory*"] via data bus 12, address bus 13, and control bus 14." *Id*., ¶¶ 0012-0013, Fig. 1. A POSA would have therefore understood that when the content of program 200 ("*data and/or instructions*") are copied from ROM 21 to cache memory 11, CPU 10 ("*instruction processing*

circuit") addresses program 200 in ROM 21 ("*main memory*") by outputting each address of ROM 21 ("*main memory address*") corresponding to each portion of the instruction content of program 200 on address bus 13 (where address bus 13 is an address/data interface of CPU 10) such that the address is provided to ROM 21. *Id.*, ¶¶ 0012-0013, Fig. 1. In my opinion, AAPA confirms this understanding that a POSA would have had of Irie because it describes "**conventional** cache memory" operations in which a processor in need of an instruction and/or data "outputs the address **with which the instruction and/or data is addressed** in main memory on its address/data interface." Ex. 1001, 2:35-39. And a POSA would have had this understanding because the address bus 13 acts as an address interface between CPU 10 and ROM 21, and therefore provides the medium by which addresses in ROM 21 required by CPU 10 are output from CPU 10. *Id.* *See also*, Ex. 1002, ¶¶ 0012-0013, Fig. 1

82. I understand that Patent Owner proposed "data and/or instructions that the instruction processing circuit addresses in the main memory" means "data and/or instructions that the instruction processing circuit identifies using addresses with which they can be retrieved from main memory" and "when the instruction processing circuit addresses the data and/or instructions in the main memory" means "when the instruction processing circuit identifies the data and/or instructions using addresses with which they can be retrieved from main memory" in the district court

litigation. *VLSI Tech. LLC v. Intel Corp.*, No. 18-966 (D. Del.), (Ex. 1006), Joint Claim Construction Chart, Exhibit A, 3. The Patent Owner's constructions in the district court litigation are also satisfied by Irie in view of AAPA.

83. As I discuss above, CPU 10 uses each "address in the ROM 21" ("*main memory*") that stores the content of program 200 ("*data and/or instructions*") when reading program 200 from ROM 21 ("*main memory*") and copying the program into cache memory 11. Ex. 1002, ¶ 0028. A POSA would have therefore understood that CPU 10 identifies each portion of the content of program 200 ("*data and/or instructions*") using its respective "address in the ROM 21." *Id.* This is because each address of ROM 21 corresponds to each instruction of program 200 and is used to copy the instruction stored at each address into cache memory 11. *Id.* Furthermore, in my opinion, a POSA would have further understood that because each address of ROM 21 is used to copy the content stored at that address into cache memory 11, each ROM 21 address can be used to retrieve its respective instruction content from ROM 21 ("*main memory*"). *Id.*

84. AAPA also satisfies the constructions proposed by Patent Owner in the district court litigation, as it describes that instructions and/or data are retrieved from main memory "given its main memory address" when the processor addresses the data and/or instructions in main memory. Ex. 1001, 1:34-40. Thus, a POSA would have understood that the AAPA processor uses "its main memory address" to

identify the data and/or instructions it retrieves from main memory. *Id*. This POSA understanding of how a processor uses addresses to identify information (*i.e.*, data and/or instructions) it retrieves from main memory is corroborated by Hennessy 2nd, which explains "[t]o access a word in memory, the instruction must supply the memory address…the address acting as the index to that [memory] array." Ex. 1019, 111. Hennessy 2nd further provides an example using Figure 3.2, reproduced below.

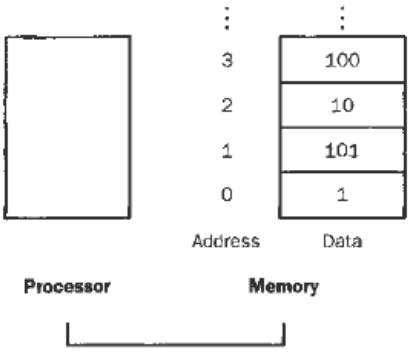


FIGURE 3.2 Memory addresses and contents of memory at those locations. This is a simplification of the MIPS addressing; Figure 3.3 shows MIPS addressing for sequential words in memory.

85. Hennessy 2nd explains that in Figure 3.2, the data value "10" is present at address "2" in memory. Ex. 1019, 111, Fig. 3.2. Thus, Ex. 1019 states that "the value of Memory[2] is 10." *Id.* Accordingly, Ex. 1019 shows how a memory address—used by a processor to access the information at the address—identifies the information stored at that address, and corroborates the understanding of POSA that a processor uses addresses to identify the information (*i.e.*, the data and/or instructions) it retrieves from main memory. *Id.*

86. As I explain above, AAPA describes known functionality of cache memory, and a POSA would have understood that the cache functionality of AAPA would have been implemented by Irie's cache memory 11. Implementing this functionality of AAPA with the addressing performed by Irie (as described in this section) would have been well within the knowledge of a POSA, and a POSA would have been motivated to combine Irie and AAPA for the reasons noted below.

87. Thus, Irie in view of AAPA teaches limitation 7[c].

2. **Motivation to Combine**

88. The combination of Irie and AAPA simply uses prior art cache memory elements according to known methods and would have yielded predictable results: namely, a cache memory that caches information. Thus, a POSA would have been motivated to combine the references. As I explain above, Irie in view of AAPA teaches each element claimed in limitation 7[c]. *See* Section IX.A.1.c. I also note that a POSA would have combined the elements of Irie and AAPA by known methods and each element performs the same function in the combination as it does separately. This is because Irie describes the following system having the following components: CPU 10, cache memory 11, and main memory (ROM 21 and RAM 22); Irie also describes that cache memory 11 operates in a normal operating mode and interacts with these other parts; Irie further describes the addressing of instruction content in main memory. *See* Section IX.A.1.a, IX.A.1.b, IX.A.1.c

(limitations 7[a], 7[b], 7[c]). Likewise, AAPA describes a known system that includes a processor, cache memory, and main memory, and that the cache memory interacts with the processor and main memory. Ex. 1001, 1:34-44. A combination of Irie with AAPA would have made Irie's cache memory 11 operate like AAPA's cache in Irie's normal operating mode. *Id.*; Ex. 1002, ¶¶ 0009-0013. The combination would have merely involved incorporating AAPA's known cache functionality into Irie's cache memory 11. *Id*. In my opinion, this combination would have been performed by known, conventional methods of memory design. This is because Irie's and AAPA's hardware (cache memory) perform the same function (caching); the adjustment would simply provide that the way Irie's cache memory 11 caches information would be adjusted. The systems of Irie and AAPA both include main memory and a processor that function with cache memory. Ex. 1001, 1:34-42, 2:35-52; Ex. 1002, ¶¶ 0009-0013. Thus, a POSA would have readily appreciated, and been readily able to implement, operating Irie's cache memory 11 like AAPA's cache memory in the normal operating mode. *Id.* I also note that Irie's cache memory 11 would still function when combined with AAPA in the same way that it does separately. Indeed, cache memory 11 would still cache information and interact with a processor (CPU 10) and a main memory (ROM 21 or RAM 22). Ex. 1002, ¶¶ 0009-0013.

89. A POSA would have further recognized that combining Irie and AAPA would have yielded predictable results. As I explain above, cache memory 11 operates like AAPA's conventional and known cache memory in the combination. Predictable results occur from the combination and the combination has a reasonable likelihood of success. Irie's system would have addressed information in ROM 21 using address bus 13, and cache memory 11 would have operated to cache information and perform information caching like AAPA describes in normal operation. The combination of Irie and AAPA would have provided a cache that operates in a normal operating mode to "*cache in cache memory a part of data and/or instructions that the instruction processing circuit addresses in the main memory during execution and to substitute cached data and/or instructions when the instruction processing circuit addresses the data and/or instructions in the main memory*" as claimed. I note that the combination of Irie and AAPA also describes this claimed recitation under each party's constructions.

90. A POSA would have also combined Irie and AAPA because the combination is use of a known technique to improve similar devices in the same way. As I explain above, Irie describes a base device that describes limitation 7[c], and AAPA describes a device that includes a processor, cache memory, and main memory - the same components as Irie. *See* IX.A.1.c. The Irie and AAPA devices are therefore comparable, and also compatible. Moreover, AAPA teaches that its

cache memory increases execution speed, which is an improvement that Irie recognizes and seeks. Ex. 1001, 1:34-37; Ex. 1002, ¶ 0023.

91. Indeed, recognizable benefits are associated with the combination of Irie and AAPA, and these benefits show that a POSA would have been motivated to combine Irie and AAPA. A POSA would have been motivated to combine Irie and AAPA to at least obtain the benefit of increased execution speed. Irie explains that its cache memory is used during the execution of program routines to receive information addressed from ROM 21 ("*main memory*"). Ex. 1002, ¶¶ 0011-0014, 0024-0032. AAPA similarly explains that "known apparatuses that contain a computer processor and a main memory with data and/or instructions for use by the processor **are often provided with a cache memory in order to speed up execution.**" Ex. 1001, 1:34-37. Therefore, operating Irie's cache memory like AAPA's known cache would have provided a beneficial increase in execution speed, which would have caused Irie's cache to receive information from ROM 21 faster and improved the functionality of Irie's mobile terminal. Ex. 1001, 1:34-37; Ex. 1002, ¶¶ 0011-0014, 0023-0032. Increasing execution speed is a known design goal for memory systems in general, and in particular, cache memory, and a POSA would have wanted to obtain this design goal. For instance, Hennessy 2nd explains that in cache operation and locating blocks in cache, "speed is of the essence." Ex. 1019, 573. Ex. 1019 further explains that in memory systems, CPU speeds continue to

increase and to keep up, access times to memory must also improve. Ex. 1019, 618-621. Ex. 1023 also explains that speed was an important consideration in memory systems and that in hierarchical memories having various levels, "[t]he goal is to provide a memory system with…speed almost as fast as the fastest level." Ex. 1023, 390.

92. Therefore, a POSA would have been motivated to combine Irie and AAPA.

93. Accordingly, Irie in view of AAPA teaches teach limitation 7[c].

d) 7[d]: "storing, in the main memory, a program of instructions for executing an interrupt function during operating in a low power operating mode,"

94. Irie discloses limitation 7[d]. First, it discloses "*a program of instructions for executing an interrupt function during operating in a low power operating mode,*" by disclosing an "interrupt processing program" that is executed when CPU 10 is in an "energy-saving mode." Ex. 1002, ¶¶ 0027, 0036, 0037. The energy-saving mode is entered when power switch 34 is "turned from on to off," which causes a "signal S34" to move "from H level to L level," leading the CPU 10 to enter an "energy-saving mode" and "power to peripheral devices" such as "circuits 24 and 25 and the memories 21 and 22" to be "turned off." *Id.*, ¶¶ 0024, 0036; *see also id.*, ¶¶ 0025-0035. Then, when CPU is in "energy-saving mode," power switch 34 can be "turned from off to on," moving signal S34 "from L level to

H level," to initiate a return to the normal operating mode. *Id.*, ¶ 0037. "When this [change in signal S34] happens, the CPU 10 is interrupted, and the interrupt program…is executed." *Id.* Irie thus discloses a "*program of instructions for executing an interrupt function*" because it describes an interrupt processing program executed when CPU 10 is in its energy-saving ("*low power operating*") mode.

95. Irie discloses "*storing*" this program of instructions "in the main memory" in both ROM 21 and RAM 22 (either of which can be the claimed "*main memory*"). Irie states that "programs"—including the interrupt processing program — are "provided in the ROM 21." Ex. 1002, ¶ 0027. Irie then describes storing that program in RAM 22 because the program is transferred from ROM 21 into cache 11 using routine 300, which "copie[s]" it from addresses of ROM 21 to a temporary location in RAM 22, and thereafter to cache 11. *Id.*, ¶¶ 0027-0032, Figs. 2-4.

96. Accordingly, Irie discloses limitation 7[d].

e) 7[e]: "wherein the interrupt program is stored at addresses in main memory that have been selected so that all instructions of the interrupt program can be stored together in the cache memory;"

97. Irie teaches limitation 7[e]. I first note limitation 7[d] recites "*storing … a program of instructions for executing an interrupt function during operating in a low power operating mode*." Ex. 1001, 8:8-10. Claim 7 then recites "*the interrupt program*" in limitation 7[e]. *Id.*, 8:11. A POSA would understand that "*the interrupt*

program" refers to the previously recited "*program of instructions for executing an interrupt function*" in limitation 7[d] because the "*program of instructions for executing an interrupt function*" is the only "program" recited in the claim up to this point. *See* Ex. 1001, claim 7. The same is true for "*instructions of the interrupt program*" because the only "instructions" recited in the claim up to this point are the "*instructions for executing an interrupt function*." *See id*. Thus, a POSA would understand that "*program of instructions for executing an interrupt function during operating in a low power mode*", "*the interrupt program*", and "*instructions of the interrupt program*" all refer to the same program and the same instructions of that program that are executing in the low power operating mode. Ex. 1001, claim 7 ("executing ***the interrupt program*** from said at least part of the cache memory"). As I explain in Section IX.A.1.d (limitation 7[d]), Irie stores the interrupt processing program at addresses (claimed "*program of instructions for executing an interrupt function*") in ROM 21 and RAM 22 (each of claimed "main memory"). Ex. 1002, ¶¶ 0027-0028, 0032. Thus, Irie discloses "*wherein the interrupt program is stored at addresses in main memory*" as claimed in limitation 7[e].

98. Limitation 7[e] concludes by requiring "*addresses in main memory that have been selected so that all instructions of the interrupt program can be stored together in the cache memory*." As I explain in Section VI (Claim Construction), this term should have its plain meaning. The plain meaning of this term at least

includes addresses in main memory that have been selected so that all instructions of the interrupt program can be stored in a single cache memory or a single part of cache memory at the same time.

99. The interrupt processing program (claimed "*program of instructions*" / "*interrupt program*") of Irie is stored in ROM 21 and RAM 22 (each of claimed "*main memory*") so that the program can be copied to cache memory 11 by routine 300. Ex. 1002, ¶¶ 0024-0032, Figs. 2-4. As I previously explained, when executed, routine 300 performs the following process:

- CPU 10 determines a starting address location in ROM 21 storing the program instructions. *Id.*, ¶¶ 0028-0032, Fig. 4.
- CPU 10 then copies instructions at that address to a temporary location in RAM 22. *Id.*
- Instructions are next moved from the temporary location in RAM 22 to area 11A of cache memory 11. *Id.*, ¶¶ 0027-0031.
- CPU 10 then determines a subsequent address location in ROM 21 where further program instructions are stored by incrementing the address by 1, reads the content stored at this subsequent address location, copies it to the same temporary location in RAM 22, and the instructions are moved from the temporary location in RAM 22 to cache memory 11. *Id.*, ¶¶ 0028-0032.

- This process is continued until a final address location in ROM 21 is reached. *Id.*, ¶¶ 0029-0030.

100. The entirety of Irie's interrupt processing program, which interrupts CPU 10 from an energy-saving mode, is copied to cache memory 11 due to this copying of instruction content at these addresses. Ex. 1002, ¶¶ 0029-0037. Peripheral circuitry to Irie's mobile information terminal, ROM 21, and RAM 22 are turned off when CPU 10 is in the energy-saving mode, which means there is no other active memory for storing the interrupt processing program that is executed to interrupt CPU 10. *Id.*, ¶¶ 0036-0037. Irie also notes that "the interrupt program [previously] copied to the cache area 11A [of cache memory 11] is executed" when switching CPU 10 out of the energy-saving mode. *Id.*, ¶ 0037. Thus, a POSA would have understood that the entire interrupt processing program (where "*program of instructions*", "*interrupt program*", and "*instructions of the interrupt program*" are all the same) would be stored in cache area 11A of cache memory 11,which is a single part of a single cache memory (*stored together in the cache memory*), at the same time. This is because there is no other active memory that can store the program. Irie discloses that the interrupt program is executed solely from cache memory 11, and in particular, area 11A of cache memory 11, without recourse to main memory ROM 21 or RAM 22. *Id.*, ¶¶ 0029-0037. Because there is no recourse

to main memory, the only location where the interrupt program can be stored is in cache memory 11. *Id.*

101. In addition, a POSA would have understood that Irie expressly discloses "*wherein the interrupt program is stored at addresses in main memory* ***that have been selected so that***" all instructions of Irie's interrupt processing program are stored together (as described above) in single cache memory 11 or a single part of cache memory (cache area 11A) at the same time. This is because Irie starts the process of copying the interrupt processing program, via routine 300, at the first address where the instructions of the interrupt processing program have been stored, namely, the starting address in ROM 21 (claimed "*main memory*") where the instructions of the interrupt processing program are stored. *See* Ex. 1002, ¶ 0028. Routine 300 then copies the instruction content at that address to a temporary location in RAM 22, and then copies the content from the temporary location to area 11A of cache memory 11, as I discuss above. *Id.*, ¶¶ 0028-0032, Figs. 2-4. The address in ROM 21 is next "incremented by 1", thereby identifying for copying the next address at which instruction content was stored, and the same process of copying instruction content occurs from this incremented address of ROM 21 to cache memory 11. *Id.*, ¶¶ 0028-0030. Route 300 repeats this process until content is copied from a final address of ROM 21 to cache memory 11. *Id.*, ¶ 0030. Thus, in my opinion, Irie stores all of the instruction content of an interrupt processing

program at selected addresses that are the starting address, final address, and intervening addresses in ROM 21 (claimed "*main memory*") so that this content can be copied to and stored together in cache memory 11 at the same time via routine 300. Therefore, in my opinion, a POSA would understand that Irie describes "*wherein the interrupt program is stored at addresses in main memory* ***that have been selected so that***" all instructions of Irie's interrupt processing program are stored together in cache memory 11 as claimed.

f) 7[f]: "detecting that it is no longer necessary to operate in the normal operating mode;"

102. Irie discloses limitation 7[f].

103. It is not expressly clear what the '331 patent means by "*detecting*." However, the '331 patent mentions U.S. Patent 5,784,628 ("Reneris," Ex. 1021), which I understand provides an example of "*detecting*" in the '331 patent. Similar to the '331 patent, Reneris describes a "method of reducing power consumption by a data processing apparatus" where the "apparatus switches to a low power mode in which most parts of the apparatus are powered down." Ex. 1021, 1:6-15. The low power mode in Reneris is initiated by a "power down condition" which is "detected" through "any detectable condition which indicates that the computer system should be placed in either a suspended or hibernated power state," such as a "user initiated" condition of "pressing of a suspend or hibernate key." Ex. 1021, 10:11-26. Reneris thus describes an example of "*detect[ing]*" a user-initiated condition (a pressing of

a key). *Id*. Because the '331 patent expressly describes Reneris as a prior art method for reducing power consumption using a low power mode, and the '331 patent does not otherwise explain "*detecting*," a POSA would have understood that the claimed "*detecting*" encompasses Reneris's disclosure of detecting a power down condition.

104. In addition, Hamilton (Ex. 1022), which I have been informed and understand was cited by the Examiner during prosecution, confirms this understanding by listing "a user[] hitting [] a special keypad button" as a condition that causes the system to leave the "normal active mode" and enter an "active sleep mode" because it is no longer necessary to operate in the normal operating mode. Ex. 1022, 7:42-8:16. Hamilton explains that in the "active sleep mode, power consumption is conserved as compared to a normal active state; however, some degree of processing power is left available." *Id*., 3:46-52.

105. Irie describes a similar mechanism for detecting that it is no longer necessary to operate in the normal operating mode. Irie states that power switch 34 is turned "from on to off" to send a signal to the CPU 10. Ex. 1002, ¶¶ 0018-0019, 0036-0037. It states that "power switch 34 does not directly turn the power to the CPU 10 or other circuits, etc., on or off," but rather, "supplies a signal S34 … to the CPU 10." *Id*., ¶ 0019. The signal S34 then changes "from H level to L level" in response to the switch, and "when this happens, the CPU 10 recognizes this" and executes routine 100. *Id.*, ¶ 0024. The routine 100 causes CPU 10 to exit the normal

operating mode and enter an energy-saving mode. *Id.*, ¶¶ 0025-0036. I also describe this process in Section IX.A.1.d (limitation 7[d]) above. This disclosure is similar to Reneris's "pressing of a suspend or hibernate key by the user" to indicate it is time to enter "either a suspended or hibernated power state," because Irie's switching of power switch 34 "from on to off" also indicates that it is "*no longer necessary*" to operate in its normal operating mode, and Irie's system recognizes signal S34 to switch to energy-saving mode. Ex. 1021, 10:11-26; Ex. 1002, ¶¶ 0018-0019, 0024-0036.

106. This parallel disclosure confirms that a POSA would have understood that Irie's recognition of signal S34 moving from high to low ("H level to L level") after a user switches power switch 34 to be the same kind of "*detecting*" as described by Reneris and Hamilton and claimed in the '331 patent.

107. Irie's "recognition" supplies the element the Applicant argued was missing from prior art during prosecution. During prosecution, the Applicant argued that in the '331 patent application, "the program of instructions is not loaded into the cache memory until it has been detected that it is no longer necessary to operate in the normal operating mode," such that "the decision threshold for loading the program of instructions is whether or not it is necessary to operate in the normal operating mode." Ex. 1011 (Amendment (11/06/07)), 6. In the prior art, on the other hand, Applicant argued that the "decision threshold for loading a program of

instructions … into a loop cache (26) is whether or not all of the instructions will fit into the loop cache (26)"—not "whether or not it is necessary for the [prior art system] to operate in the normal operating mode." *Id*. The Applicant's statements are consistent with Reneris's description of "detecting," where the user decides to move to a "suspended or hibernated power state" (*i.e.*, decides that it is no longer necessary to operate in the normal operating mode) by "pressing of a suspend or hibernate key." Ex. 1021, 10:11-26. These prosecution history statements are also consistent with the Irie, which recognizes that signal S34 moves "from H level to L level" in response to a switch as a decision threshold for when it is not necessary to operate in the normal operating mode. Ex. 1002, ¶¶ 0024-0036; *see also* Ex. 1011 (Amendment (11/06/07)).

108. Irie therefore teaches "detecting that it is no longer necessary to operate in the normal operating mode" as claimed in limitation 7[f].

g) 7[g]: "switching to the low power operating mode once it is detected that it is no longer necessary to operate in the normal operating mode, by loading the interrupt program into the cache memory from the main memory, wherein all instructions of the interrupt program are stored together in the cache memory;"

109. Irie teaches limitation 7[g]. As I already noted for limitation 7[f], Irie discloses "*detect[ing] that it is no longer necessary to operate in the normal operating mode*" because it discloses recognizing signal S34 by CPU 10. *See* Section IX.A.1.f (limitation 7[f]).

110. When this signal is recognized, routine 100 is executed to cause CPU 10 to switch from a normal operating mode to an energy-saving mode. Ex. 1002, ¶¶ 0024-0036; *see also* Sections IX.A.1.d (limitation 7[d]), IX.A.1.f (limitation 7[f]), *supra*. This energy-saving mode is the claimed "*low power operating mode*" because CPU 10 consumes less current and therefore power in that mode. Ex. 1002, ¶ 0036 (explaining that "the current consumed when the CPU 10 is [in] an energy-saving mode can be saved"); *see also id.*, ¶ 0042. Thus, Irie performs "*switching to the low power operating mode once it is detected that it is no longer necessary to operate in the normal operating mode*" by executing routine 100 to enter the energy-saving mode when CPU 10 recognizes signal S34 going "from H level to L level." *Id.*, ¶¶ 0024-0036.

111. Irie further discloses that switching to the energy-saving mode (the claimed "*low power operating mode*") is done "*by loading the interrupt program into the cache memory from the main memory.*" Ex. 1002, ¶¶ 0024-0036. Specifically, Irie explains that in step 106 of routine 100, "programs and tables which are needed for later processes are read into memory 11," and that routine 300 is executed to read and copy the program stored at starting address in ROM 21 to a temporary location in RAM 22, from which it is copied to cache memory 11. *Id.*, ¶¶ 0027-0031. The address is incremented by CPU 10 to copy the next bit of the program at the next location in ROM 21 to the same temporary location in

RAM 22, before moving it to cache memory 11. *Id.*, ¶¶ 0028-0029. This loop continues until the final address location of the program in ROM 21 is reached. *Id.*, ¶¶ 0029-0030. The program moved from ROM 21 to cache 11 is an interrupt program. *Id.*, ¶ 0032. Irie thus discloses a "*program of instructions for executing an interrupt function*"/ "*interrupt program,*" which is "*load[ed]*" from ROM 21 (claimed "*main memory*") to "cache area 11A" (claimed "*cache memory*") using routine 300. *Id.*, ¶¶ 0027, 0032.

112. As I explained in section IX.A.1.e (for limitation 7[e]), Irie discloses that the interrupt program is "stored at addresses in main memory that have been selected so that all instructions of the interrupt program can be stored together in the cache memory." Thus, when the interrupt program is loaded into the cache memory from main memory in Irie, "*all instructions of the interrupt program are stored together in the cache memory.*"

113. Irie therefore teaches limitation 7[g].

> ***h) 7[h]: "[switching to the low power operating mode once it is detected that it is no longer necessary to operate in the normal operating mode, by]***
>
> ***deactivating the main memory to reduce power consumption, but keeping active at least a part of the cache memory, that is needed for retrieving the interrupt program and for executing the interrupt function;"***

114. Irie discloses limitation 7[h]. As I discuss above in Section IX.A.1.g (limitation 7[g]), Irie discloses "*switching to the low power operating mode once it*

is detected that it is no longer necessary to operate in the normal operating mode" as claimed because Irie discloses switching a mobile information terminal from a normal operating mode to an energy-saving mode when it detects signal S34 changing "from H level to L level" in response to switch 34 being turned "from on to off." *Id.*, ¶¶ 0018-0019, 0024.

115. Irie further discloses that this switching is performed "*by deactivating the main memory to reduce power consumption, but keeping active at least a part of the cache memory, that is needed for retrieving the interrupt program and for executing the interrupt function*" as claimed.

116. Irie's mobile information terminal transitions from a normal operating mode to an energy-saving mode when Irie turns its power switch 34 from on to off. Ex. 1002, ¶¶ 0024-0036. When switching to the energy-saving mode, "the power to the peripheral devices of the CPU 10, namely…the memories 21 and 22…is turned off." *Id*, ¶ 0036. Irie therefore discloses that ROM 21 and RAM 22, which can each be considered the claimed "*main memory*", (as I describe in Section IX.A.1.a (limitation 7[a])), are "*deactivated*" as claimed because power supplied to them is "turned off." *Id.* Because the power is turned off, ROM 21 and RAM 22 are made not active and not operational; indeed, Irie states, "the current consumed" by its system "can be saved," and that "power consumption…can be reduced." *Id.*, ¶¶ 0036, 0042. Thus, Irie performs switching from a normal operating mode to an

energy-saving mode "*by deactivating the main memory to reduce power consumption*" as claimed.

117. I also note that Irie provides that when switching to the energy-saving mode from the normal operating mode, the operating mode of cache memory 11 (the claimed "*cache memory*") is switched to a second mode where an area 11A of cache memory 11 is set as dedicated cache memory, and an area 11B is set as "general purpose memory." Ex. 1002, ¶ 0025.

118. As I explain in section IX.A.1.g (limitation 7[g]), area 11A of cache memory 11 has the interrupt processing program (claimed "*interrupt program*") loaded into it from main memory. *Id.*, ¶¶ 0027-0028, 0031-0033. Cache memory 11 remains active in case the mobile information terminal must return to a normal operating mode, but peripheral devices to CPU 10 are "turned off" during the transition to the energy-saving mode (and therefore deactivated). *Id.*, ¶¶ 0034-0037. Cache memory 11 (area 11A,"*a part of*" cache memory 11), is kept active; this is because when power switch 34 is moved "from off to on" and the mobile information terminal must return to a normal operating mode, "the interrupt program copied to the cache area 11A (of cache memory 11) is executed." *Id.*, ¶ 0037. If cache memory 11 or cache area 11A were deactivated, the interrupt program stored therein would not be accessible for execution to return to the normal operating mode as needed because there would be no active memory that could be accessed to execute the

program. Thus, Irie's switching from the normal operating mode to the energy-saving mode is performed by "*keeping active at least a part of the cache memory*."

119. Irie discloses that its cache memory 11 and cache area 11A (of cache memory 11) is kept active during the energy-saving mode and "*is needed for retrieving the interrupt program and for executing the interrupt function*" as claimed. When the mobile information terminal is in an energy-saving mode and must be placed in a normal operating mode, Irie explains that "the interrupt program copied to the cache area 11A (of cache memory 11) is executed". Ex. 1002, ¶¶ 0036-0037. Irie exits the energy-saving mode by executing the interrupt routine from the cache memory 11. Cache memory 11 is therefore kept active so that the interrupt program can be read from the cache memory 11 when the terminal needs to exit the energy-saving mode. Irie therefore discloses "*retrieving the interrupt program*" as claimed because to execute the program, it must be located in the cache memory 11 (the only powered-up memory in the energy-saving mode) and made available to the CPU 10 for execution. *Id.* Irie also discloses "*executing the interrupt function*" as claimed because the interrupt program stored in cache memory 11 is executed by CPU 10 in response to power switch 34 moving "from off to on." Ex. 1002, ¶¶ 0036-0037.

120. Irie therefore discloses that its switching from a normal operating mode to an energy-saving mode is performed "*by deactivating the main memory to*

reduce power consumption, but keeping active at least a part of the cache memory, that is needed for retrieving the interrupt program and for executing the interrupt function" as claimed.

121. Thus, Irie discloses limitation 7[h].

i) 7[i]: "executing the interrupt program from said at least part of the cache memory."

122. Irie discloses limitation 7[i]. Irie describes "*executing the interrupt program from said at least part of the cache memory*" because it described executing an interrupt program (including the interrupt processing program) that interrupts CPU 10 from an energy-saving mode from cache area 11A (the claimed "*at least part*") of cache memory 11 (the claimed "*cache memory*"). Ex. 1002, ¶¶ 0035-0036. Irie confirms that "[w]hen the power switch 34 is turned off" and "the CPU 10 enters standby mode" so that "the current consumed when CPU 10 is [in] an energy-saving mode can be saved," "[w]hen the power switch 34 is turned from off to on…the CPU 10 is interrupted, and the interrupt program [previously] copied to the cache area 11A is executed in step 106." *Id.*; *see also id.*, ¶¶ 0032, 0037.

123. Accordingly, Irie discloses limitation 7[i].

B. Ground II: Claim 7 is Rendered Obvious by Irie in View of AAPA and Bourekas

1. Claim 7

a) 7[a]-7[d], 7[f], 7[h], and 7[i]

124. Irie and AAPA teach limitations 7[a]-7[d], 7[f], 7[h], and 7[i] as I explain above in sections IX.A.1.a-IX.A.1.d, IX.A.1.f, IX.A.1.h, and IX.A.1.i.

b) 7[e] "wherein the interrupt program is stored at addresses in main memory that have been selected so that all instructions of the interrupt program can be stored together in the cache memory;"

125. As I explain in section IX.A.1.e, Irie alone teaches limitation 7[e]. If "*addresses in main memory that have been selected so that all instructions of the interrupt program can be stored together in the cache memory*" is asserted or otherwise determined to mean the addresses in main memory are selected so that all instructions of the interrupt program can be stored together in **contiguous memory addresses of a cache memory at the same time**, in my opinion, Irie in view of Bourekas teaches this narrower interpretation.

126. As I note in Section IX.A.1.e (limitation 7[e]), the claimed "*interrupt program*" and "*instructions of the interrupt program*" of limitation

7[e] refer to the claimed "*program of instructions for executing an interrupt function*" that is recited in limitation 7[d]. Ex. 1001, claim 7.

127. Also, as I explain in Section IX.A.1.d (limitation 7[d]), Irie describes storing the interrupt processing program at addresses (claimed "*program of instructions for executing an interrupt function*") in ROM 21 and RAM 22 (each of claimed "main memory"). Ex. 1002, ¶¶ 0027-0028, 0032. Thus, Irie discloses "*wherein the interrupt program is stored at addresses in main memory*" as claimed in limitation 7[e].

128. The remaining parts of limitation 7[e] recites "*addresses in main memory that have been selected so that all instructions of the interrupt program can be stored together in the cache memory*." To the extent that the narrower interpretation of this recitation discussed above is adopted, Bourekas teaches this recitation.

129. Bourekas' system includes a cache memory, main memory, and CPU. Ex. 1005, 3:46-64, 4:21-24. The main memory and the cache memory are mapped to each other: physical address range 0000 to 0111 of the main memory maps to cache index range 000 to 011, and physical address range 1000 to 1111 of the main memory maps to cache index range 100 to 111. *Id.*, 3:65 - 4:2, Fig. 3. This direct mapping allows for a "4 word program" stored at main memory physical address range 0000 to 0011 to be locked by a user

in the "lower-order half of the cache RAM," which refers to the cache index addresses 000 to 011. *Id.*, 4:2-5, Fig. 3. Figure 3 below shows this arrangement. In Figure 3, the entire "4 word program" is formed by WORD 0, WORD 1, WORD 2, and WORD 3, and is stored at contiguous cache locations having address indexes 000, 001, 010, and 011, respectively. *Id.*

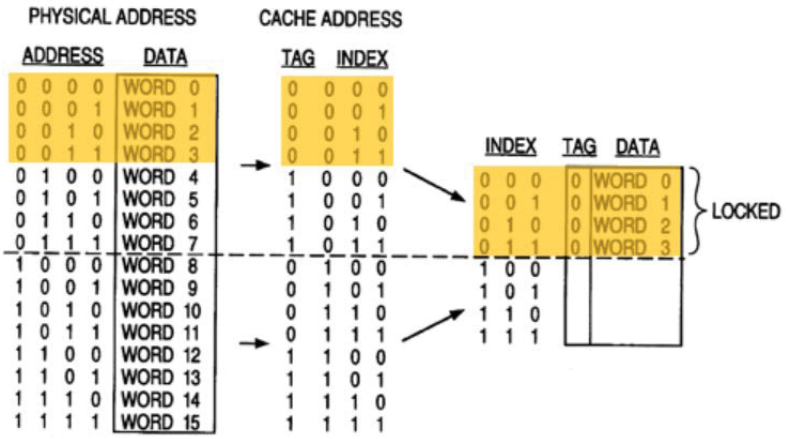


FIG. 3

Ex. 1005, Figure 3

130. Thus, a user may store the entire 4 word program at "*addresses in main memory*" 0000 to 0011 "*that have been selected so that*" all the words of this program (WORD 0, WORD 1, WORD 2, and WORD 3) are stored together in cache memory at contiguous addresses 000 to 011. *Id.* A POSA would understand this disclosed functionality and mapping between main memory and cache memory described by Bourekas is not limited to a 4 word program. Rather, a POSA would understand that programs having relatively

more words would have been stored in relatively larger caches in the same way; put another way, the storing is scalable - a larger program would have been stored in the manner disclosed by Bourekas in a larger cache. Bourekas indicates as much, stating that bits used for indexing may be adjusted and cache divided into various portions "as supported by the size of the RAM cache." Ex. 1005, 6:19-28. Thus, a POSA would have understood that programs having relatively more words would have been stored in relatively larger caches in the same way described by Bourekas at contiguous cache memory addresses. *Id.*

131. Further, a POSA would understand that, with this manner of direct mapping described by Bourekas, any contiguous set of main memory addresses that would fit in one-half (top half or bottom half) of the cache would also appear contiguously in that portion of cache. In particular, with the direct mapping described by Bourekas, a contiguous set of main memory addresses in the top half (or bottom half) of the main memory, having a size not exceeding one-half of the cache, would fit contiguously in the top half (or bottom half) of the cache. Ex. 1005, 3:65-4:10, Fig. 3. As I explain above, Fig. 3 of Bourekas shows the one-to-one mapping that results when the technique of Bourekas operates on the physical address (main memory address). Specifically, after the higher order address bits are removed, the last three bits of the cache address span half of the cache. *See id.*

132. In addition, a POSA would have known that at least one reasonable way to store information in cache is by using direct cache mapping with contiguous cache locations. This is because this approach allows for less complexity and improved control, which is a known, consistent benefit when designing memory systems. Ex. 1005, 2:25-3:20. Bourekas explains that the direct mapping provides a "direct mapped cache without TLBs, additional sets of tag comparators, or additional page management or operating system software." *Id.*, 4:10-20. The reference explains that this approach simplifies the memory system. *Id.* I note that direct cache mapping may introduce an increased cache miss rate. Ex. 1019, 570. However, this would not stop a POSA from using the teachings of Bourekas in Irie's system. As I explain, direct cache mapping provides simplicity (which provides reduced power) and improved control over where information is placed. *Id.* Moreover, direct map caching provides increased speed by having a faster "hit" time, which is a recognized benefit in memory systems. *Id*. Hennessy 2nd explains that "[t]he choice among direct-mapped, set-associative, or fully associative mapping in any memory hierarchy will depend on the cost of a miss versus the cost of implementing associativity, both in time and in extra hardware." Ex. 1019, 575. Thus, at least the benefits of simplicity, control, and speed could outweigh any possible increase in cache misses for an implementation in

which miss time and hardware cost are more important to the POSA than cache hit rate. *See id*., 570-575. Also, in the combination, cache misses would not impact Irie in the energy-saving mode because its entire program is loaded into cache using Irie's process and is therefore already located in cache and executed from the cache; Bourekas simply specifies how information is stored in main memory so it ends up in a certain part of cache memory. *See* Sections IX.A.1.d – IX.A.1.f (limitations 7[d]-7[f]) and IX.B.1.b (limitation 7[e]).

133. Hennessy 3rd corroborates this knowledge. Direct mapping between a main memory and a cache memory, as described by Bourekas, provides contiguous storage of data in cache memory. Specifically Hennessy 3rd shows a direct mapping scheme in Figure 5.4, where contiguous addresses in main memory (of which 32 such blocks are shown in Fig. 5.4) are mapped into contiguous addresses in the cache (blocks 0 through 7). Ex. 1023, 398, Fig. 5.4. For instance, Figure 5.4 shows that memory addresses corresponding to blocks 0 through 7 of main memory are mapped to cache memory addresses corresponding to blocks 0 through 7 of the cache memory, contiguously. *Id.* Likewise, contiguous main memory addresses corresponding to blocks 8 through 15 are also mapped to the cache memory addresses corresponding to 0 through 7, contiguously. *Id.* The mathematical formula for this manner of direct mapping from a set of contiguous addresses in main memory to a set of

contiguous addresses in the cache is: (main_memory_block_address MOD cache_size), where cache_size represents the number of blocks in the cache.[6] *Id.*, 397. A POSA would understand that, with this manner of direct mapping, any contiguous set of main memory addresses that would fit in the cache would also appear contiguously in the cache.

134. Hennessy 2nd also corroborates that a POSA would have known that at least one reasonable way to store information in cache is by using direct cache mapping with contiguous cache locations. Hennessy 2nd describes that when a cache structure is direct mapped, "each [main] memory location is mapped to exactly one location in the cache. The typical mapping between addresses and cache locations for a direct-mapped cache is usually simple. For example, almost all direct-mapped caches use the mapping: (Block address) modulo (Number of cache blocks in the cache)." Ex. 1019, 546. This is the

[6] The operator MOD stands for *modulo* and is also often written as "%" in computer languages, and this calculation would appear as: main_memory_block_address % cache_size. Ex. 1023, 397.

same formula for the manner of direct mapping discussed above with respect to Hennessy 3rd.

135. Hennessy 3rd further explains the benefits of direct mapping, explaining "[a] benefit of direct-mapped placement is that hardware decisions are simplified – in fact, so simple that there is no choice: Only one block frame is checked for a hit, and only that block can be replaced." Ex. 1023, 399. In addition, Hennessy 2nd explains that "[i]n a direct-mapped cache…only a single comparator is needed, because the entry can be in only one block, and we access the cache simply by indexing…[t]he costs of an associative cache are the extra comparators and any delay imposed by having to do the compare and select from among the elements of the set." Ex. 1019, 574-575. Thus, Hennessy 2nd explains the cost effectiveness of direct mapped caching and the extra components and delay that is avoided using the approach used in Bourekas. *Id*.

136. Irie in view of Bourekas therefore teaches storing a program "*at addresses in main memory that have been selected so that*" all instructions can be stored at contiguous memory address locations in cache memory at the same time.

1. **Motivation to Combine**

137. In my opinion, a POSA would have been motivated to combine Irie and AAPA with Bourekas. The combination involves combining prior art elements according to known methods to yield predictable results.

138. Bourekas discloses a technique, that is known, of storing a program at selected addresses in main memory that have been mapped to specific contiguous addresses of cache memory. Ex. 1005, 3:65-4:2, Fig. 3; 4:64-5:3, Fig. 4. Bourekas performs this to accomplish Irie's goal of moving an entire interrupt processing program from main memory to a specific part (cache area 11A) of cache memory. *Id.*; Ex. 1002, ¶¶ 0027-0032. Thus, a POSA would have been motivated to make this combination. This is because the combination uses prior art memory elements according to known methods. The combination would have yielded predictable results: main memory having addresses mapped to a portion of contiguous addresses in cache memory.

139. In my opinion, a POSA would have combined the elements of Irie and AAPA with Bourekas by known methods, and in the combination, each element performs the same function as it does separately. Irie describes CPU 10, cache memory 11, and main memory (ROM 21 and RAM 22) in a system, and that these components interact with each other in a normal operating mode. *See S*ections IX.A.1.a, IX.A.1.b (limitations 7[a] and 7[b]).

Bourekas similarly describes interaction between a CPU, main memory, and cache memory. Ex. 1005, 3:46-55. If Bourekas' functionality were applied and used in Irie, Irie's system would include selecting addresses in ROM 21 at which to locate Irie's interrupt processing program and also specifying how these addresses map to contiguous addresses in cache area 11A of Irie's cache memory 11. A POSA would have had knowledge to make this adjustment and the adjustment would have only involved minor modifications to Irie's design using known conventional memory and logic techniques. Modifying Irie with Bourekas would have been performed by known, conventional methods of memory design and logic design because the hardware of Irie and Bourekas (cache memory, main memory) perform the same functionality (caching, program storage). Ex. 1002, ¶¶ 0009-0013. The combination would just provide a specified mapping between addresses in ROM 21 and contiguous addresses in cache memory 11 and also specify the selected addresses where the interrupt program is stored in ROM 21. *Id.*

140. Program content would still be copied from ROM 21 to a temporary location, and then to cache memory 11 as Irie alone explains, but in the combination the contiguous addresses of cache memory 11 (cache area 11A) and ROM 21 would be mapped to each other as taught by Bourekas. Storing content in one part of ROM 21 would have resulted in that content

being placed in Irie's temporary location, and then copied to contiguous addresses within a certain portion of cache memory 11, specifically, cache area 11A. Ex. 1002, ¶¶ 0024-0032.

141. Predictable results would have occurred by the combination, and the combination would have had a reasonable likelihood of success. This is because there is considerable overlap between the references. The sequential copying of program instruction content from a start address to an end address in ROM 21 (main memory) to a cache memory 11 via a temporary location in RAM 22 is described by Irie. Ex. 1002, ¶¶ 0024-0032. In Bourekas, sequential, contiguous main memory addresses mapped to certain parts of cache memory such that program content stored in selected main memory addresses are copied to cache memory and stored together at contiguous addresses in certain parts of cache (cache index range 000-011). Ex. 1005, 3:65-4:20. Modifying Irie's system would have therefore provided that sequential addresses of ROM 21 are mapped to contiguous addresses in cache memory 11. Contiguous addresses in ROM 21 would store the processing program of Irie, and these addresses would have been selected so that they correspond to contiguous addresses of cache area 11A. Irie's interrupt programs would have been copied from the selected sequential ROM 21 addresses to Irie's temporary location, and then to contiguous addresses in the

cache area 11A such that the programs are stored together in contiguous memory addresses in cache memory 11 at the same time. A POSA would have readily recognized this result and it would have been readily apparent because Irie and Bourekas describe systems that are similar and compatible. Also, for this reason, the combination would have had a reasonable likelihood of success.

142. Indeed, Bourekas' direct mapping scheme was well known in the art, as shown by its similarity to the direct mapping scheme described by Hennessy 2nd: (Block address) modulo (Number of cache blocks in the cache). Ex. 1019, 546. Due to this similarity, a POSA would have readily known how to implement Bourekas' functionality into the system of Irie and AAPA and combine their teachings and had a reasonable likelihood of success in making the combination.

143. For instance, Bourekas explains "FIG. 2 shows a block diagram of a direct mapped cache memory according to one embodiment of the present invention. Cache memory 100 uses a physical address latch 110 to receive a physical address from the CPU. The physical address is divided into a tag portion and an index portion. Physical address latch 110 stores the address received from the CPU and sends the address to a multiplexing circuit 120, which exchanges a physical address tag bit with a physical address index bit

to generate the cache address to access a cache RAM 130." Ex. 1005, 3:45-55. Bourekas further explains that, regarding its "direct mapped cache," "[t]he programmer can treat this embodiment as a cache RAM divided into 2 equal portions, each portion servicing a contiguous half of the physical address range. Thus, the programmer can store in the lower-order half of the physical address range programs to be locked in the cache RAM, while storing other programs contiguously in the upper order of the physical address range, which are serviced by the upper-order portion of the cache RAM. Thus, this embodiment realizes a 'lockable' direct mapped cache without TLBs, additional sets of tag comparators, or additional page management or operating system software." *Id.*, 3:45-46, 4:10-20. Thus, Bourekas explains a direct mapping technique. *Id.*

144. Hennessy 2nd describes a direct mapped cache where using the mapping (Block address) modulo (Number of cache blocks in the cache), where "each memory location is mapped to exactly one location in the cache." Ex. 1019, 546. This disclosure of Hennessy 2nd overlaps with Bourekas because both are directed to similar direct mapping techniques; this overlap shows how Bourekas' direct mapping was well known in the art and supports

that a POSA would have known how to implement Bourekas' caching technique into Irie's system.

145. A POSA would have further understood that the combination would have provided a less complex system and increased control, which are known benefits in memory systems, and would have been motivated to combine the references for this reason as well. Ex. 1005, 4:10-20 (explaining how a programmer has control over where information is cached based on where the information is stored in main memory). Hennessy 2nd supports my opinion: "The simplest way to assign a location in the cache for each word in memory is to assign the cache location based on the address of the word in memory. This cache structure is called *direct mapped*, since each memory location is mapped to exactly one location in the cache." Ex. 1019, 548 (emphasis in original). Hennessy continues that "[t]his mapping is attractive because if the number of entries in the cache is a power of two, then modulo can be computed simply by using only the low-order $\log_2$ (cache size in blocks) bits of the address; hence the cache may be accessed directly with the low-order bits." *Id*. Bourekas explains that its direct mapping provides a "direct mapped cache without TLBs, additional sets of tag comparators, or additional page management or operating system software," which means these items are not needed and complexity is thereby reduced. Ex. 1005, 4:10-

20. Bourekas further explains that its system allows for the adjustment of bits that allow for defining "variable sizes of contiguous address spaces." *Id.*, 6:19-28. By modifying Irie with Bourekas, users would have thus had the ability to configure Irie's system such that programs stored in selected ROM 21 (main memory) locations are stored in particular contiguous addresses in a portion of cache memory 11 (*i.e.*, cache area 11A); this would have provided additional control over exactly how and where information is cached, and would have provided that the entirety of a program (such as Irie's program copied to cache memory 11) would be retained in cache memory because Bourekas' direct (one-to-one) mapping ensures that a subsequent write will not overwrite a previous write. *Id.*, 4:10-17. The combination would have therefore afforded a less complex system and would have provided that a programmer can more easily control where programs are saved. *Id.,* 2:25-3:20, 3:65-4:20.

146. In my opinion, a POSA would have been further motivated to make this combination because the combination involves using a known technique to improve similar devices in the same way.

147. As I explain in Section IX.A.1.e, Irie describes a base device that corresponds to the elements of 7[e]. *See* Section IX.A.1.e; *see also* Section IX.B.1.b. As I explain in Section IX.B.1.b., Bourekas describes a device that

includes CPU, cache memory, and main memory, which are the same components as Irie. *Id.* The Irie and Bourekas devices are thus comparable. Bourekas' device's functionality is further disclosed as providing decreased complexity and more control when caching information–an improvement to cache memory functionality. Ex. 1005, 2:25-3:20, 3:65-4:20.

148. A POSA would have applied Bourekas' cache memory functionality to Irie's system and the results would have been recognizable and predictable to a POSA. Indeed, the results would have been decreased complexity and more control when caching. Irie stores program instructions in ROM 21 (main memory) and copies them to cache memory 11. Ex. 1002, ¶¶ 0011-0014, 0024-0032. Bourekas explains that its system reduces complexity as it does not require "TLBs, additional sets of tag comparators, or additional page management or operating system software" and that its system allows for increased control as a programmer may store programs in certain main memory locations such that these programs are also stored in particular portions of cache memory. Ex. 1005, 4:10-20. A POSA would

have therefore recognized that the combination would have provided a cache that has this predictable and advantageous configuration.

149. A POSA would have therefore been motivated to combine Irie and AAPA with Bourekas.

150. Accordingly, the combination teaches limitation 7[e].

c) 7[g] "switching to the low power operating mode once it is detected that it is no longer necessary to operate in the normal operating mode, by loading the interrupt program into the cache memory from the main memory, wherein all instructions of the interrupt program are stored together in the cache memory;"

151. As I explain in section IX.B.1.b (limitation 7[e]), Irie in view of Bourekas, describes that the interrupt program is stored at addresses in main memory that have been selected so that all instructions of the interrupt program can be stored together in the cache memory. Thus, upon loading the interrupt program into the cache memory from the main memory, "*all instructions of the interrupt program are stored together in the cache memory*" as claimed.

152. Regarding the remaining recitations of limitation 7[g], Irie teaches these recitations for the same reasons as explained in section IX.A.1.g.

X. AVAILABILITY FOR CROSS-EXAMINATION

153. 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.

XI. RIGHT TO SUPPLEMENT

154. 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.

XII. JURAT

155. 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 knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Dated: June 24 , 2019

Carl Sechen

Dr. Carl Sechen

APPENDIX A

Carl Sechen

Professor
Dept. of Electrical Engineering
University of Texas at Dallas
carl.sechen@utdallas.edu
972-835-1611

Research Interests

My research interests center primarily on the design and computer-aided design of digital and analog integrated circuits. Ongoing projects include fast, accurate simulation-based timing analysis, and more accurate power estimation for digital circuits. Also, the design of secure ICs that cannot be reverse engineered and reverse engineering PCBs that are damaged and/or discarded. We are also working on the design of novel types of embeddable field-programmable circuits for design obfuscation.

Education

Ph.D., Electrical Engineering, University of California, Berkeley, 1986
Thesis: *Placement and Global Routing of Integrated Circuits Using Simulated Annealing*
Advisor: Prof. Alberto Sangiovanni-Vincentelli
M.S., Electrical Engineering, Massachusetts Institute of Technology, 1977
Advisor: Prof. Stephen Senturia
B.E.E., Electrical Engineering, University of Minnesota, 1975

Employment history

Professor, University of Texas at Dallas August 15, 2005 – present
Professor, University of Washington July 1999 – August 14, 2005
Associate Professor, University of Washington July 1992 – June 1999
Associate Professor, Yale University July 1990 – June 1992
Assistant Professor, Yale University July 1986 – June 1990

University Administrative Positions

Director of ECS Tech Support Services, UTD, June 2012 – February 2014

Honors and Awards

- *Best Paper Award* at the 2017 IEEE PhD Research in Microelectronics and Electronics Conference (PRIME), for the paper "Improved Lagrangian Relaxation-based Gate Size and VT Assignment for Very Large Circuits", Bariloche, Argentina, February 2017.
- Received the *Distinguished Teaching Award* for the Erik Jonsson School of Engineering and Computer Science, University of Texas at Dallas, 2014.
- Received the *Distinguished Teacher of the Year* Award, Dept. of Electrical Engineering, Erik Jonsson School of Engineering and Computer Science, University of Texas at Dallas, 2008.
- Elected *IEEE Fellow* in 2002
- Received the *Outstanding Research Advisor* Award, Department of Electrical Engineering, University of Washington, 2002.
- Received the *Best Project Award*, NSF Center for the Design of Digital and Analog ICs (CDADIC), July 2002.
- Received the Semiconductor Research Corporation's 2001 *SRC Inventor's Recognition Award*
- Received the Semiconductor Research Corporation's 1994 *SRC Technical Excellence Award*
- Received the Semiconductor Research Corporation's 1988 *SRC Inventor's Recognition Award*

Graduated Ph.D. Students

1. Kai-Win Lee (Yale May 1990) "Global Routing of Row-Based Integrated Circuits".
2. Dahe Chen (Yale May 1992) "Mickey: A Graph-Based Macro-Cell Global Router".
3. Mark Chiang (Yale May 1992) "A Perturbation Approach to the Symbolic Analysis of Analog Circuits".

4. William Swartz (Yale May 1993) “Automatic Layout of Analog and Digital Mixed Macro/Standard Cell Integrated Circuits”.
5. Ted Stanion (Yale May 1994) “Boolean Algorithms for Combinational Synthesis and Test Generation”.
6. Kalapi Roy (University of Washington June 1994) “A Timing-Driven Multi-Way Partitioning System for Integrated Circuits and Multi-Chip Systems”.
7. Jer-Jaw Hsu (University of Washington December 1994) “Fully Symbolic Analysis of Large Analog Integrated Circuits”.
8. Wern-Jieh Sun (University of Washington December 1994) “Effective and Efficient Placement for Very Large Integrated Circuits”.
9. Qicheng Yu (University of Washington March 1995) “Approximate Symbolic Analysis of Large Analog Integrated Circuits”.
10. Bingzhong David Guan (University of Washington August 1996) “Automatic Layout Generation of Static CMOS Combinational Cells and Blocks”.
11. Eugene Liu (University of Washington, December 1997) “Global Routing and Pin Assignment for Multi-layer Chip-level Layout”.
12. Hsiao-Ping Tseng (University of Washington, December 1997) “Detailed Routing Algorithms for VLSI Circuits”.
13. Gin Yee (University of Washington, June 1999) “Dynamic Logic Design and Synthesis Using Clock-Delayed Domino”.
14. Tyler Thorp (University of Washington, December 1999), “Design and Synthesis of Dynamic Circuits”.
15. Tatjana Serdar (University of Washington, December 2000), “Automatic Datapath Tile Placement and Routing”.
16. Jovanka Ciric (University of Washington, August 2001), “Boolean Matching and Level-Based Technology Mapping”.
17. Yi Han (University of Washington, December 2004), “A High-Performance CMOS Programmable Logic Core for System-on-Chip Applications”.
18. Hiran Tennakoon (University of Washington, August 2005), “Efficient and Accurate Gate Sizing With Piecewise Convex Delay Models”.
19. Miodrag Vujkovic (University of Washington, March 2006), “Efficient Fully-Automated, Refinement-Based Power-Delay Optimization Design Flow for Standard Cell Designs”.
20. Kian Hour (Alfred) Chong (University of Washington, June 2006), “Self-Calibrating Differential Output Prediction Logic”.
21. Xinyu (Sunny) Guo (University of Washington, June 2006), “ A High-Throughput Divider Based on Output Prediction Logic”.
22. Sheng Sun (University of Washington, August 2006), “High Performance and Energy Efficient Adder Design”.
23. Mohammad Rahman (UT-Dallas, December 2011), “Power and Leakage Minimization for Digital ICs”.
24. Chiu-Wei Pan (UT-Dallas, August 2012), “High Speed and Power Efficient Compression of Partial Products”.
25. Zhao Wang (UT-Dallas, October 2012), “Accurate Wire Endpoint Delay Estimation”.
26. Akshay Sridharan (UT-Dallas, October 2015), “STARK: Synchronous to Asynchronous Redesign Kit”.
27. Anitha Yella (UT-Dallas, January 2016), “Power Optimization in ICs”.
28. Meisam Roshan (UT-Dallas, September 2016), “A MEMS-Assisted Dual-Resonator Temperature-to-Digital Converter”.
29. Huihua (Helen) Huang (UT-Dallas, May 2018), “A 0.1ps Resolution Coarse-Fine Time-to-Digital Converter with 2.21ps Single-Shot Precision”.

Current Ph.D. Students

1. Vahid Moalemi, Ph.D. expected 12/20
2. Jingsheng Tian, Ph.D. expected 08/19
3. Xiangyu Xu, Ph.D. expected 12/20
4. Lubaba Nahar, Ph.D. expected 12/20

5. Thomas Broadfoot, Ph.D. expected 12/21
6. Bo Hu, Ph.D. expected 12/20
7. Qiongdan (Olivia) Huang, Ph.D. expected 12/21

Teaching Activities

Yr	*Qtr*	*Course*	*Brief Title, Credits, #students*	*Stud. Opin. Survey*
1992	Aut	EE538	Auto Layout, 4 credits, 20 students	4.38/4.08
1993	Win	EE356	Analog ICs, 4 credits, 55 students	3.21/3.00
1993	Spr	EE535	VLSI Design, 4 credits, 25 students	4.27/4.47
1993	Sum	EE332	Analog ICs, 5 credits, 35 students	3.57/3.48
1993	Aut	EE538	Auto Layout, 4 credits, 15 students	3.60/3.50
1993	Aut	EE332	Analog ICs, 5 credits, 40 students	3.93/3.90
1994	Win	EE433	Analog MOS ICs, 4 credits, 65 students	3.96/4.08
1994	Spr	EE476	Digital ICs, 5 credits, 55 students	3.89/3.89
1994	Aut	EE538	Auto Layout, 4 credits, 15 students	3.63/3.63
1994	Aut	EE433	Analog MOS ICs, 4 credits, 55 students	4.32/4.30
1995	Win	EE476	Digital ICs, 5 credits, 85 students	3.75/3.85
1995	Spr	EE473	Adv. Analog MOS, 5 credits, 45 students	3.53/3.59
1995	Aut	EE476	Digital ICs, 5 credits, 85 students	4.40/4.20
1996	Win	EE535	VLSI Design, 4 credits, 55 students	4.10/4.00
1996	Spr	EE538	Auto Layout, 4 credits, 15 students	4.30/4.50
1996	Aut	EE476	Digital ICs, 5 credits, 125 students	3.40/3.40
1997	Win	EE477	Custom Dig ICs, 4 credits, 45 students	4.30/4.10
1997	Win	EE541	Auto Layout, 4 credits, 20 students	4.70/4.70
1997	Spr	EE535	VLSI Design, 4 credits, 20 students	3.83/4.00
1997	Aut	EE476	Digital ICs, 5 credits, 60 students	3.88/3.89
1998	Win	EE477	Custom Dig ICs, 4 credits, 60 students	3.66/3.61
1998	Spr	EE476	Digital ICs, 5 credits, 60 students	3.39/3.50
1998	Spr	EE535	Digital VLSI Design, 4 credits, 35 stud.	3.57/3.94
1998	Aut	EE476	Digital ICs, 5 credits, 35 students	4.15/4.50
1999	Win	EE477	Custom Dig ICs, 4 credits, 55 students	4.00/3.94
1999	Spr	EE535	Digital VLSI Design, 4 credits, 35 stud.	3.79/4.13
1999	Aut	EE476	Digital ICs, 5 credits, 50 students	4.70/4.70
2000	Win	EE477	Custom Dig ICs, 4 credits, 55 students	4.53/4.27
2000	Spr	EE535	Digital VLSI Design, 4 credits, 35 stud.	4.35/4.44
2000	Aut	EE476	Digital ICs, 5 credits, 125 students	4.20/4.19
2001	Win	EE477	VLSI II, 5 credits, 75 students	4.30/4.37
2001	Win	EE525	VLSI II, 5 credits, 25 students	4.25/4.50
2001	Spr	EE526	VLSI III, 4 credits, 50 students	4.35/4.50
2001	Aut	EE476	VLSI I, 5 credits, 150 students	4.1/4.2
2002	Win	EE477	VLSI II, 5 credits, 75 students	4.1/4.1
2002	Win	EE525	VLSI II, 5 credits, 25 students	4.1/4.1
2002	Spr	EE526	VLSI III, 4 credits, 50 students	3.6/3.3
2002	Aut	EE476	VLSI I, 5 credits, 150 students	3.7/3.6
2003	Win	EE477	VLSI II, 5 credits, 55 students	3.4/3.8
2003	Win	EE525	VLSI II, 5 credits, 25 students	3.6/3.3
2003	Spr	EE526	VLSI III, 4 credits, 32 students	3.3/3.5
2003	Aut	EE476	VLSI I, 5 credits, 30 students	4.5/4.6
2004	Win	EE477	VLSI II, 5 credits, 12 students	3.2/3.7
2004	Win	EE525	VLSI II, 5 credits, 7 students	2.2/2.2
2004	Spr	EE526	VLSI III, 4 credits, 12 students	3.2/3.2
2004	Aut	EE476	VLSI I, 5 credits, 65 students	3.8/3.9
2005	Fall	EE6325	VLSI Design, 3 units, 42 students	4.4/4.6
2006	Spr	EE7325	Advanced VLSI Design, 3 units, 20 students	4.5/4.3
2006	Fall	EE6325	VLSI Design, 3 units, 66 students	4.4/4.2
2007	Spr	EE7325	Advanced VLSI Design, 3 units, 15 students	4.5/4.2

2007	Fall	EE6325	VLSI Design, 3 units, 40 students	4.8/4.6
2008	Spr	EE4325	Introduction to VLSI Design, 40 students	3.5/3.4
2008	Spr	EE7325	Advanced VLSI Design, 3 units, 19 students	4.6/4.4
2008	Sum	EE6325	VLSI Design, 3 units, 36 students	4.0/3.7
2008	Fall	EE6325	VLSI Design, 3 units, 40 students	4.1/4.1
2009	Spr	EE4325	Introduction to VLSI Design, 22 students	4.2/3.9
2009	Spr	EE7325	Advanced VLSI Design, 3 units, 37 students	3.7/3.6
2009	Sum	EE6325	VLSI Design, 3 units, 17 students	4.1/4.0
2009	Fall	EE6325	VLSI Design, 3 units, 67 students	4.3/4.0
2010	Spr	EE4325	Introduction to VLSI Design, 25 students	4.6/4.5
2010	Spr	EE7325	Advanced VLSI Design, 3 units, 35 students	4.4/4.4
2010	Sum	EE6325	VLSI Design, 3 units, 17 students	4.7/4.5
2010	Sum	EE3320	Digital Circuits, 3 units, 17 students	4.1/3.8
2010	Fall	EE6325	VLSI Design, 3 units, 77 students	4.4/4.2
2011	Spr	EE4325	Introduction to VLSI Design, 37 students	4.0/4.1
2011	Spr	EE7325	Advanced VLSI Design, 3 units, 17 students	4.2/4.2
2011	Sum	EE6325	VLSI Design, 3 units, 16 students	4.14
2011	Sum	EE3320	Digital Circuits, 3 units, 26 students	4.13
2011	Fall	EE6325	VLSI Design, 3 units, 91 students	4.73
2012	Spr	EE4325	Introduction to VLSI Design, 27 students	4.25
2012	Spr	EE7325	Advanced VLSI Design, 3 units, 17 students	4.90
2012	Sum	EE6325	VLSI Design, 3 units, 35 students	4.46
2012	Sum	EE3311	Electronic Circuits, 3 units, 22 students	3.62
2012	Fall	EE6325	VLSI Design, 3 units, 144 students	4.76
2013	Spr	EE6325	VLSI Design, 3 units, 41 students	4.85
2013	Spr	EE4325	Introduction to VLSI Design, 7 students	4.75
2013	Sum	EE7325	Advanced VLSI Design, 3 units, 67 students	4.38
2013	Sum	EE3311	Electronic Circuits, 3 units, 46 students	4.58
2013	Fall	EE6325	VLSI Design, 3 units, 129 students	4.64
2014	Spr	EE4325	Introduction to VLSI Design, 25 students	4.69
2014	Sum	EE7325	Advanced VLSI Design, 3 units, 74 students	4.53
2014	Sum	EE3311	Electronic Circuits, 3 units, 40 students	4.75
2014	Fall	EE6325	VLSI Design, 3 units, 139 students	4.79
2015	Spr	EE4325	Introduction to VLSI Design, 25 students	4.72
2015	Sum	EE7325	Advanced VLSI Design, 3 units, 49 students	4.25
2015	Sum	EE3311	Electronic Circuits, 3 units, 25 students	4.80
2015	Fall	EE6325	VLSI Design, 3 units, 142 students	4.73
2016	Spr	EE4325	Introduction to VLSI Design, 25 students	3.67
2016	Sum	EE7325	Advanced VLSI Design, 3 units, 70 students	4.77
2016	Sum	EE3311	Electronic Circuits, 3 units, 35 students	4.88
2016	Fall	EE6325	VLSI Design, 3 units, 140 students	4.79
2017	Spr	EE4325	Introduction to VLSI Design, 32 students	4.10
2017	Sum	EE7325	Advanced VLSI Design, 3 units, 30 students	4.88
2017	Sum	EE3311	Electronic Circuits, 3 units, 32 students	4.50
2017	Fall	EE6325	VLSI Design, 3 units, 80 students	4.82
2018	Spr	EE4325	Introduction to VLSI Design, 32 students	4.63
2018	Sum	EE7325	Advanced VLSI Design, 3 units, 30 students	4.88
2018	Fall	EE6325	VLSI Design, 3 units, 80 students	4.88

Course Development

U. of Washington: EE 541 Automatic Layout of Integrated Circuits (first offering: Fall 1992)
U. of Washington: EE 477 Custom Digital CMOS Circuit Design (first offering: Win 1997)
U. of Washington: EE 476 Digital Integrated Circuit Design (first offering: Spr 1994)

Yale University: EE 880 Automatic Layout of Integrated Circuits.
Yale University: EE 877 Introduction to VLSI CAD Tools

Yale University: EE 878 Introduction to VLSI Synthesis
Yale University: EE 988 Analog Integrated Circuits II

Journal Publications

1. Roshan, C. Sechen, *et al*., "A MEMS-Assisted Temperature Sensor with 20μK Resolution, Conversion Rate of 200S/s and FOM of 0.04pJK", *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 185-197, January 2017.
2. Z. Wang, X. He, and C. Sechen, "A New Approach for Gate-Level Delay-Insensitive Asynchronous Logic", *Journal of Circuits, Systems & Signal Processing*, October 2014.
3. Z. Wang, C. Pan, Y. Song, and C. Sechen, "High-Throughput Digital IIR Filter Design", *Journal of Algorithms and Optimization (JAO)*, Vol. 2, No. 2, pp. 15-27, April 2014.
4. C. Pan, Z. Wang and C. Sechen, "High Speed and Power Efficient Compression of Partial Products and Vectors," *Journal of Algorithms and Optimization (JAO)*, Vol. 1, No. 1, pp. 39-54, October 2013.
5. Z. Wang and C. Sechen, "A New Algorithm for Accurate Wire Endpoint Delay Estimation", *Journal of Algorithms and Optimization (JAO)*, Vol. 2, No. 1, pp. 30-43, Jan. 2014.
6. M. Rahman, H. Tennakoon, and C. Sechen, "Library-Based Cell-Size Selection using Extended Logical Effort", *IEEE Trans. on Computer-Aided Design*, July 2013.
7. H. Tennakoon and C. Sechen, "Non-convex Gate Delay Modeling and Delay Optimization," *IEEE Trans. on Computer-Aided Design*, Vol. 27, No. 9, September 2008, pp. 1583-1594.
8. Don Bouldin, Warren Snapp, Paul Haug, Dave Sunderland, Roger Brees, Carl Sechen, and Wayne Dai, "Automated Design of Digital Signal Processing ASICs," *IEEE Circuits and Devices*, vol. 20, n. 4, pp. 17-21, July 2004.
9. J. Ciric and C. Sechen, "Efficient Canonical Form for Boolean Matching of Complex Functions in Large Libraries," *IEEE Trans. on Computer-Aided Design*, Vol. 22, No. 5, May 2003, pp. 535-544.
10. T. Thorp and C. Sechen, "Design and Synthesis of Dynamic Circuits," *IEEE Transactions on VLSI Systems*, Vol. 11, No. 1, February 2003, pp. 141-149.
11. G. Hoyer, G. Yee and C. Sechen, "Locally-Clocked Pipelines and Dynamic Logic," *IEEE Transactions on VLSI Systems*, Vol. 10, No. 1, February 2002, pp. 58-62.
12. H. P. Tseng, L. Scheffer and C. Sechen, "Timing- and Crosstalk-Driven Area Routing," *IEEE Transactions on Computer Aided Design*, vol. 20, no. 4, pp. 528-544, April 2001.
13. G. Yee and C. Sechen, "Clock-Delayed Domino for Dynamic Circuit Design," *IEEE Transactions on VLSI Systems*, Vol. 8, No. 4, August 2000, pp. 425-431.
14. L. Liu and C. Sechen, "Multi-layer Chip-level Global Routing Using an Efficient Graph-based Steiner Tree Heuristic," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, Vol. 18, No. 10, October 1999.
15. L. Liu and C. Sechen, "Multi-layer Pin Assignment for Macro Cell Circuits", *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, Vol. 18, No. 10, October 1999.
16. H. P. Tseng and C. Sechen, "A Gridless Multi-Layer Router for Standard Cell Circuits using CTM Cells," *IEEE Transactions on Computer Aided Design*, Vol. 18, No. 10, October 1999.
17. W. Sun and C. Sechen, "A Parallel Standard Cell Placement Algorithm," *IEEE Transactions on Computer Aided Design*, vol. 16, no. 11, November 1997, pp. 1342-1357.
18. Q. Yu and C. Sechen, "Efficient Approximation of Symbolic Network Functions Using Matroid Intersection Algorithms," *IEEE Transactions on Computer Aided Design*, vol. 16, no. 10, October 1997, pp. 1073-1081.
19. Q. Yu and C. Sechen, "Generation of Color-Constrained Spanning Trees with Application in Symbolic Analysis," *Int. Journal of Circuit Theory and Applications*, Vol. 24, No. 5, pp. 597-603, 1996.
20. Q. Yu and C. Sechen, "A Unified Approach to the Approximate Symbolic Analysis of Large Analog Integrated Circuits," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, Vol. 43, No. 8, pp. 656-669, August 1996.
21. K. Roy and C. Sechen, "A Timing-Driven Partitioning System for Multiple FPGAs," *Int. Journal of VLSI Design*, Vol. 4, No. 4, pp. 309-328, 1996.

22. K. Roy, D. Guan and C. Sechen, "A Sea-of-Gates Style FPGA Placement Algorithm," *Int. Journal of VLSI Design*, Vol. 4, No. 4, pp. 293-307, 1996.
23. T. Stanion, D. Bhattacharya, and C. Sechen, "An Efficient Method for Generating Exhaustive Test Sets," *IEEE Transactions on Computer-Aided Design*, Vol. 14, No. 12, December 1995.
24. W. Sun and C. Sechen, "Efficient and Effective Placement for Very Large Circuits," *IEEE Transactions on Computer-Aided Design*, Vol. 14, No. 3, pp. 349-359, March 1995.
25. J. J. Hsu and C. Sechen, "DC Small Signal Symbolic Analysis of Large Analog Integrated Circuits," *IEEE Transactions on Circuits and Systems I*, Vol. 41, No. 12, pp. 817-828, December 1994.
26. T. Stanion and C. Sechen, "Boolean Division and Factorization Using Binary Decision Diagrams," *IEEE Transactions on Computer-Aided Design*, vol. 13, no. 9, September 1994, pp. 1179-1184.
27. C. Sechen, "Chip-Planning, Placement, and Global Routing of Macro Cell Integrated Circuits Using Simulated Annealing," *Int. J. Computer-Aided VLSI Design*, vol. 2, no. 2, 1990, pp. 127-158.
28. C. Sechen, D. Braun, and A. Sangiovanni-Vincentelli, "ThunderBird: A Complete Standard Cell Layout Package," *IEEE J. of Solid State Circuits*, vol. 23, n. 2, pp. 410- 420, April 1988.
29. C. Sechen and A. Sangiovanni-Vincentelli, "The TimberWolf Placement and Routing Package," *IEEE J. of Solid State Circuits*, vol. 20, n. 2, April 1985, pp. 510-522.
30. C. Sechen, "An Improved Lock Layer Transistor," *Solid State Electronics*, Vol. 21, No. 6, June 1978, pp. 911-913.
31. S. Senturia and C. Sechen, "The Use of the Charge-Flow Transistor to Distinguish Surface and Bulk Components of Thin-Film Sheet Resistance," *IEEE Transactions on Electron Devices,* vol.24, no.9, p.1207, September 1977.
32. S. Senturia, C. Sechen, and J. Wishneusky, "The Charge-Flow Transistor: A New MOS Device," *Applied Physics Letters*, Vol. 30, No. 2, Jan. 1977, p. 106-108.

Fully Refereed Conference Proceedings

1. S. Sapatnekar, S. Reda, C. Sechen, R Dreslinski, et al., "OpenROAD: Toward a Self-Driving, Open-Source Digital Layout Implementation Tool Chain," *Proc. Government Microcircuit Applications and Critical Technology Conference, (GOMACTech)*, March 25-28, 2019, Albuquerque, NM.
2. M. Shihab, G. Reddy, J. Tian, B. Hu, W. Swartz, B. Carrion Schaefer, C. Sechen, Y. Makris, "Design Obfuscation through Selective Post-Fabrication Transistor-Level Programming," *Proc. of Design Automation and Test in Europe (DATE) Conference*, March 25-29, 2019, Florence, Italy.
3. A. Yella, S. Gunturi and C. Sechen, "Are Standalone Gate Size and VT Optimization Tools Useful?", *Proc. IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE), Devices, Circuits, and Systems Track*, Windsor, ON, Canada, April 30 - May 3, 2017.
4. T. Broadfoot, C. Sechen and J. Rajendran, "On Designing Optimal Camouflaged Layouts", *Proc. IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, May 1-4, 2017, McLean, VA.
5. A. Yella and C. Sechen, "Improved Lagrangian Relaxation-based Gate Size and VT Assignment for Very Large Circuits, *Proc. IEEE PhD Research In Microelectronics and Electronics Conference Latin America (PRIME),* Bariloche, Argentina, February 20-23, 2017. **BEST PAPER AWARD**
6. J. Tian, G. Reddy, J. Wang, W. Swartz, Y. Makris and C Sechen, "A Field Programmable Transistor Array Featuring Single-Cycle Partial/Full Dynamic Reconfiguration", *Proc. of Design Automation and Test in Europe (DATE) Conference*, March 27-31, 2017, Lausanne, Switzerland.
7. M. Roshan, C. Sechen, *et al*., "Dual-MEMS Resonator Temperature-to-Digital Converter with 39µK Resolution and a FoM of 0.11pJK^2", *Proc. Int. Solid-State Circuits Conf. (ISSCC),* Feb. 2016, San Francisco, CA.
8. M. Jain and C. Sechen, "Chip Design: Matrix Multiplier Based on Systolic Architecture", *IEEE Computer Society 6th International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, July 13-15, 2015, Dallas, TX.
9. Z. Wang, X. He and C. Sechen, "TonyChopper: A Desynchronization Package", *Proc. Int. Conf. on Computer Aided Design (ICCAD),* Nov. 2-6, 2014, San Jose, CA. (20% accepted)
10. H. Huang and C. Sechen, "A 14-b, 0.1ps Resolution Coarse-Fine Time-to-Digital Converter in 45 nm CMOS", *2014 Dallas Circuits and Systems Conference (DCAS)*, Oct 12-13, 2014, Dallas, TX.

11. A. Sridharan and C. Sechen, "Minimum Energy Operation using Robust Asynchronous Logic with Sleep Transistors", *Proc. Dallas Circuits and Systems Conference (DCAS)*, Oct 12-13, 2014, Dallas, TX.
12. Yashaswini Prathivadi, Carl Sechen, and Roozbeh Jafari, "A Swarm of Wearable Sensors at the Edge of the Cloud for Robust Activity Recognition", *Proc*. *First International Workshop on the Swarm at the Edge of the Cloud (SEC'13)*, Sept. 29, 2013, Montreal, Canada.
13. Mohammad-Mahdi Bidmeshki, Carl Sechen, and Roozbeh Jafari, "Low Power Programmable Architecture for Periodic Activity Monitoring", *Proc. SRC Techcon*, 2013, Austin, TX.
14. Thomas Broadfoot, Jingxiang Tian, HeeEun Choi, Mohammad-Mahdi Bidmeshki, Roozbeh Jafari, and Carl Sechen, "Platform Design for Swarm Wearable Computing", *Proc*. *First International Workshop on the Swarm at the Edge of the Cloud (SEC'13),* Sept. 29, 2013, Montreal, Canada.
15. A. Sridharan, C. Sechen, and R. Jafari, "Low-Voltage Low-Overhead Asynchronous Logic", *Int. Symp. on Low Power Electronics and Design (ISLPED)*, Beijing, China, Sept. 4-6, 2013. (25% accepted)
16. M. M. Bidmeshki, C. Sechen and R. Jafari, "Low Power Programmable Architecture for Periodic Activity Monitoring," *IEEE Texas Workshop on Integrated System Exploration (TexasWISE)*, 8 March 2013, Round Top, Texas.
17. Hee-Eun Choi, C. Sechen and R. Jafari, "24kb, 11T Subthreshold SRAM for Ultra-Low Power Medical and Health Monitoring Embedded Systems," *IEEE Texas Workshop on Integrated System Exploration (TexasWISE)*, 8 March 2013, Round Top, Texas.
18. M. Rahman, and C. Sechen, "Post-Synthesis Leakage Power Minimization", *Design Automation and Test in Europe (DATE) Conference*, March 12-16, 2012, Dresden, Germany.
19. M. Rahman, H. Tennakoon, R. Afonso and C. Sechen, "Power Reduction via Separate Synthesis and Physical Libraries", *Proc. Design Automation Conference (DAC)*, June 2011, San Diego, CA.
20. C. Pan and C. Sechen, "Power Efficient Partial Product Compression", *Proc. Great Lakes Symposium on VLSI*, Lausanne, May 2-4, 2011, Switzerland.
21. M. Rahman, H. Tennakoon, and C. Sechen, "Power Reduction via Near-Optimal Library-Based Cell-Size Selection", *Design Automation and Test in Europe (DATE) Conference*, March 14-18, 2011, Grenoble, France.
22. M. Rahman, H. Tennakoon, R. Afonso, and C. Sechen, October 17-18, "Design Automation Tools and Libraries for Low Power Digital Design", *Proc. IEEE Dallas Circuits and Systems Conference (DCAS)*, Oct. 17-18, 2010, Richardson, TX.
23. M. Rahman, H. Tennakoon, and C. Sechen, "Near Optimal Power Efficiency Through Gate Sizing", *Proc. Austin Conf. on Integrated Circuits and Systems (ACISC),* Oct. 26-27, 2009, Austin, TX.
24. H. Huang and C. Sechen, "A 22mW 227Msps 11b Self-Tuning ADC Based on Time-to-Digital Conversion", *Proc. Austin Conf. on Integrated Circuits and Systems (ACISC),* Oct. 26-27, 2009, Austin, TX.
25. E. Kiefer, W. Swartz, and C. Sechen, "Low Power Automated Clock Tree Generation", *Proc. Austin Conf. on Integrated Circuits and Systems (ACISC),* Oct. 26-27, 2009, Austin, TX.
26. R. Afonso, M. Rahman, H. Tennakoon, and C. Sechen, "Power Efficient Standard Cell Library Design", *Proc. IEEE Dallas Circuits and Systems Society Workshop*, Oct. 4-5, 2009, Dallas, TX.
27. C.W. Pan, Y. Song, Z. Wang, and C. Sechen, "DSP Power Reduction through Generalized Carry-Save Arithmetic", *Proc. IEEE Dallas Circuits and Systems Society Workshop*, Oct. 4-5, 2009, Dallas, TX.
28. R. Afonso, M. Rahman, H. Tennakoon, and C. Sechen, "Power Efficient Standard Cell Library Design", *Proc. Austin Conf. on Integrated Circuits and Systems (ACISC),* Oct. 26-27, 2009, Austin, TX.
29. C.W. Pan, Y. Song, Z. Wang, and C. Sechen, "DSP Power Reduction through Generalized Carry-Save Arithmetic", *Proc. Austin Conf. on Integrated Circuits and Systems (ACISC),* Oct. 26-27, 2009, Austin, TX.
30. E. Kiefer, W. Swartz, and C. Sechen, "Low Power Automated Clock Tree Generation", *Proc. IEEE Dallas Circuits and Systems Society Workshop*, Oct. 4-5, 2009, Dallas, TX.
31. H. Huang and C. Sechen, "A 22mW 227Msps 11b Self-Tuning ADC Based on Time-to-Digital Conversion", *Proc. IEEE Dallas Circuits and Systems Society Workshop*, Oct. 4-5, 2009, Dallas, TX.
32. S. Sun and C. Sechen, "Post-Layout Comparison of High Performance 64b Static Adders in Energy-Delay Space," *Proc. IEEE Int. Conf. on Computer Design (ICCD)*, Lake Tahoe, CA, October 2007.

33. M. Rahman, H. Tennakoon, and C. Sechen, "Optimal Area-versus-Delay Circuit Sizing Using Extended Logical Effort", *Proc. Austin Conf. on Integrated Circuits and Systems (ACISC),* May 14-15, 2007, Austin, TX.
34. K. H. Chong, L. McMurchie and C. Sechen, "A 64b Adder Using Self-calibrating Differential Output Prediction Logic," *Proc. Int. Solid-State Circuits Conference (ISSCC)*, February 2006, San Francisco, CA.
35. J. Zhang, M. Vujkovic, D. Wadkins, and C. Sechen, "Post-Layout Energy-Delay Analysis of Parallel Multipliers," *Proc. Int. Symp. on Circuits and Systems (ISCAS)*, May 2006, Greece.
36. M. Vujkovic, D. Wadkins and C. Sechen, "Efficient Post-Layout Power-Delay Curve Generation," *Prof. 15th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 21-23 September 2005, Leuven, Belgium. In: *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation,* Series: Lecture Notes in Computer Science, Vol. 3728 Paliouras, Vassilis; Vounckx, Johan; Verkest, Diederik (Eds.) 2005, XV, 753 p., Softcover ISBN: 3-540-29013-3.
37. X. Guo and C. Sechen, "High Throughput Divider Using Output Prediction Logic," *Proc. PRIME (Ph.D. Research in Microelectronics) Conf., July 25-28, 2005, Lausanne, Switzerland.*
38. X. Guo and C. Sechen, "A High Throughput Divider Implementation," *Proc. Custom Integrated Circuits Conference (CICC)*, 18-21 September 2005, San Jose, CA.
39. H. Tennakoon and C. Sechen, "Efficient and Accurate Gate Sizing with Piecewise Convex Delay Models," *Proc. Design Automation Conference (DAC)*, Anaheim, CA, June 13-17, 2005.
40. S. Sun, Y. Han, X. Guo, K. Chong, L. McMurchie and C. Sechen, "409ps 4.7 FO4 64b Adder Based on Output Prediction Logic in 0.18um CMOS," *Proc. IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, May 11-12, 2005, Tampa, FL.
41. X. Guo and C. Sechen, "High Speed Redundant Adder and Divider in Output Prediction Logic," *Proc. IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, May 11-12, 2005, Tampa, FL.
42. J. Kim, L. McMurchie and C. Sechen, "Mitigation of Single- and Multiple-Cycle-Duration SETs using Double-Mode Redundancy (DMR) in Time", *Proc. IEEE Aerospace Conference*, Big Sky, Montana, Mar. 5-12, 2005.
43. J. Lan, D. Lam, L. McMurchie, and C. Sechen, "SEE-Hardened-by-Design Area-Efficient SRAMs", *Proc. IEEE Aerospace Conference*, Big Sky, Montana, Mar. 5-12, 2005.
44. Y. Han, L. McMurchie and C. Sechen, "A High Performance CMOS Programmable Logic Core," *Proc. Custom Integrated Circuits Conference (CICC)*, October 3-6, 2004, Orlando, FL.
45. V. Tang, J. Lan, D. Lam, L. McMurchie, and C. Sechen, "High-Performance SEE-Hardened Programmable DSP Array," Proc. 2004 Military and Aerospace Programmable Logic Device (MAPLD) Conference, Washington, DC, Sept. 8-10, 2004.
46. D. Lam, J. Lan, L. McMurchie, and C. Sechen, "Radiation-Hardened-by-Design Area-Efficient SRAMs," Proc. Hardness by Design (HBD) Workshop, Albuquerque, NM, August 24-25, 2004.
47. M. Vujkovic, D. Wadkins, B. Swartz and C. Sechen, "Efficient Timing Closure Without Timing Driven Placement and Routing," *Proc. Design Automation Conference (DAC),* San Diego, CA, June 7-11, 2004.
48. Y. Lam, L. McMurchie, and C. Sechen, "SEU Hardening of Peripheral Circuitry in Self-Scrubbing SRAMs," *Proc. Single Event Effects Symposium*, Manhattan Beach, CA, April 27-29, 2004.
49. V. Tang, J. Lan, D. Lam, L. McMurchie and C. Sechen, "Low-Overhead SEE-Hardened Programmable DSP Array," *Proc. Single Event Effects Symposium*, Manhattan Beach, CA, April 27-29, 2004.
50. L. McMurchie and C. Sechen, "RADAR -- Reconfigurable Analog and Digital Array for Radiation-Hardened Circuits", *Proc. IEEE Aerospace Conference*, Big Sky, Montana, Mar. 6-13, 2004.
51. Y. Han and C. Sechen, "A Novel High Speed FPGA Architecture, *Proc. SRC Techcon*, Dallas, TX, Aug. 25 – 28, 2003.
52. C. Sechen, "The End of Libraries as We Know Them," *Proc. 40th Design Automation Conference (DAC)*, pp. 642-643, 2003.
53. S. Kio, K. Chong, and C. Sechen, "A Low Power Delayed-Clock Generation and Distribution System," *Proc. Int. Symp. on Circuits and Systems (ISCAS)*, May 25-28, 2003, Bangkok, Thailand.
54. Warren Snapp, Paul Haug, Dave Sunderland, Roger Brees Don Bouldin, Wayne Dai, and Carl Sechen, "MSP Liberator ASIC Design Flow Produces Full Custom Performance Required for Next Generation

Military Electronics*," Proc. Government Microelectronics and Applications Conference (GOMAC)*, April 1-3, 2003, Tampa, FL.

55. H. Tennakoon and C. Sechen, "Gate Sizing Using Lagrangian Relaxation Combined with a Fast Gradient-Based Preprocessing Step," *Proc. Int. Conf. on Computer-Aided Design (ICCAD)*, Nov. 2002, San Jose, CA.
56. M. Vujkovic and C. Sechen, "Optimized Power-Delay Curve Generation for Standard Cell ICs," *Proc. Int. Conf. on Computer-Aided Design (ICCAD)*, Nov. 2002, San Jose, CA.
57. L. McMurchie and C. Sechen, "WTA – Waveform-Based Timing Analysis for Deep Submicron Circuits," *Proc. Int. Conf. on Computer-Aided Design (ICCAD)*, Nov. 2002, San Jose, CA.
58. M. Vujkovic and C. Sechen, "Optimized Power-Delay Curve Generation for Standard Cell ICs," *Proc. Int. Workshop on Logic Synthesis (IWLS)*, June 4-7, 2002, New Orleans, LA.
59. Yi Han, Larry McMurchie, Jovanka Ciric, and Carl Sechen, "Giga hertz-Speed FPGA Architecture based on Output Prediction Logic", *Tenth ACM/SIGDA International Symposium on Field Programmable Gate Arrays* (FPGA 2002).
60. J. Ciric and C. Sechen, "Efficient Canonical Form for Boolean Matching of Complex Functions in Large Libraries," *Proc. of IEEE Int. Conf. on Comp. Aided Design (ICCAD)*, San Jose, CA, November 5-7, 2001.
61. R. Rutenbar, M. Baron, T. Daniel, R. Jayaraman, Z. Or-Bach, J. Rose and C. Sechen, "When Will FPGAs Kill ASICs?" *Proc. Design Automation Conference (DAC)*, June 18-22, 2001, Las Vegas, NV.
62. J. Ciric and C. Sechen, "Efficient Canonical Form for Boolean Matching of Complex Functions in Large Libraries," *Proc. of IEEE Int. Workshop on Logic Synthesis*, Lake Tahoe, CA, June 12-15, 2001.
63. S. Kio, L. McMurchie, and C. Sechen, "Application of Output Prediction Logic to Differential CMOS," *Proc. of IEEE Computer Society Annual Workshop on VLSI*, Orlando, FL, April 19-20, 2001.
64. S. Sun, L. McMurchie, and C. Sechen, "A High-Performance 64-bit Adder Implemented in Output Prediction Logic," *Proc. 19th Conference on Advanced Research in VLSI*, March 14-16, 2001, Salt Lake City, UT.
65. T. Serdar and C. Sechen, "Automatic Datapath Tile Placement and Routing," *Proc. of the Design, Automation and Test in Europe (DATE) Conference*, March 13-16, 2001, Munich, Germany.
66. S. Kio, L. McMurchie, and C. Sechen, "Output Prediction Logic", *Proc. SRC Techcon*, September 21-23, 2000, Phoenix, AZ.
67. M. Vujkovic, G. Yee, and C. Sechen, "Post-Fabrication Automatically Tunable Programmable Delay Elements for Clock-Delayed Domino Logic," *Proc. SRC Techcon*, September 21-23, 2000, Phoenix, AZ.
68. G. Hoyer and C. Sechen, "A Locally-Clocked Dynamic Logic Serial/Parallel Multiplier," *Proc. SRC Techcon*, September 21-23, 2000, Phoenix, AZ.
69. L. McMurchie, S. Kio, G. Yee, T. Thorp and C. Sechen, "Output Prediction Logic: A High Performance CMOS Design Technique", *Proc. Int. Conf. On Computer Design (ICCD)*, September 17-20, 2000, Austin, TX.
70. G. Hoyer and C. Sechen, "A Locally-Clocked Dynamic Logic Serial/Parallel Multiplier", *Proc. of Custom Integrated Circuits Conference (CICC)*, May 21-25, 2000, Orlando, FL.
71. Gin Yee, Ron Christopherson, Tyler Thorp, Ban P. Wong, and Carl Sechen, "An automated shielding algorithm and tool for dynamic circuits," *Proc. Intl. Symposium on Quality Electronic Design (ISQED)*, San Jose, CA, March 20-22, 2000.
72. J. Ciric, G. Yee, C. Sechen, "Delay Minimization and Technology Mapping of Two-Level Structures and Implementation Using Clock-Delayed Domino Logic," *Proc. Design Automation & Test in Europe (DATE2000)*, 27-30 March 2000, Paris, France.
73. T. Serdar and C. Sechen, "AKORD: Transistor Level and Mixed Transistor/Gate Level Placement Tool for Digital Data Paths," *Proc. Int. Conf. on Computer-Aided Design (ICCAD)*, San Jose, CA, November 1999.
74. T. Thorp, G. Yee and C. Sechen, "Monotonic Static CMOS and Dual Vt Technology," *Proc. Int. Symp. on Low Power Electronics and Design (ISPLED)*, San Diego, CA, August 16-17, 1999.
75. T. Thorp, G. Yee and C. Sechen, "Monotonic Logic and Dual Vt Technology", *Proc. of Int. Conf. on Computer Design (ICCD)*, Austin, TX, October 1999.
76. G. Hoyer, G. Yee and C. Sechen, "SEU Tolerant Locally-Clocked Dynamic Logic, *Proc. Government Microelectronics and Applications Conference (GOMAC)*, March 1999, Monterey, CA.

77. T. Thorp and C. Sechen, "Domino logic synthesis using complex static gates", *Proc. Int. Conf. on Computer-Aided Design (ICCAD)*, Nov. 9-11, 1998, San Jose, CA.
78. H. Tennakoon and C. Sechen, "Linear Time Heuristic for Timing Optimization", *Proc. SRC TECHCON Conf.*, Sept. 9-11, 1998, Las Vegas, NV.
79. G. Hoyer, G. Yee and C. Sechen, "Locally-clocked Dynamic Logic", *Proc. Midwest Conf. On Circuits and Systems*, Aug. 10-12, 1998, Notre Dame, IN.
80. J. Ciric and C. Sechen, "Clock-delayed Domino Synthesis Using Decomposition into Two-level Structures", *Proc. SRC TECHCON Conf.*, Sept. 9-11, 1998, Las Vegas, NV.
81. G. Hoyer and C. Sechen, "Locally-clocked Dynamic Logic", *Proc. SRC TECHCON Conf.*, Sept. 9-11, 1998, Las Vegas, NV.
82. H. P. Tseng, L. Liu and C. Sechen, "Chip-level Area Routing", *Proc. International Symposium on Physical Design (ISPD)*, April 1998, Monterey, CA.
83. H. P. Tseng, L. Scheffer and C. Sechen, "Timing and Crosstalk Driven Area Routing," *Proc. IEEE Design Automation Conference (DAC)*, June 1998, San Francisco, CA.
84. T. Thorp, G. Yee and C. Sechen, "Dynamic Logic Synthesis using Alternating Dynamic and Static Gates," *Proc. Int. Workshop on Logic Synthesis*, June 1998, Lake Tahoe, CA.
85. G. Yee, R. Chandra, H. Ganesan, and C. Sechen, "Wire Delay in the Presence of Crosstalk," *ACM/IEEE Int. Workshop on Timing Issues in the Specification and Synthesis of Digital Systems,* Dec. 4-5, 1997, Austin, Texas.
86. M. Lefebvre, D. Marple and C. Sechen, "The Future of Custom Cell Generation in Physical Synthesis," *Proc. Design Automation Conference (DAC)*, Anaheim, CA, June 1997, pp. 446-451.
87. G. Yee and C. Sechen, "Dynamic Logic Synthesis," *Proc. IEEE Custom Integrated Circuits Conference (CICC)*, May 5-8, 1997, Santa Clara, CA, pp. 345-348.
88. H. P. Tseng and C. Sechen, "Multi-layer Over-the-Cell Routing with Obstacles," *Proc. IEEE Custom Integrated Circuits Conference*, May 5-8, 1997, Santa Clara, CA, pp. 565-568.
89. L. Liu and C. Sechen, "Multi-layer Chip-Level Global Routing Using an Efficient Graph-based Steiner Tree Heuristic," *Proc. European Design and Test Conference (EDTC)*, March 18-20, 1997, Paris, France.
90. H. P. Tseng and C. Sechen, "A Gridless Multi-layer Router for Standard Cell Circuits Using Central Terminal Model Cells," *Proc. European Design and Test Conference (EDTC)*, March 18-20, 1997, Paris, France.
91. B. Guan and C. Sechen, "Large Standard Cell Libraries and Their Impact on Layout Area and Circuit Performance," *Proc. IEEE Int. Conf. on Computer Design (ICCD)*, October 7-9, 1996, Austin, TX.
92. G. Yee and C. Sechen, "Clock-delayed Domino for Adder and Random Logic Design," *Proc. IEEE Int. Conf. on Computer Design (ICCD)*, October 7-9, 1996, Austin, TX.
93. B. Guan and C. Sechen, "ASIC Automatic Layout Generation Using Large Standard Cell Libraries," *Proc. International Conference on ASIC (ASICON)*, October 21-24, 1996, Shanghai, China.
94. B. Guan and C. Sechen, "Efficient Standard Cell Generation When Diffusion Strapping Is Required," *Proc. IEEE Custom Integrated Circuits Conference (CICC)*, May 1996, San Diego, CA.
95. Q. Yu and C. Sechen, "Approximate Symbolic Analysis of Large Analog Integrated Circuits," *Proc. of the European Conference on Circuit Theory and Design (ECCTD)*, August 27-31, 1995, Istanbul, Turkey.
96. W. Swartz and C. Sechen, "Timing Driven Placement for Large Circuits," *Proc. 1995 Design Automation Conference (DAC)*, San Francisco, CA, June 1995.
97. T. Stanion and C. Sechen, "A Method for Finding Good Ashenhurst Decompositions and Its Application for FPGA Synthesis," *Proc. 1995 Design Automation Conference (DAC)*, San Francisco, CA, June 1995.
98. B. Guan and C. Sechen, "An Area Minimizing Layout Generator for Random Logic Blocks," *Proc. IEEE Custom Integrated Circuits Conference (CICC)*, May 1-4, 1995, Santa Clara, CA.
99. Q. Yu and C. Sechen, "Efficient Approximation of Symbolic Network Functions Using Matroid Intersection Algorithms," *Proc. of the IEEE Int. Symp. on Circuits and Systems (ISCAS)*, May 1-3, 1995, Seattle, WA.
100. J. Hsu and C. Sechen, "Simplified Symbolic Pole and Zero Expressions from the Sifting Approach to Symbolic Analysis," *Proc. of the IEEE Int. Symp. on Circuits and Systems (ISCAS)*, May 1-3, 1995, Seattle, WA.

101. T. Stanion and C. Sechen, "Quasi-Algebraic Decomposition of Switching Functions," *Proc. 16th Conference on Advanced Research in VLSI*, March 27-29, 1995, Chapel Hill, NC, pp. 358-367.
102. K. Roy and C. Sechen, "Multiple FPGA Partitioning with Performance Optimization," *Proc. of the IEEE/ACM International Symposium on FPGAs*, March 12-14, Monterrey, CA, 1995, pp. 146-152.
103. W. Sun and C. Sechen, "A Loosely Coupled Parallel Algorithm for Standard Cell Placement," *IEEE Int. Conf. on Computer-Aided Design (ICCAD)*, Nov. 6-10, 1994, Santa Clara, CA, pp. 137-144.
104. Q. Yu and C. Sechen, "Approximate Symbolic Analysis of Large Analog Integrated Circuits," *IEEE Int. Conf. on Computer-Aided Design (ICCAD)*, Nov. 6-10, 1994, Santa Clara, CA, pp. 664-671.
105. J. J. Hsu and C. Sechen, "Fully Symbolic Analysis of Large Analog Integrated Circuits," *IEEE Custom Integrated Circuits Conference (CICC)*, May 2-4, 1994, San Diego, CA, pp. 457-460.
106. Q. Yu and C. Sechen, "Generation of Color-Constrained Spanning Trees with Application in Symbolic Analysis," *Fourth Great Lakes Symposium on VLSI*, March 4-5, 1994, Notre Dame, IN, pp. 252-255.
107. K. Roy and C. Sechen, "A Timing Driven N-Way Chip and Multi-Chip Partitioner," *Proc. SRC TECHCON Conference*, September 28-30, 1993, Atlanta, GA.
108. W. Swartz and C. Sechen, "Precise Timing Driven Placement for Large Circuits," *Proc. SRC TECHCON Conference*, September 28-30, 1993, Atlanta, GA.
109. W. Swartz and C. Sechen, "A New Generalized Row-Based Global Router," *Proc. of the SRC TECHCON Conference*, September 28-30, 1993, Atlanta, GA.
110. T. Stanion and C. Sechen, "Maximum Projections of Don't Care Conditions in a Boolean Network," *IEEE Int. Conf. on Computer-Aided Design (ICCAD)*, Nov. 7-11, 1993, Santa Clara, CA, pp. 674-679.
111. W. Sun and C. Sechen, "Efficient and Effective Placement for Very Large Circuits," *IEEE Int. Conf. on Computer-Aided Design (ICCAD)*, Nov. 7-11, 1993, Santa Clara, CA, pp. 170-177.
112. K. Roy and C. Sechen, "A Timing Driven N-Way Chip and Multi-Chip Partitioner," *IEEE Int. Conf. on Computer-Aided Design (ICCAD)*, Nov. 7-11, 1993, Santa Clara, CA, 240- 247.
113. W. Swartz and C. Sechen, "A New Generalized Row-Based Global Router," *IEEE Int. Conf. on Computer-Aided Design (ICCAD)*, Nov. 7-11, 1993, Santa Clara, CA, pp. 491- 498.
114. K. Roy, B. Guan and C. Sechen, "A Sea-of-Gates Style FPGA Placement Algorithm," *Int. Conf. on VLSI Design*, January 5-8, 1994, Calcutta, India.
115. J. J. Hsu and C. Sechen, "Low-Frequency Symbolic Analysis of Large Analog Integrated Circuits," *IEEE Custom Integrated Circuits Conference (CICC)*, May 9-12, 1993, San Diego, CA, pp. 14.7.1-14.7.5.
116. T. Stanion and C. Sechen, "Boolean Division and Factorization Using Binary Decision Diagrams," *IEEE Asia-Pacific Conference on Circuits and Systems (APCCAS '92)*, Dec. 8-11, 1992, Sydney, Australia.
117. K. W. Lee and C. Sechen, "A Global Router for Sea-of-Gates Circuits," *Proc. European Design Automation Conference*, Amsterdam, The Netherlands, February 26-28, 1991, pp. 242-247.
118. D. Chen and C. Sechen, "Mickey: A Macro Cell Global Router", *Proc. European Design Automation Conference*, Amsterdam, The Netherlands, February 26-28, 1991, pp. 248- 252.
119. W. Swartz and C. Sechen, "New Algorithms for the Placement and Routing of Macro Cells," *Proc. Int. Conf. on Comp. Aided Design (ICCAD)*, Santa Clara, CA, November 11-15, 1990, pp. 336-339.
120. D. Chen and C. Sechen, "Mickey: A New Macro Cell Global Router," *Proc. SRC Techcon Conference*, San Jose, CA, October 16-18, 1990, pp. 241-244.
121. W. Swartz and C. Sechen, "New Algorithms for the Placement and Routing of Macro Cells," *Proc. SRC Techcon Conference*, San Jose, CA, October 16-18, 1990, pp. 233-236.
122. K. W. Lee and C. Sechen, "A New Global Router for Sea-of-Gates Circuits," *Proc. 9th Australian Microelectronics Conference*, Adelaide, S. Australia, July 2-4, 1990, pp. 233-244.
123. K. W. Lee and C. Sechen, "A New Global Router for Row-Based Layout," *Proc. of Int. Conf. on Computer-Aided Des. (ICCAD)*, Nov. 7-10, 1988, Santa Clara, CA, pp. 180-183.
124. C. Sechen and D. Chen, "An Improved Objective Function for Mincut Circuit Partitioning," *Proc. of Int. Conf. on Computer-Aided Des. (ICCAD)*, Nov. 7-10, 1988, Santa Clara, CA, pp. 502-505.
125. C. Sechen, "Chip-Planning, Placement, and Global Routing of Macro/Custom Cell Circuits Using Simulated Annealing," *Proc. of the 25th Design Automation Conference (DAC)*, June 1988, Anaheim, CA, pp. 73-80.
126. C. Sechen and K. W. Lee, "An Improved Simulated Annealing Algorithm for Row-Based Placement," *Proc. of Int. Conf. on Computer-Aided Des. (ICCAD)*, Nov. 9-13, 1987, Santa Clara, CA, pp. 478-481.

127. C. Sechen, "Average Interconnection Length Estimation for Random and Optimized Placements," *Proc. of Int. Conf. on Computer-Aided Des. (ICCAD)*, Nov. 9-13, 1987, Santa Clara, CA, pp. 190-193.
128. J. Burns, A. Casotto, M. Igusa, F. Marron, F. Romeo, A. Sangiovanni-Vincentelli, C. Sechen, H. Shin, G. Srinath, and H. Yaghutiel, "Mosaico: An Integrated Macro-Cell Layout System," in C. Sequin, editor, *Proc. of VLSI '87*, Vancouver, Canada, August, 1987.
129. C. Sechen and A. Sangiovanni-Vincentelli, "TimberWolf3.2: A New Standard Cell Placement and Global Routing Package," *Proc. 23rd Design Automation Conference (DAC)*, June 29-July 2, 1986, Las Vegas, NV, pp. 432-439.
130. D Braun, C. Sechen, and A. Sangiovanni-Vincentelli, "ThunderBird: A Complete Standard Cell Layout Package," *Proc. IEEE Custom Integrated Circuits Conf. (CICC)*, May 1986, Rochester, NY.
131. F. Romeo, C. Sechen, and A. Sangiovanni-Vincentelli, "Simulated Annealing Research at Berkeley," *Proc. Int. Conf. on Computer Design (ICCD)*, October 1984, Rye Brook, NY.
132. C. Sechen and A. Sangiovanni-Vincentelli, "The TimberWolf Placement and Routing Package," *Proc. IEEE Custom Integrated Circuits Conf. (CICC)*, May 1984, Rochester, NY.
133. M. Huberman, S. Senturia, and C. Sechen, "The Use of the Charge-Flow Transistor to Distinguish Surface and Bulk Components of Thin-Film Sheet Resistance," *IEEE Device Research Conference (DRC)*, Cornell University, June 1977.
134. S. Senturia and C. Sechen, "The Charge-Flow Transistor," *IEEE Int. Electron Devices Meeting (IEDM)*, Washington DC, Dec. 1976.
135. L. Liu and C. Sechen, "A Multi-Layer Chip-Level Global Router," *Proc. of the Fifth ACM/SIGDA Physical Design Workshop*, April 15-17, 1996, Reston, Virginia.
136. L. Liu and C. Sechen, "Multi-Layer Pin Assignment for Macro Cell Circuits," *Proc. of the Fifth ACM/SIGDA Physical Design Workshop*, April 15-17, 1996, Reston, Virginia.
137. G. Yee and C. Sechen, "Clock-Delayed Domino for Adder and Combinatorial Logic Design," *Proc. of the Fifth ACM/SIGDA Physical Design Workshop*, April 15-17, 1996, Reston, Virginia.
138. B. Guan and C. Sechen, "Efficient Standard Cell Generation When Diffusion Strapping is Required," *Proc. of the Fifth ACM/SIGDA Physical Design Workshop*, April 15-17, 1996, Reston, Virginia.
139. H. P. Tseng and C. Sechen, "A Gridless Multi-Layer Channel Router Based on a Combined Constraint Graph and Tile Expansion Approach," *Proc. of the Fifth ACM/SIGDA Physical Design Workshop*, April 15-17, 1996, Reston, Virginia.
140. J. J. Hsu and C. Sechen, "The Sifting Approach to Symbolic Analysis of Large Analog Integrated Circuits," *Proc. of the Third International Workshop on Symbolic Methods and Applications to Circuit Design (SMACD)*, Sevilla, Spain, October 6-7, 1994, pp. 211-230.
141. Q. Yu and C. Sechen, "Approximate Symbolic Analysis of Large Analog Integrated Circuits," *Proc. of the Third International Workshop on Symbolic Methods and Applications to Circuit Design (SMACD)*, Sevilla, Spain, October 6-7, 1994, pp. 241-259.
142. Q. Yu and C. Sechen, "Efficient Approximation of Symbolic Network Functions Using Matroid Intersection Algorithms," *Proc. of the Third International Workshop on Symbolic Methods and Applications to Circuit Design (SMACD)*, Sevilla, Spain, October 6-7, 1994, pp. 261-277.
143. T. Stanion and C. Sechen, "Maximum Projections of Don't Care Conditions in a Boolean Network," *Proc. of the International Workshop on Logic Synthesis*, Lake Tahoe, CA, May 23-26, 1993.
144. W. Sun, K. Roy, and C. Sechen, "Fast, High-Quality Placement for Large Circuits," *Proc. of the 4th ACM/SIGDA Physical Design Workshop*, Lake Arrowhead, CA, April 19-21, 1993.
145. K. Roy, D. Guan and C. Sechen, "FPGA MCM Partitioning and Placement," *Proc. of the 4th ACM/SIGDA Physical Design Workshop*, Lake Arrowhead, CA, April 19-21, 1993.
146. T. Stanion and C. Sechen, "Incremental and Simultaneous Computation of Satisfiability and Observability Don't Cares," *Proc. IEEE/ACM Winter VLSI Workshop*, Jan. 25-27, 1993, Asilomar, CA.
147. R. Weier and C. Sechen, "TimberWolfAR: An Arbitrary Design Rule Multi-Layer Area Router," *Int. Workshop on Layout Synthesis*, May 18-21, 1992, MCNC, Research Triangle Park, NC.
148. W. Swartz and C. Sechen, "A Generalized Row-Based Global Router for FPGAs," *Int. Workshop on Layout Synthesis*, May 18-21, 1992, MCNC, Research Triangle Park, NC.
149. K. Roy and C. Sechen, "A Timing Driven N-Way Chip Partitioner," *Int. Workshop on Layout Synthesis*, May 18-21, 1992, MCNC, Research Triangle Park, NC.
150. K. W. Lee and C. Sechen, "A Global Router for Sea-of-Gates Circuits," *Proc. Int. Workshop on Layout Synthesis*, Research Triangle Park, NC, May 8-11, 1990.

151. D. Chen and C. Sechen, "Mickey: A New Macro Cell Global Router", *Proc. Int. Workshop on Layout Synthesis*, Research Triangle Park, NC, May 8-11, 1990.
152. W. Swartz and C. Sechen, "New Algorithms for the Placement and Routing of Macro Cells," *Proc. Int. Workshop on Layout Synthesis*, Research Triangle Park, NC, May 8-11, 1990.
153. C. Sechen, "TimberWolfSG: A Sea-of-Gates Layout Package," *Proc. of SRC Workshop on Sea-of-Gates Design*, Sept. 22-23, 1988, Berkeley, CA.
154. C. Sechen, "A Global Router for the Sea-of-Gates Environment," *Proc. IFIP Workshop on Physical Design of Sea-of-Gates VLSI*, August 11-12, 1988, Lenggries, Germany.
155. J. Lam, J. M. Delosme, and C. Sechen, "A New Simulated Annealing Schedule for Row- Based Placement," *Proc. IEEE International Workshop on Placement and Routing*, May 10-13, 1988, Research Triangle Park, NC.
156. F. Romeo, C. Sechen, and A. Sangiovanni-Vincentelli, "Some Theoretical Results on Simulated Annealing and TimberWolf: A Placement and Routing Package for the Layout of Integrated Circuits," *Proc. Workshop on Statistical Physics in Engineering and Biology*, IBM T.J. Watson Research Center, Yorktown Heights, NY, April 1984.

Books Authored

Carl Sechen, **Placement and Global Routing of Integrated Circuits Using Simulated Annealing**, 278 pages, Kluwer Academic Publishers, Boston, MA, 1988.

Patents

C. Sechen, Y. Makris, G. Reddy, J. Tian, U.S. Provisional Patent Application No. 62/477,144, filed March 27, 2018, entitled "Field Programmable Transistor Arrays".

C. Sechen, L. McMurchie, G. Yee and T. Thorp, "Output Prediction Logic", #6,549,038 issued March 2003.

Research Funding

- NSF SHF: Medium: PATCH-UP: Programmable Array of Transistors for Cost-effective Hardware Updates and IP Protection", PI: Yiorgos Makris – Co-PIs: Carl Sechen, Benjamin Carrion Schaefer, William Swartz Jr., Sept. 1, 2019 – Aug. 31, 2023, $1.2 million. Status: Under Review.

- "OpenROAD: Foundations and Realizations of Open, Accessible Design", co-PIs Carl Sechen and William Swartz, Defense Advanced Research Products Agency (DARPA) IDEA Program, May 1, 2018 through April 30, 2022, $869,341.

- "BOOTSTRAP: Behavior ObfuscatiOn and Hardware Trojan Detection through Selective Post-Fabrication TRAnsistor-Level Programming", PI: Yiorgos Makris, co-PI: Carl Sechen, Ben Carrion Schaefer, Defense Advanced Research Products Agency (DARPA), September 1, 2017 through February 28, 2019, $543,234. Current status: (still) under review.

- CSR: "Small: Host-Assisted Software-Defined Solid-State Disk", PI: Carl Sechen, October 1, 2014 – September 30, 2019, $265,067.

- TerraSwarms Research Center, DARPA/SRC FCRP research center based at UC Berkeley, Jan. 15, 2013 – Jan. 14, 2014, co-PI: Carl Sechen portion: $100,000.

- "Optimal Gate Size and Vt Selector, and Innovative Approach to Path-Based Timing Analysis", PI: Carl Sechen, Texas Instruments, Jan. – Dec., 2012, $50,000. Current status: completed.

- "Optimal Gate Size and Vt Selection", Texas Instruments, Wireless Division, Jan. – Dec., 2011, $50,000, PI: Carl Sechen.

- "Power Minimization Research", Texas Instruments, ASIC Division, Jan. – Dec., 2011, $50,000, PI: Carl Sechen.

- "Power Minimization Research", Texas Instruments, ASIC Division, Jan. – Dec., 2010, $75,000, PI: Carl Sechen.

- "Low Power HD Video Decoder Research", Texas Instruments, Wireless Division, Jan. – Dec., 2010, $50,000, PI: Carl Sechen.

- "Power Minimization Research", Texas Instruments, ASIC Division, Jan. – Dec., 2009, $50,000, PI: Carl Sechen.

- "Low Power HD Video Decoder Research", Texas Instruments, Wireless Division, Jan. – Dec., 2009, $50,000, PI: Carl Sechen.

- "Power Optimization and Circuit Hardening", MARCO Focus Center Research Program (C2S2), $155,000, September 2005 – August 2008.

- "Ultra High Performance Digital Circuit Design and Synthesis", National Science Foundation (NSF), $80,000, (Jan. 2006 – Dec., 2008).

- "Low Power HD Video Decoder Research", Carl Sechen, Texas Instruments, Wireless Division, $80,000, Jan. – Dec., 2008.

- "Power Minimization Research", Carl Sechen, Texas Instruments, ASIC Division, $37,000, Jan. – Dec., 2008.

- "Power Minimization and Yield Optimization in Digital IC's", Carl Sechen, Semiconductor Research Corporation TxACE, $150,000, August 2007 – July 2010.

- "Analog-to-Digital Converter based on Time-to-Digital Conversion", Carl Sechen, Texas Instruments, $40,000, Sep. 2007 – Aug. 2008.

- "Low Power HD Video Decoder Research", Carl Sechen, Texas Instruments, Wireless Division, $45,000, Jan. – Dec., 2007.

- "Three-Transistor Memory Cell Research", Carl Sechen, Texas Instruments, $35,000, Jan. – Dec., 2007.

- "Power Minimization Research", Carl Sechen, Texas Instruments, ASIC Division, $37,000, Jan. – Dec., 2007.

- "Low Power HD Video Decoder Research", Carl Sechen, Texas Instruments, Wireless Division, $30,000, Sept. – Dec., 2006.

- "DRAM-Density SRAM", Carl Sechen, Honeywell, $105,000, January 2006 – April 2007.

- "RADHARD by Design," DARPA/Boeing, Phase I, $45,000 (co-PI, my portion), November 2004 – August 2005).

- "CLASS: Clockless Logic, Analysis, Synthesis, and Systems," DARPA/Boeing, Phase II/III, $709,000 (co-PI, my portion), (Sept. 2004 – August 2006).

- "MSP Phase 2: Digital IC Optimization Techniques," DARPA/Boeing, $285,000 (co-PI, my portion), (September 2004 – August 2006).

- "Power Optimization and Circuit Hardening," MARCO FCRP C2S2 Center, $100,000, PI (Oct. 04 – Sep. 05).

- "High Speed and Low Power Digital IC Design," NSF ITR, $350,000 (my portion) co-PI, (Sept. 00 – Aug. 05).

- "Reconfigurable RF, Analog, and Digital Array for Radiation-Hardened Communication Circuits," Air Force Research Lab, $110,000, PI (Sep 04 – Aug 05).

- "High Speed & Low Power Digital Circuit Synthesis and Design," MARCO FCRP C2S2 Center, $100,000, PI (Oct. 03 – Sep. 04), full indirect.

- "CLASS: Clockless Logic, Analysis, Synthesis, and Systems," DARPA/Boeing, Phase I, $135,000 (co-PI, my portion), (Oct. 2003 – May 2004).

- "Reconfigurable RF, Analog, and Digital Array for Radiation-Hardened Communication Circuits," Air Force Research Lab, $105,000, PI (Sep 03 – Aug 04), full indirect.

- "Low Power Digital IC Design," NSF Center for the Design of Analog and Digital ICs (CDADIC), $55,000, PI (Sep 03 – Aug 04), no indirect.

- "Reconfigurable RF, Analog and Digital Array for Radiation-Hardened Wireless Communication Circuits," DARPA/SPAWAR, $150,000, PI (April 03 – May 04), full indirect.

- "Development of Pulse-Based Logic," DARPA, $100,000, PI (March 03 – May 04), full indirect.

- "High Performance and Power Efficient Digital IC Optimization Techniques," MSP Phase 1: Boeing/DARPA, $400,000, PI (April 02 – August 03), full indirect.

- Unrestricted grant, Intel Corporation, $60,000, PI (May 03 – April 04).

- "High Performance Digital IC Design," MARCO/DARPA C2S2 Center, $170,000, PI (Jan. 01 – Aug. 03), full indirect.

- "Ultra High Performance Digital Circuit Design and Synthesis," National Science Foundation, $180,000, PI, (Sep 02 – Aug 05).

- "Reconfigurable RF, Analog, and Digital Array for Radiation-Hardened Communication Circuits," Air Force Research Lab, $95,000, PI (Sep 02 – Aug 03), full indirect.

- "PLL Design, Clock Generation and High-Speed/Low-Power DSP-block Design," NSF Center for the Design of Analog and Digital ICs (CDADIC), $55,000, PI (Sep 02 – Aug 03), no indirect.

- Unrestricted grant, Intel Corporation, $60,000, PI (Mar 02 – Feb 03), no indirect.

- "High Performance Digital IC Design," Semiconductor Research Corporation (SRC), June 1, 2000 – May 31, 2003, $525,000 (full indirect).

- "High Performance Digital IC Design," NSF Center for the Design of Analog and Digital ICs (CDADIC), $60,000 (no indirect), PI (Sep. 00 - Aug. 01).

- Unrestricted grant, Intel Corporation, $60,000, PI (Mar 01 – Feb 02).

- Unrestricted grant, Sun Microsystems, $30,000, PI (Feb 00 – Mar 01).

- Unrestricted grant, Compaq Computer Corp., $25,000, PI (Oct. 99 – Sep. 00).
- Unrestricted grant, Cadabra, Inc., $20,000, PI (Feb. 00 – Mar 01).
- "Ultra High Speed Digital Circuit Synthesis and Layout NSF Center for the Design of Analog and Digital ICs (CDADIC), $55,000 (no indirect), PI (Sep. 99-Aug. 00).
- "Ultra High Speed Digital Circuit Synthesis and Layout", National Science Foundation, $255,000 (full indirect), PI (Sep 99 – Aug 02).
- "Ultra High Speed Digital Circuit Synthesis and Layout", Semiconductor Research Corporation, $154,000 (full indirect), PI (Mar 99-Mar 00).
- "Ultra High Speed Digital Circuit Synthesis and Layout", , Semiconductor Research Corporation, $154,000 (full indirect), PI (Mar 99-Mar 00).
- "Synthesis of Ultra High Speed Random Logic Blocks", Semiconductor Research Corporation, $132,000, PI (Mar 98-Feb 99).
- "Ultra High Speed Digital Circuit Synthesis and Layout", NSF Center for the Design of Analog and Digital ICs (CDADIC), $65,000, PI (Sep. 98-Aug. 99).
- "Radiation-hard High Speed Digital Circuit Synthesis", Boeing, $35,000, PI (Sep. 98-Aug. 99).
- "Datapath Cell Layout", Compaq/Digital Equipment, $25,000, PI (Sep. 98 – Aug. 99).
- Unrestricted grant, Sun Microsystems, $8,000, PI (June 99 – May 00).
- Unrestricted grant, Compaq Computer Corp., $25,000, PI (Oct. 98 – Sep. 99).
- Unrestricted grant, Sun Microsystems, $30,000, PI (Jan. 99 – Dec. 99).
- Unrestricted grant, TimberWolf Systems, Inc., $20,000, PI (Apr. 98 – Mar. 99).
- Unrestricted grant, Sun Microsystems, Inc., $8,000, PI (June 98 – May 99).
- Various industrial gifts, $65,000, PI (Sep 97-Aug 98).
- "High Speed Digital Circuit Design", NSF Center for the Design of Analog and Digital ICs (CDADIC), $50,000, PI (Sep. 97-Aug. 98).
- "Automatic Layout Research", Semiconductor Research Corporation, $132,000, PI (Mar 97-Feb 98).
- Various industrial gifts, $112,000, PI (Sep 96-Aug 97).
- "High Speed Circuit Design and Mixed Signal Layout", NSF Center for the Design of Analog and Digital ICs (CDADIC), $50,000, PI (Sep. 96-Aug. 97).
- "Symbolic Analysis of Large Analog Circuits", National Science Foundation, $90,000, PI (Sep 94-Aug 97).
- "Automatic Layout Research", Semiconductor Research Corporation, $132,000, PI (Mar 96-Feb 97).

- "High Speed Circuit Design and Mixed Signal Layout", NSF Center for the Design of Analog and Digital ICs (CDADIC), $45,000, PI (Sep. 95-Aug. 96).

- Various industrial gifts, $117,000, PI (Sep 95-Aug 96).

- "Automatic Layout Research", Semiconductor Research Corporation, $132,000, PI (Mar 95-Feb 96).

- "High Speed Circuit Design and Mixed Signal Layout", NSF Center for the Design of Analog and Digital ICs (CDADIC), $45,000, PI (Sep. 94-Aug. 95).

- Various industrial gifts, $208,000, PI (Sep 94-Aug 95).

- "Automatic Layout Research", Semiconductor Research Corporation, $144,000, PI (Mar 94-Feb 95).

- "High Speed Circuit Design and Mixed Signal Layout", NSF Center for the Design of Analog and Digital ICs (CDADIC), $45,000, PI (Sep. 93-Aug. 94).

- Various industrial gifts, $343,000, PI (Sep 93-Aug 94).

- "Automatic Layout Research", Semiconductor Research Corporation, $140,000, PI (Mar 93-Feb 94).

- "High Speed Circuit Design and Mixed Signal Layout", NSF Center for the Design of Analog and Digital ICs (CDADIC), $45,000, PI (Sep. 92-Aug. 93).

- Various industrial gifts, $312,000, PI (Sep 92-Aug 93).

- "Automatic Layout Research", Semiconductor Research Corporation, $140,000, PI (Mar 92-Feb 93).

- Various industrial gifts, $306,000, PI (Sep 91-Aug 92).

- Yale University, Various Industrial Gifts, $1.65 million, PI (1986-1992).

Conference Service Activities

- Session Chairman for the Design Automation and Test in Europe Conference, 2012-2014.
- Session Chairman for the IEEE Design Automation Conference, 1987, 1990, 1991, 1997.
- Session Chairman for the International Symposium on Physical Design, 1997, 1998.
- Session Chairman for the IEEE Int. Conf. on Computer Design (ICCD), 1988, 89, 90, 91, 92.
- Session Chairman for the IEEE Int. Conf. on Comp. Aided Design (ICCAD), 1989, 1990, 1991, 1992, 1993, 1998.
- Elected Chairman for the Placement and Floorplanning sub-committee for the 2004 Design Automation Conference (DAC), 2004.
- Elected to the Placement and Floorplanning sub-committee for the 2003 Design Automation Conference (DAC), 2003.
- Elected Program Chair for the 2000 ACM/IEEE Int. Workshop on Timing Issues in the Specification and Synthesis of Digital Systems (TAU).
- Elected chairman of the placement and floorplanning committee of the 2001 Int. Conf. On Computer Aided Design (ICCAD).
- Elected chairman of the placement and floorplanning committee of the 2000 Int. Conf. On Computer Aided Design (ICCAD).
- Elected chairman of the placement and routing committee of the 1999 Int. Conf. On Computer Aided Design (ICCAD).

- Elected chairman of the placement and floorplanning committee of the 1998 Int. Conf. On Computer Aided Design (ICCAD).
- Elected Program Chairman of the ACM Physical Design Workshop, Lake Arrowhead, CA, April 1993.
- Elected Chairman of the 1992 Int. Workshop on Layout Synthesis, May, MCNC, Research Triangle Park, NC.
- Elected Chairman of the 1990 Int. Workshop on Layout Synthesis, May, MCNC, Research Triangle Park, NC.
- CAD Track Chairman for the IEEE Int. Conf. on Computer Design (ICCD), 1991.

Service as a Referee

- Technical Program Committee for the Design Automation Conference, 2014, 2015.
- Technical Program Committee for the Design Automation and Test in Europe Conference, 2012-2014.
- Technical Program Committee for the IEEE Int. Symposium on VLSI Design, 2010-2011.
- Technical Program Committee for the IEEE Int. Conf. on Comp. Aided Design (ICCAD), 1989-93, 98.
- Technical Program Committee for the IEEE Int. Conf. on Comp. Design (ICCD), 1987-97.
- Technical Program Committee for the European Design and Test Conference (EDTC), 1995, 96, 97, 98.
- Technical Program Committee for the Design Automation and Test in Europe (DATE) Conference, 1998, 99, 00, 01, 02, 03.
- Technical Program Committee for ACM International Physical Design Workshop, 1996.
- Technical Program committee for the International Symposium on Physical Design, 1997, 98.
- Technical Program Committee for the International Workshop on Layout Synthesis, 1988, 90, 92.
- IEEE/ACM Design Automation Conference, 1986-1998.
- National Science Foundation, 1986-present.
- IEEE Transactions on Computer Aided Design, 1986-present.
- IEEE Transactions on VLSI Systems, 1992-present.
- ACM Transactions on Design Automation of Electronic Systems, 1995-present.

University Service

UTD
Director of ECS Tech Support Services, June 2012 – February 2014
Information Technology Planning and Policy Committee, Sept. 1, 2016 – August 31, 2019

UW
Faculty Council on Research, 1994-1997
Faculty Senate, 2002-2004

Department Service

UTD
Chair of the Computer Resources Committee, ECE Dept., 2007-2019
Advisory Committee to TxACE (SRC Texas Analog Center of Excellence) 2008-2009
Chair of the Analog Search Sub-Committee, EE Dept., 2007-2008
Faculty Search Committee (computer engineering) 2010-11

UW
Faculty Search Committee, 2005-2006
Group Chair, VLSI and Digital Systems Group, 1997-2005
Faculty Search Committee, 98-99.
Graduate Studies and Research Committee, 1992-93, 96-97, 97-98.
Group Chair, Electronics, 94-95.
Graduate Admissions Committee, Electrical Engineering Dept., 1992-93.
Qualifying Examination Subcommittee, 1992-93.
Computer Resources Committee, 1992-93.

Outside University Service

Selected by the NSF to serve as a panelist for (invited) full proposals submitted to the CISE Expeditions in Computing (Expeditions) program, NSF 08-568, April 6-7, 2010.

Selected by the NSF to serve on the 2008 proposal review panel (April 14-15, 2008) to review a number of proposals that were submitted to the Computing Processes and Artifacts (CPA) cluster of the Computer, Information Sciences & Engineering (CISE) directorate of the National Science Foundation for Fiscal Year 2009 funding.

Selected by the NSF to serve on the 2007 proposal review panel (January 8-9, 2007) to review a number of proposals that were submitted to the Computing Processes and Artifacts (CPA) cluster of the Computer, Information Sciences & Engineering (CISE) directorate of the National Science Foundation for Fiscal Year 2008 funding.

Selected by the NSF to serve on the 2005 proposal review panel (October 20-21, 2005) to review a number of proposals that were submitted to the Computing Processes and Artifacts (CPA) cluster of the Computer, Information Sciences & Engineering (CISE) directorate of the National Science Foundation for Fiscal Year 2006 funding.

Selected by the NSF to serve on the 2004 proposal review panel (May 5-6, 2004) to review a number of proposals that have been submitted to the Computing Processes and Artifacts (CPA) cluster of the Computer, Information Sciences & Engineering (CISE) directorate of the National Science Foundation for Fiscal Year 2004 funding.

Selected by the NSF to serve on the 2002 CAREER proposal evaluation panel for the Design Automation Program (Division of Communications-Computer Research or C-CR) of the National Science Foundation.

Selected by the NSF to serve on the 1999 CAREER proposal evaluation panel for the Design Automation Program (Division of Communications-Computer Research or C-CR) of the National Science Foundation.

Co-Director of the NSF Center for the Design of Analog and Digital ICs (CDADIC), 1997-2005; this center provides about $450,000 annually to the University of Washington.

Professional Society Memberships

Eta Kappa Nu
Tau Beta Pi
Sigma Xi
IEEE Fellow
ACM