

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

TCL ELECTRONICS HOLDINGS LTD. and TCL INDUSTRIES HOLDINGS
CO., LTD.,
Petitioners,

v.

INTELLECTUAL VENTURES I LLC,
Patent Owner.

Case IPR2024-01245

U.S. PATENT NO. 7,089,443

**DECLARATION OF NADER BAGHERZADEH, PH.D. IN SUPPORT OF
PETITION FOR *INTER PARTES* REVIEW**

TABLE OF CONTENTS

I.	BACKGROUND AND QUALIFICATIONS	4
II.	LEGAL STANDARDS	12
III.	PRIORITY DATE AND LEVEL OF ORDINARY SKILL IN THE ART	15
IV.	OVERVIEW OF OPINIONS	17
V.	OVERVIEW OF THE ‘443 PATENT	18
A.	Background of the Technology.....	18
B.	Purported Invention	26
C.	Prosecution History.....	28
VI.	SUMMARY OF THE PRIOR ART.....	29
A.	Iyer (Ex. 1005).....	29
B.	Soerensen (Ex. 1006).....	30
C.	Hemani (Ex. 1007).....	32
D.	Moore (Ex. 1008).....	33
E.	Njolstad (Ex. 1009).....	33
F.	Liljeberg (Ex. 1010).....	34
VII.	GROUND.....	35
A.	Ground I: Claims 1, 3, 4, 5, 8, 10, and 12 Are Rendered Obvious By Iyer (Ex. 1005)	35
1.	Claim 1	35
2.	Claim 3: “The microprocessor of claim 1, wherein there are at least four of said domains.”	40
3.	Claim 4.....	40
4.	Claim 5: “The microprocessor of claim 1, further comprising a queue for communication between at least two of the domains.”	46
5.	Claim 8.....	48

Decl. ISO Petition for *Inter Partes* Review of U.S. Patent No. 7,089,443

6.	Claim 10: “The method of claim 8, wherein step (d) comprises providing a queue for communication between at least two of the domains.”	51
7.	Claim 12: “The microprocessor of claim 8, wherein there are at least four of said domains.”	51
B.	Ground II.A: Claims 1, 3, 4, 5, 8, 10, and 12 Are Rendered Obvious By Njolstad (Ex. 1009).....	51
1.	Claim 1	51
2.	Claim 3: “The microprocessor of claim 1, wherein there are at least four of said domains.”	57
3.	Claim 4.....	57
4.	Claim 5: “The microprocessor of claim 1, further comprising a queue for communication between at least two of the domains.”	60
5.	Claim 8.....	62
6.	Claim 10: “The method of claim 8, wherein step (d) comprises providing a queue for communication between at least two of the domains.”	64
7.	Claim 12: “The microprocessor of claim 8, wherein there are at least four of said domains.”	64
C.	Ground IIB: Claims 5 and 10 Are Rendered Obvious By Njolstad (Ex. 1009) in view of Liljeberg (Ex. 1010)	64
1.	Motivation to Combine Njolstad and Liljeberg	64
2.	Claim 5: “The microprocessor of claim 1, further comprising a queue for communication between at least two of the domains.”	67
3.	Claim 10: “The method of claim 8, wherein step (d) comprises providing a queue for communication between at least two of the domains.”	70
D.	Ground III.A: Claims 1, 3, 4, 5, 8, 10, and 12 Are Rendered Obvious By Soerensen	70
1.	Claim 1	70
2.	Claim 3: “The microprocessor of claim 1, wherein there are at least four of said domains.”	78

Decl. ISO Petition for *Inter Partes* Review of U.S. Patent No. 7,089,443

3.	Claim 4.....	78
4.	Claim 5: “The microprocessor of claim 1, further comprising a queue for communication between at least two of the domains.”	81
5.	Claim 8.....	83
6.	Claim 10: “The method of claim 8, wherein step (d) comprises providing a queue for communication between at least two of the domains.”	87
7.	Claim 12: “The microprocessor of claim 8, wherein there are at least four of said domains.”	87
E.	Ground III.B: Claims 1, 3, 4, 5, 8, 10, and 12 Are Rendered Obvious By Soerensen in view of Hemani	88
1.	Motivation to Combine Soerensen and Hemani	88
2.	Claim 1	89
3.	Claim 3: “The microprocessor of claim 1, wherein there are at least four of said domains.”	97
4.	Claim 4.....	98
5.	Claim 5: “The microprocessor of claim 1, further comprising a queue for communication between at least two of the domains.”	102
6.	Claim 8.....	105
7.	Claim 10: “The method of claim 8, wherein step (d) comprises providing a queue for communication between at least two of the domains.”	109
8.	Claim 12: “The microprocessor of claim 8, wherein there are at least four of said domains.”	109
F.	Ground III.C: Claims 5 and 10 Are Rendered Obvious By Soerensen (Ex. 1006) in view of Moore (Ex. 1008)	109
1.	Motivation to Combine Soerensen and Moore	109
2.	Claim 5: “The microprocessor of claim 1, further comprising a queue for communication between at least two of the domains.”	111
3.	Claim 10: “The method of claim 8, wherein step (d) comprises providing a queue for communication between at least two of the domains.”	114

VIII. OBJECTIVE INDICIA OF NON-OBVIOUSNESS.....	114
IX. CONCLUSION	116

TABLE OF EXHIBITS

Exhibit No.	Description
1001	U.S. Patent No. 7,089,443 (the “443 Patent”)
1002	Certified File History of the ‘443 Patent
1005	Anoop Iyer, and Diana Marculescu. “Power efficiency of voltage scaling in multiple clock, multiple voltage cores.” <i>Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design</i> (2002).
1006	U.S. Patent App. No. 20030071657 (“Soerensen”)
1007	A. Hemani, T. Meincke, S. Kumar, A. Postula, T. Olsson, P. Nilsson, J. Oberg, P. Ellervee, and D. Lundqvist. Lowering power consumption in clock by using globally asynchronous, locally synchronous design style. <i>Proc. ACM/IEEE Design Automation Conference</i> , 1999.
1008	Moore, Simon, George Taylor, Bob Mullins, and Peter Robinson. “Channel communication between independent clock domains.” <i>First ACiD-WG Workshop of the European Commission’s Fifth Framework Programme</i> . 2001.
1009	Tormod Njolstad, Odd Tjore, Kjetil Svarstad, Lars Lundheim, Tor O. Vedal, Jukka Typo, Tor Ramstad, Lars Wanhammar, E. J. Aar, and Helge Danielsen. “A socket interface for GALS using locally dynamic voltage scaling for rate-adaptive energy saving.” <i>Proceedings 14th Annual IEEE International ASIC/SOC Conference (IEEE Cat. No. 01TH8558)</i> . IEEE, 2001.
1010	Liljeberg, Pasi, Juha Plosila, and Jouni Isoaho. “Asynchronous interface for locally clocked modules in ULSI systems.” <i>ISCAS 2001. The 2001 IEEE International Symposium on Circuits and Systems (Cat. No. 01CH37196)</i> , IEEE, 2001.
1011	<i>Curriculum Vitae</i> of Dr. Nader Bagherzadeh
1012	D. M. Chapiro, <i>Globally Asynchronous Locally Synchronous Systems</i> . PhD thesis, Stanford University, 1984.
1013	Hsu, Chung-Hsing, Ulrich Kremer, and Michael Hsiao. “Compiler-directed dynamic voltage/frequency scheduling for energy reduction in microprocessors.” <i>Proceedings of the 2001 international symposium on low power electronics and design</i> , 2001.

Exhibit No.	Description
1014	I. E. Sutherland, “Micropipelines,” <i>Communications of the ACM</i> , June 1989.
1019	U.S. Patent No. 5,309,561
1022	U.S. Provisional App. No. 60/315,655

DECLARATION OF NADER BAGHERZADEH, PH.D.

1. I, Nader Bagherzadeh, make this declaration in connection with the proceedings identified above.

2. I have been retained on behalf of Petitioners TCL Electronics Holdings Ltd. and TCL Industries Holdings Co., Ltd. (“TCL”) to offer my expert opinions in connection with the above-captioned proceeding relating to U.S. Patent 7,089,443 (the “‘443 Patent”).

3. I am being compensated for my time in connection with this inter partes review (IPR) at my standard consulting rate of \$700 per hour plus reasonable expenses. No part of my compensation depends on the outcome of this proceeding.

4. I have been asked to provide my opinions as to whether claims 1, 3, 4, 5, 8, 10, and 12 (the “Challenged Claims”) of the ‘443 Patent are invalid from the perspective of a person having ordinary skill as of January 23, 2003.

5. In preparing this declaration, I have reviewed the ‘443 Patent (Ex. 1001), the file history of the ‘443 Patent (Ex. 1002), and all other materials referenced in this declaration. I have relied upon my education, training, knowledge, and experience in the art, and have considered the viewpoint of a person of ordinary skill in the art (POSITA) as of the dates listed above.

6. All emphases and annotations below are added, unless noted otherwise.

I. BACKGROUND AND QUALIFICATIONS

7. I summarize in this section my educational background, career history, and other relevant qualifications. My curriculum vitae is Exhibit 1011.

8. I am currently a professor in the Department of Electrical Engineering and Computer Science at the University of California, Irvine. Previously, I held the position of Professor and Chair of Electrical Engineering and Computer Science (ECE) at the University of California, Irvine, from 1998 to 2002. Prior to that I was an Associate Professor in the ECE Department at the University of California, Irvine from 1993 to 1997. Prior to that I was an Assistant Professor in the ECE Department at the University of California, Irvine from 1987 to 1992.

9. I earned a Ph.D. in Computer Engineering from the University of Texas at Austin in 1987. I earned a Master of Science degree in Electrical Engineering from the University of Texas at Austin in 1979, with a focus on communications theory and signal detection algorithms. I earned a Bachelor of Science degree, with honors, in Electrical Engineering from the University of Texas at Austin in 1977.

10. I have over 45 years of experience in the Electrical Engineering and Computer Science field. That experience includes holding the position of Member of Technical Staff, No. 5 ESS Hardware/Software T1 Line Interface Design at AT&T Bell Labs from 1980 to 1984. In this position I received the Outstanding Achievement award from AT&T Bell Labs in 1984. While at Bell Labs I developed

T1 line framing and error monitoring algorithms that were stored on electrically erasable programmable read-only memory (EEPROM)—a charge based non-volatile memory which is considered the predecessor to contemporary flash memory. Additionally, I designed the microprocessor-based board that included the EEPROM among many other components for directly interfacing the T1 line.

11. I have particular experience in development of System-on Chip (SoC) architecture with hundreds of cores (processors) based on the Network-on-Chip (NoC). In multi-core processors, NoC is responsible for data routing between different cores using basic networking principles. Incorporating a Globally Asynchronous Locally Synchronous (GALS) design can provide significant advantages for NoCs, improving modularity, reducing power consumption, and managing clock skew. As part of my work in this area, I developed a GALS based asynchronous NoC router and measured its power consumption and area overhead.

12. In 2002, I received the Best Paper award in Institute of Electrical and Electronics Engineers (IEEE) Transactions on Very Large Scale Integration (VLSI) Design related to task scheduling for multicore processors. I also received another Best Student Paper award co-authored with my doctoral students in the proceedings of Asia and South Pacific Design Automation Conference (ASPDAC) '02. In 2020, I received the Best Paper award in the proceedings of the IEEE International Conference on Edge Computing (EDGE).

13. In 2014, I received the Khwarizmi International Award and became an IEEE Fellow. In 2015, I received the award for Professor of the Year for the Department of Electrical Engineering and Computer Science at the University of California at Irvine.

14. Over the past 45+ years my research has focused on hardware, software, and VLSI design of low-power and high-performance multicore computer systems for wireless communication applications.

15. These applications have included mobile systems, computer graphics processing, 3D integrated circuits, memory structures (e.g., static random-access memory (SRAM), dynamic random-access memory (DRAM), flash memory, and register files), Network-on-Chip (NoC), and AI accelerators.

16. Since joining the University of California, Irvine in 1987, I have been teaching, researching, and consulting regarding almost all aspects of high-performance computer systems, including but not limited to VLSI design, computer architecture, parallel implementation of computationally critical applications such as wireless communication signal processing and error correcting schemes, DRAMs, SRAMs, low power design, and interfacing multicores.

17. Over the course of my career, I have supervised more than 45 doctoral students and 30 masters' students who have taken key positions in the electrical engineering and computer engineering sector. I have served as post-doctoral advisor

for more than 10 scholars that worked with my research group. The majority of my work with the doctoral students and visiting scholars have been on design and development of low power and high performance architectures. Our seminal work in this area has been highly cited by the research community and industry.

18. In 2000, I cofounded a company called Morpho Technologies, which was focused on the design and development of low power and high-performance digital signal processors (DSPs) for mobile applications. As part of my duties as the chief technologist for the company, I evaluated patents, technical reports, and presentations related to memory chip designs, DSPs, and parallel processing algorithms for mobile platforms.

19. For Morpho, I designed, developed, and tested the key components of a Coarse Grain Reconfigurable Architecture (CGRA), including the memory block responsible for exchanging data between the host processor and the multicore processor (SIMD array processor).

20. I also designed the main data storage of the microcontroller (a reduced instruction set computer, or “Tiny RISC”) that was used for storing temporary data between the Tiny RISC and the array processor, based on SRAM technology, which is a type of volatile memory. I was one of the primary contributors for the prototype board design of the Morpho chip, which utilized DRAMs, field-programmable gate arrays (FPGAs), flash memory and other prototype board components.

21. As part of my teaching and research experience, I have been responsible for developing circuits, electronic modules, and related architectures for a variety of multicore systems. For example, I designed the memory block responsible for managing out-of-order execution of data registers corresponding to different threads in the multithreaded superscalar DSP that was developed at University of California, Irvine. I also designed the original temporary data storage subcomponents for the VLIW integer processor (ViPER) and the original Tiny RISC processor. ViPER was considered the very first ever VLIW processor designed using Metal Oxide Semiconductor Implementation Service (MOSIS) 1.2-micron technology that included memory and logic circuits. I have also done extensive work on the design of efficient data correction schemes using reconfigurable computing based on the Morpho parallel array architecture.

22. In 2006, I analyzed the hardware implementation of low-density parity-check (LDPC) as a consultant for a local flash memory module design company. In this capacity, with the help of one of my doctoral students we modeled the hardware and evaluated the computation complexity of using an application-specific integrated circuit (ASIC) for implementing LDPC on a flash memory-based board.

23. Throughout my career, I have worked on many aspects of computer architecture optimizations, encompassing hardware, processor allocation and scheduling on 2D NoC array, and software techniques. Ideas based on NoC design,

modeling, and evaluation have been the hallmark of my research work for more than 15 years. I also extended my original ideas toward three-dimensional (3D) NoC and demonstrated some of the advantages of 3D integrated circuit (IC) design as well as potential complications that could arise from the thermal footprint of these architectures. NoC uses packet switching communication across a network of routers to facilitate core-to-core and core-to-memory communications. My main contributions have been on the critical aspects of NoC and IC design, including router design, through-silicon via (TSV) cross talk, power efficiency, task mapping and scheduling onto NoC, aging of ICs, and other related topics.

24. For NoC, I used Constraint Programming (CP) for solving the Voltage-Frequency Island (VFI) problem. In general, VFI is a superior design alternative in terms of thermal constraints, power consumption, and performance. Given a Communication Task Graph (CTG) and subsequent task assignments for cores, cores are allocated to the best possible places on the chip to minimize communication and power consumption. Moreover, an elegant CP model was proposed to solve the VFI problem by mapping and grouping cores at the same time while scheduling the computation tasks as a limited capacity resource allocation model using realistic benchmark datasets.

25. I proposed a new architecture for NoC called diagonally linked mesh NoC, or DMesh NoC. This novel architecture not only enhances network

transmission performance while maintaining a feasible implementation cost, but also provides a power-efficient solution for interconnection network scenarios. The proposed architecture uses an adaptive quasi-minimal routing algorithm so that it can improve average latency and saturation traffic load owing to its flexibility and adaptiveness. Based on these features, a congestion-aware routing algorithm was proposed to balance traffic load so as to alleviate congestion caused by high throughput network activities. Simulation results showed that saturation load is improved dramatically for various traffic patterns. Implementation results also suggested that employing diagonal links is a more area-efficient method for improving network performance than using large buffers. It was also shown that a congestion-aware router requires negligible cost overhead but provides better throughput.

26. I proposed a Quality of Service (QoS)-aware wirelessNoC architecture that not only enables quality-oriented network transmission with a feasible implementation overhead but also maintains a well-balanced workload inside hybrid networks to enhance system throughput. The proposed advanced routers employ congestion management which first grants high priority packets from more congested areas and then directs them to paths without contention. This adaptive load-balanced routing effectively relaxes congestion situations and prevents networks from excessive contention. Simulation results showed that average latency of high priority

is significantly improved. Flexible buffer management further improved transmission performance and resource utilization efficiency

27. I have worked on an efficient method for reaching the destination corresponding to 3D NoC nodes where power and latency have been considerably reduced as compared to what has been reported in the past. I devised an adaptive routing algorithm for partially connected 3D NoCs where an analytical formulation based on queuing theory for the first time was developed and its results were checked by simulation. I have worked on an energy and performance aware application mapping algorithm for inhomogeneous 3D NoCs that showed significant improvements. 2.5D NoC was the subject of another project that I did, for which I introduced a Controlled Load Balancing Mechanism (CLBM) for an efficient load balancing approach with more than 25% average latency and power reduction improvements with little accumulated overhead.

28. Through-silicon via (TSV) is one of the key enabling technologies for facilitating connections among cores of different layers of 3D ICs used for implementing NoC architectures with 'hundreds of cores. Some of my first work on designing routing algorithms for 3D NoC appeared in a journal (2016) when I introduced a new and robust method that is adaptive and can tolerate faults on the vertical TSV links. As part of this work, I proposed a novel algorithm to systematically generate hybrid 3D NoC topologies for a given application such that

the vertical connections are minimized while the NoC performance is not sacrificed. Experimental results using synthetic and real-world traffic patterns show that the proposed algorithm generates optimized architectures with lower energy consumption and a significant reduction in packet delay as compared to the existing solutions.

29. Currently, I am working on developing accelerators and optimization techniques for implementation of efficient machine learning platforms. As part of this work, I am taking standard matrix vector computations that are the most demanding aspect of the Convolutional Neural Network (CNN) algorithms and implementing them on highly parallel structures, taking advantage of inherent parallelism of neural network layers. The focus of this work has been on developing low power inference engines for edge based AI nodes.

30. I have been involved in numerous research projects, journal articles, and books. The specific research projects, journal articles, and books may be found in my Curriculum Vitae, which is attached to my Declaration as Exhibit 1011. Over the years, I have also received funding accumulating over eight million dollars. The specific funding and associated projects may be found in my Curriculum Vitae.

II. LEGAL STANDARDS

31. I am not a lawyer or legal expert, and I do not offer any opinions regarding applicable legal standards. Below, I set forth my understanding of the law

as it has been explained to me by counsel, and I have applied this understanding in forming my opinions in this declaration.

32. I understand from counsel that, to invalidate a patent claim as obvious in the context of an IPR proceeding, the petitioner bears the burden of showing by a preponderance of the evidence that the differences between the invention and the prior art are such that the subject matter as a whole would have been obvious to a person of ordinary skill in the art to which the subject matter pertains at the time the claimed invention was made. Factors relevant to the determination of obviousness include (1) the scope and content of the prior art, (2) the level of ordinary skill in the art at the time of the invention, (3) differences between the claimed invention and the prior art, and (4) “secondary considerations” or objective evidence of obviousness or nonobviousness.

33. I understand from counsel that obviousness can be established by showing that a person of ordinary skill in the art would have combined or modified the teachings of the prior art to produce the claimed invention and would have had a reasonable expectation of success in achieving the claimed subject matter. There is no requirement that the prior art provide an express teaching, suggestion, or motivation to combine elements of the prior art. Rather, the petitioner in an IPR must identify some apparent reason why a person of ordinary skill in the art would have combined the elements or concepts from the prior art in the same way as in the

claimed invention when considered as a whole. The motivation to combine may be implicit and may be found in the knowledge of a person of ordinary skill in the art, from the nature of the problem to be solved, market demand, or common sense. I am informed that some non-exclusive rationales that may support a conclusion of obviousness include:

- Combining prior art elements according to known methods to yield predictable results;
- Simple substitution of one known element for another to obtain predictable results;
- Use of known technique to improve similar devices (methods, or products) in the same way;
- Applying a known technique to a known device (method, or product) ready for improvement to yield predictable results;
- “Obvious to try” – choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success;
- Known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces if the variations are predictable to one of ordinary skill in the art; and
- Some teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention.

34. I understand from counsel that a determination regarding obviousness must be based on the knowledge possessed by a person of ordinary skill in the art at the time when the invention was made, and one should not consider in hindsight what is known today or what is learned from the teachings of the patent. Thus, it is inappropriate to use hindsight reconstruction of references to reach the claimed invention without any explanations as to how or why the references would be combined to produce the claimed invention.

35. I understand from counsel that an obviousness analysis cannot be based on nonanalogous art. The rationale behind this rule is that when art is directed to a different purpose than a claimed invention, an inventor would have less motivation or occasion to consider it. Art is analogous when it is from the same field of endeavor as the claimed invention, or if the art is not from the same field of endeavor, when it is reasonably pertinent to the particular problem faced by the inventor. To be “reasonably pertinent,” art must “logically commend itself” to an inventor’s attention in considering his problem.

III. PRIORITY DATE AND LEVEL OF ORDINARY SKILL IN THE ART

36. I have been asked by counsel to perform my analysis from the perspective of one of ordinary skill in the art as of January 23, 2003 (the claimed priority date on the face of the patent). None of my opinions would change if a somewhat earlier priority date were adopted.

37. In the event that Patent Owner provides any such evidence of an earlier priority date, I reserve the right to analyze such evidence and form opinions regarding whether the '443 Patent is indeed entitled to that earlier claimed priority date.

38. In my opinion, a person of ordinary skill in the art ("POSITA") at the time of the '443 Patent would have at least a Master's degree in electrical engineering, computer engineering, computer science, or a related field, with at least two or more years of experience in the field of processor design, with more education substituting for work experience and vice versa. I have applied this ordinary level of skill in my analysis, but none of my opinions would change if a somewhat higher or lower level of skill is adopted.

39. I have been qualified as at least a person of ordinary skill in the art since well before January 2003. I have also interacted extensively with and taught persons of ordinary skill in the art. My opinions set forth in this declaration are presented from the standpoint of one skilled in the art. Thus, where I specify in the analysis below that I am rendering my opinion, it should be understood that my opinion is from the perspective of a person having the level of ordinary skill in the art that I have described at the time of the invention.

IV. OVERVIEW OF OPINIONS

40. Below is a summary of my opinions as reflected in the more detailed analysis below:

Ground	Basis	Claims	Proposed Statutory Rejection
I	§ 103	1, 3, 4, 5, 8, 10, 12	Obvious over Iyer
II.A	§ 103	1, 3, 4, 5, 8, 10, 12	Obvious over Njolstad
II.B	§ 103	5, 10	Obvious over Njolstad in view of Liljeberg
III.A	§ 103	1, 3, 4, 5, 8, 10, 12	Obvious over Soerensen
III.B	§ 103	1, 3, 4, 5, 8, 10, 12	Obvious over Soerensen in view of Hemani
III.C	§ 103	5, 10	Obvious over Soerensen in view of Moore

41. I understand that in construing or interpreting a patent claim, the words of the claim are normally given their ordinary and customary meaning, which is the meaning that words would have had to a person of ordinary skill in the art in question at the time of the invention. The person of ordinary skill in the art is deemed to read the claim term in the context of not only the particular claim in which the disputed term appears, but also in the context of the entire patent, including the other claims and the specification, as well as the prosecution history of the patent. I understand that this is referred to as “intrinsic” evidence.

42. In the parallel litigation between Petitioners and Patent Owner, I understand there is a dispute regarding Claim 8(d), which reads “operating the microprocessor such that each domain operates synchronously, while the domains operate asynchronously relative to one another”, the term “synchronously” means “using the same clock with a fixed phase relationship”; and “asynchronously” means “using independent clocks with no known phase relationship.” I understand that Patent Owner disputes this construction. My analysis applies regardless of which construction is adopted.

43. I am not presently aware of any other terms that need to be construed to resolve the controversy presented by the prior art asserted herein.

V. OVERVIEW OF THE ‘443 PATENT

A. Background of the Technology

44. A microprocessor is an essential component in modern electronic devices, acting as the brain that performs computations and controls operations. Its integration, programmability, and ability to execute instructions efficiently have enabled the development of a myriad of applications ranging from personal computing and embedded systems to large data centers.

45. A microprocessor is an integrated circuit (IC) that serves as the central processing unit (CPU) of a computer or an embedded system. It performs the computation and control functions by executing a sequence of stored instructions

called the microprocessor program. Depending on the target application, the microprocessor may work on general purpose applications, such as those found in PCs, laptops, or data centers. A general purpose microprocessor is responsible for executing user applications ranging from word processing and Internet browsing, to streaming and graphics applications. On the other hand, dedicated microprocessors, such as those designed for digital signal processing (DSP), have specific applications related to processing digital signals and images in a very short period of time while meeting other embedded system specifications such as power consumption, cost, and reliability. A modern microprocessor consists of many processing cores on the same IC working in tandem for meeting the users' needs for computation and control.

46. Microprocessors are used in PCs and laptops for individual applications and tasks such as word processing, Internet, computer games, and streaming. In the case of embedded systems, microprocessors are an important component of smartphones, autonomous systems, medical devices, and consumer electronics. Microprocessors are essential for industrial control and manufacturing. Finally, in recent years they have played a key role in data centers and supercomputing facilities, for modeling computationally intensive applications in AI, chemistry, physics, weather forecasting and many other critically important applications.

47. Generally, microprocessors have the following major components:

- **Arithmetic Logic Unit (ALU):** Performs arithmetic and logic operations (e.g., addition, subtraction, AND, OR).
- **Registers:** Small, fast storage locations used to hold data and intermediate results during computation.
- **Control Unit:** Directs the operation of the processor, including fetching instructions from memory, decoding them, and managing their execution.
- **Cache Memory:** Small, fast memory located close to the CPU to speed up retrieval of frequently accessed data and instructions.
- **Bus Interface:** Manages communication with memory and peripheral devices through data, address, and control buses.
- **Clock source:** Responsible for providing a timing signal that synchronizes all operations within the microprocessor and ensures that instructions are executed in a precise and orderly manner.

48. Operationally, a microprocessor based on the concept of Reduced Instruction Set Architecture (RISC) does the following steps to execute user program:

- **Fetch:** The control unit fetches the next instruction to be executed from the memory.
- **Decode:** The fetched instruction is decoded to determine the operation to be performed and the operands involved.

- **Execute:** The decoded instruction is executed by the ALU or other relevant parts of the processor.
- **Memory:** The data memory is accessed for retrieving information or storing.
- **Store:** The result of the execution is stored in the appropriate register.

49. Globally Asynchronous and Locally Synchronous (GALS) is a design approach for complex ICs and multicore SoCs where different cores (processors) or digital computation blocks operate with their own local frequencies (synchronously) but communicate with each other asynchronously. This approach aims to combine the benefits of synchronous and asynchronous design techniques to improve performance, power efficiency, and scalability.

50. Locally synchronous means that individual domains operate synchronously with their own local frequency. That is, all their operations are coordinated and occur at regular intervals determined by the local frequency. Since the clock is distributed locally, it mitigates the design restriction of clock skew that occurs when a clock source has to travel long distances within an IC. A typical large IC can have multiple clock domains, each with their own local frequency.

51. Globally asynchronous means that the communication between different cores belonging to different clock domains are handled asynchronously.

There is no global clock that synchronizes all cores within an IC. Some of the advantages of GALS include:

- Scalability: GALS systems can scale more easily because they avoid the complexity and power consumption associated with distributing a single global clock across a large chip.
- Power Efficiency: Local clock signals can be optimized for the specific needs of each clock domain, potentially reducing overall power consumption. Cores belonging to the same clock domain can also enter low-power states independently when not in use. Scaling down frequency enables a reduction in the requisite voltage.
- Performance Optimization: Each module can run at its optimal speed without being constrained by a single clock signal, allowing for different clock domains to adjust operationally to meet application computational requirements. This can improve the overall performance of the system.
- Reduced Clock Skew: By using local clock signals, the problem of clock skew (timing differences between different parts of the chip) is minimized, enhancing the reliability of the system.
- Modularity: GALS systems are inherently modular, making it easier to design, test, and integrate different parts of the system independently.

52. Designing GALS requires management of communication complexity between different clock domains to guarantee proper data transfer integrity and reliability. Additional circuitry is needed for interfacing and buffering between asynchronous clock domains, which can introduce overhead in terms of area, power, and latency. First-In-First-Out (FIFO) buffers are commonly used to handle data transfers between asynchronous clock domains. They provide the means for safely transferring data without loss or corruption despite the absence of a global clock. Additionally, handshaking protocols, such as request/acknowledge signals, may be used to coordinate data transfers between asynchronous clock domains, ensuring that data is transmitted and received correctly.

53. In summary, GALS is a design methodology that combines the benefits of synchronous and asynchronous circuits by allowing different clock domains to operate with their own frequencies while communicating asynchronously. This approach enhances scalability, power efficiency, and performance but requires careful design to manage the complexity of asynchronous communication. GALS is particularly useful in large, complex SoCs where global clock distribution would be challenging and inefficient.

54. The total power consumption of a microprocessor is the sum of static power and dynamic power. Static power consumption, also known as leakage power, occurs even when the microprocessor is idle and not switching. It primarily consists

of sub-threshold leakage and gate oxide leakage. One of the techniques for reducing static power is power gating. Power gating involves turning off the power supply to idle parts of the microprocessor. This can be done using sleep transistors that disconnect the power supply to inactive circuits, significantly reducing leakage current.

55. Dynamic power consumption, also called switching power, occurs when transistors within the microprocessor switch states (from 0 to 1 or 1 to 0). Dynamic power consumption is $C * V^2 * F$, where C is a typical output capacitance of a transistor being switched per clock cycle, V is the voltage supply, and F is the clock frequency. In order to reduce power consumption for a given IC fabrication technology, lowering the voltage will have a quadratic effect in reducing the power consumption and lowering the frequency will reduce the power consumption linearly.

56. One of the main techniques for lowering the dynamic power consumption is to use Dynamic Voltage and Frequency Scaling (DVFS). DVFS adjusts the supply voltage and clock frequency based on the workload, reducing power consumption when full performance is not needed. By lowering voltage, the power consumption is dropped quadratically, but the frequency has to be lowered proportionally as well for the digital circuits to work properly.

57. The relationship between voltage and frequency can be approximated by considering the delay of a CMOS inverter, which is a basic building block of digital circuits. The delay of a CMOS inverter for transition from high to low and low to high can be expressed as:

$$t_{PHL} = \frac{C_{load} V_{DD}}{k_n (V_{DD} - V_{Tn})^2}$$
$$t_{PLH} = \frac{C_{load} V_{DD}}{k_p (V_{DD} - |V_{TP}|)^2}$$

58. Vdd is supply voltage, Vth is the transistor gate threshold voltage, C is the inverter output load capacitance, and K is related to the dimensions of the transistor gate. Therefore, if the voltage is reduced, then the delay of the gate increases, requiring the frequency to be lowered, and the operation is reversed when voltage is increased, meaning the frequency can be increased, since the delay is lowered.

59. This is the basic CMOS circuit equation that describes the relationship between voltage and frequency and how it can be incorporated as part of the DVFS for managing the overall power of a multicore microprocessor. For correct operation of the CMOS circuit, a design can lower the frequency and keep the voltage the same, although this would defeat the power-saving goals of domain-specific frequency reduction. Notably, the reverse is not true. Meaning, it is not possible to

increase the frequency and keep the voltage the same. The voltage has to be increased according to the delay equations shown above.

60. In 2003, microprocessors incorporated the following key features that they are still being used in modern processor: high clock speeds, increased cache sizes, 64-bit computing, branch prediction, and superscalar functionality.

B. Purported Invention

61. In my opinion, the ‘443 Patent is directed to a “straightforward extension” of the prior art—a microprocessor design utilizing multiple clock domain (MCD) microarchitecture and a globally asynchronous, locally synchronous (GALS) clocking style. Ex. 1001 at 4:51-54 (“In the present application, we describe an initial implementation of an MCD microprocessor that is a straightforward extension of a singly-clocked synchronous dynamic superscalar design.”). Each functional block within this microprocessor operates with its own clock frequency, and queue structures manage communication between domains.

62. The ‘443 Patent explains that the concept of using separate clock domains to address clock distribution and wire delay issues was well understood in the field. Ex. 1001 at 1:31-37 (“The result of these frequency and dimensional trends is that microprocessor clock speeds have become increasingly limited by wire delays, so much so that some of the more recent microprocessors, e.g., the Pentium

IV [14], have pipeline stages solely dedicated to moving signals across the chip.”). And while ‘443 Patent doesn’t explain the history of GALS systems, the concept of GALS was understood to date back to Chapiro’s 1984 PhD thesis. *See, e.g.*, Ex. 1005 at §1.1 (“GALS systems have been studied in detail by Chapiro in his 1984 thesis”); Ex. 1012. At the time of the ‘443 Patent’s prosecution, this concept had already been implemented in mass production. Ex. 1001 at 1:55-58 (“For example, several multiprocessor systems run the memory bus off of a different clock than the processor core in order to allow a single system to accommodate processors of different frequencies.”). Regarding queue structures specifically, the ‘443 Patent notes that it uses industry standard queues to facilitate interdomain communication. Ex. 1001 at 5:2-3.

63. The ‘443 Patent also proposes the use of dynamic voltage scaling in each domain to lower energy consumption. The ‘443 Patent’s discussion of the prior art discloses industry standard techniques for this concept as well. *See, e.g.*, Ex. 1001 at ref. [18]; Ex. 1013 at §1.2 (“In the context of dynamic voltage scaled microprocessors, voltage scheduling is a problem that assigns appropriate clock speeds to a set of tasks, and adjusts the voltage accordingly such that no task misses its predefined deadline while the total energy consumed is minimized.”). To the extent that the controlling algorithm managing performance and energy consumption is the claimed invention, the ‘443 Patent’s references also disclose this

topic. *See id.* (“Researchers have proposed many ways of determining ‘appropriate’ clock speeds through on-line and off-line algorithms [].”). The ‘443 Patent models several assumption-based outputs relating to performance overhead and energy-delay product, but I do not see these findings in the ‘443 Patent’s Claims. *See generally* Ex. 1001.

64. In my opinion, the ‘443 Patent’s claimed invention is a combination of known techniques regarding clock domain separation and dynamic voltage and frequency scaling, applied to the context of microprocessor design. The incremental nature of these improvements and the reliance on well-established principles suggest that the ‘443 Patent does not present an advancement over the prior art. A person of ordinary skill in the art would recognize that the purported benefits of the MCD microarchitecture were disclosed in existing methods, and thus question the claimed invention of the ‘443 Patent.

C. Prosecution History

65. I have reviewed the prosecution history of the ‘443 Patent. *See* Ex. 1002. To the extent my opinion is impacted by that history, I have addressed it below in my detailed analysis of each Ground. Further, in my opinion, Overhouse (Ex. 1019) discloses a synchronous processor unit, which can be considered a microprocessor because it integrates key functions of a central processing unit, such as instruction execution, memory management, and input/output operations.

VI. SUMMARY OF THE PRIOR ART

A. Iyer (Ex. 1005)

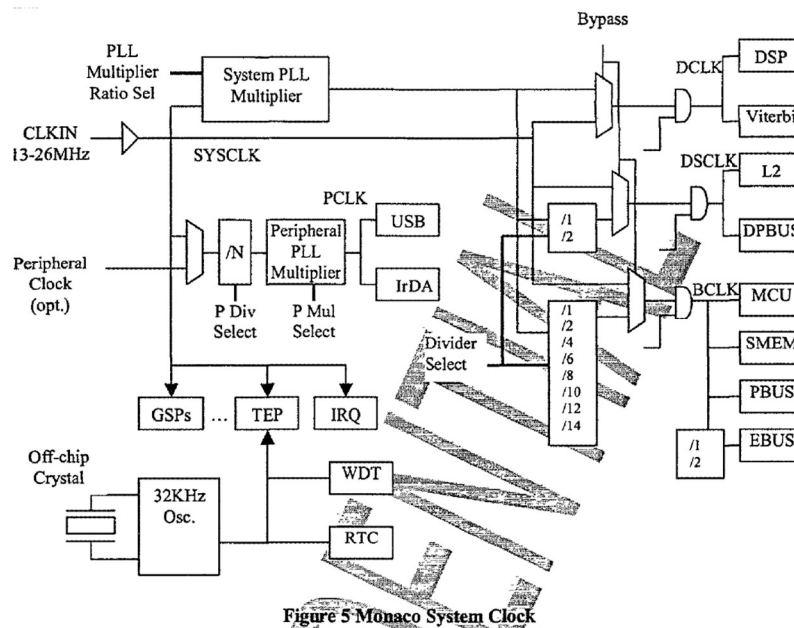
66. Iyer is an Institute of Electrical and Electronics Engineers (IEEE) conference paper titled “Power Efficiency of Voltage Scaling in Multiple Clock, Multiple Voltage Cores.” *See* Ex. 1005 at Cover. Iyer discusses how “[b]y exploiting the flexibility of independent dynamic voltage scaling the various clock domains, the power efficiency of GALS designs can be improved.” *Id.* at Abstract. Iyer outlines the design and modeling of a globally asynchronous locally synchronous (GALS) microprocessor where each domain is clocked independently.

67. Iyer is analogous prior art, including because (a) it is in the same field as the ‘443 Patent, and (b) is directed to the same problem as the ‘443 Patent in improving communication and power efficiency in a microprocessor. *See id.* at Abstract (“Due to increasing clock speeds, increasing design sizes and shrinking technologies, it is becoming more and more challenging to distribute a single global clock throughout a chip. In this paper we study the effect of using a Globally Asynchronous Locally Synchronous (GALS) organization for a superscalar, out-of-order processor, both in terms of power and performance. To this end, we propose a novel modeling and simulation environment for multiple clock cores with static or dynamically variable voltages for each synchronous block. Using this design exploration environment we were able to assess the power/performance tradeoffs

available for Multiple Clock, Single Voltage (MCSV), as well as Multiple Clock, Dynamic Voltage (MCDV) cores.”).

B. Soerensen (Ex. 1006)

68. Soerensen is a United States Patent Application Publication titled “Dynamic Voltage Control Method and Apparatus.” *See* Ex. 1006 at Cover. Soerensen discusses using a dynamic power controller which identifies a clock frequency requirement of a processor and determines a voltage requirement to support the clock frequency requirement in each independent domain. The control logic transitions the processor to a power state defined by the clock frequency requirement and the voltage requirement. I understand that Soerensen (Ex. 1006) claims priority to provisional App No. 60/315,655 (Ex. 1022). While I do not opine on the legal bases for priority, it is my opinion that (i) at least one of Soerensen’s claims is supported by the provisional application and (ii) Soerensen’s disclosures relied on as prior art in this proceeding find support in the provisional application. For example, Soerensen’s Claim 1 claims: “In a system including at least one processor having at least one clock signal capable of operating at a plurality of frequencies, a method of dynamic power control comprising acts of: determining a clock frequency requirement of the at least one processor; determining a voltage requirement to support the clock frequency requirement; operating the at least one clock signal according to the clock frequency requirement; and providing a voltage



Ex. 1022 at p. 169; *see also id.* at p. 170 (“frequencies can be changed dynamically . . . in conjunction with dynamically varying the internal logic voltage levels, to optimize power consumption”). For the same reasons, the provisional application supports the disclosures in Soerensen relied upon as prior art.

69. Soerensen is analogous prior art because (a) it is in the same field as the ‘443 Patent, and (b) it addresses the same problem as the ’443 Patent in improving communication and power efficiency in a microprocessor. *See id.* at Abstract (“A dynamic power controller is provided that identifies a clock frequency requirement of a processor and determines a voltage requirement to support the clock frequency requirement. The dynamic power controller transitions the processor to a power state defined by the clock frequency requirement and the voltage requirement. In particular, a voltage level indicated by the voltage requirement is supplied to the processor and the frequency distribution indicated by the frequency requirement is provided to the clocks signals of the processor.”).

C. Hemani (Ex. 1007)

70. Hemani is a paper titled “Lowering Power Consumption in Clock by Using Globally Asynchronous Locally Design Synchronous Style.” *See* Ex. 1007, Cover. Hemani discusses partitioning a microprocessor into a plurality of individually clocked domains which are internally synchronous but asynchronous to

one another. Hemani is focused on reducing power consumption and also disclosed means for inter-domain communication.

71. Hemani is analogous prior art because (a) it is in the same field as the '443 Patent, and (b) it addresses the same problem as the '443 Patent in improving communication and power efficiency in a microprocessor. *See id.*

D. Moore (Ex. 1008)

72. Moore is a paper titled “Channel Communication Between Independent Clock Domains.” Ex. 1008 at Cover. Moore discusses reliable low-latency channel communication between clock domains using an asynchronous interconnect and queues for interdomain communications. Like Iyer, these techniques are implemented in a GALS (Globally Asynchronous Locally Synchronous) system, where each clock domain operates independently.

73. Moore is analogous prior art because (a) it is in the same field as the '443 Patent, and (b) it addresses the same problem as the '443 Patent in improving communication and power efficiency in a microprocessor. *See id.*

E. Njolstad (Ex. 1009)

74. Njolstad is a paper titled “A Socket Interface for GALS Using Locally Dynamic Voltage Scaling for Rate-Adaptive Energy Saving.” *See* Ex. 1009 at Cover. Njolstad discusses a low-power amendment to the GALS paradigm using locally dynamic voltage scaling (LDVS) adapted to the actual processing rate requirements

for each module. What Njolstad terms GALS-LDVS is a microprocessor-focused design with a plurality of independent clock domains which communicate asynchronously. *See id.* Each domain's frequency and voltage is scalable in order to facilitate maximal power efficiency. *See id.*

75. Njolstad is analogous prior art because (a) it is in the same field as the '443 Patent, and (b) it addresses the same problem as the '443 Patent in improving communication and power efficiency in a microprocessor. *See id.*

F. Liljeberg (Ex. 1010)

76. Liljeberg is a paper titled "Asynchronous Interface for Locally Clocked Modules in ULSI Systems." *See* Ex. 1010 at Cover. Liljeberg discusses the construction of globally asynchronous locally synchronous (GALS) systems using an architecture that employs interface units to manage communication between locally clocked modules in an ultra large scale integration (ULSI)—i.e., microprocessor—system. *See generally* Ex. 1010. Liljeberg emphasizes the use of dual-port RAM as a FIFO queue for data transfer between asynchronous and synchronous domains, allowing each module to use an independent local clock. *Id.*

77. Liljeberg is analogous prior art because (a) it is in the same field as the '443 Patent, and (b) it addresses the same problem as the '443 Patent in improving communication and power efficiency in a microprocessor. *See id.*

VII. GROUNDS

A. Ground I: Claims 1, 3, 4, 5, 8, 10, and 12 Are Rendered Obvious By Iyer (Ex. 1005)

1. Claim 1

78. In my opinion, Iyer renders obvious Claim 1.

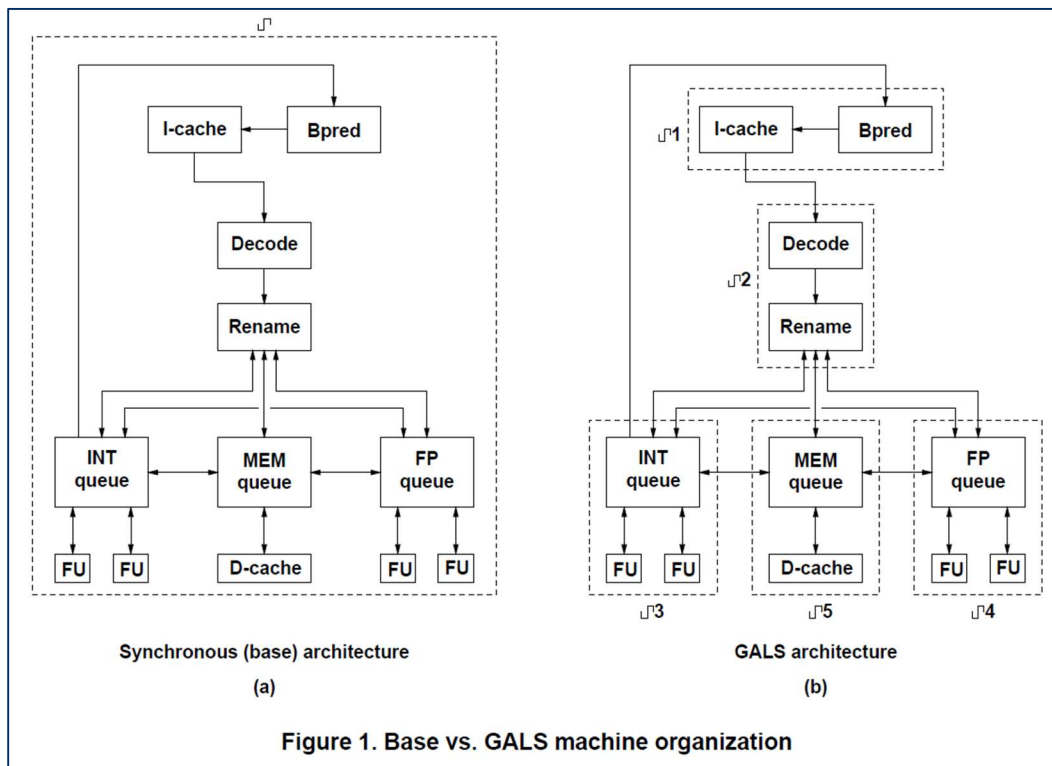
a. Limitation 1[pre]: “A multiple clock domain microprocessor comprising:”

79. In my opinion, to the extent the preamble is limiting, Iyer teaches this limitation. For example, Iyer discloses the “design [of] a microprocessor in a GALS style with multiple clock domains.” Ex. 1005 at §1. (“If we design a microprocessor in a GALS style with multiple clock domains (also called Multiple Clock, Single Voltage, or MCSV core), how much performance overhead will it incur over a fully synchronous machine?”).

b. Limitation 1[a]: “a plurality of domains;”

80. In my opinion, Iyer teaches this limitation. Iyer teaches a globally asynchronous, locally synchronous (GALS) design. Ex. 1005 at §1 (“Globally Asynchronous Locally Synchronous systems (which we refer to as GALS systems in this paper) are an intermediate style of design between these two. GALS systems contain several independent synchronous blocks which operate with their own local clocks and communicate asynchronously with each other.”). A GALS design entails “the use of several distinct local clocks (or clock domains).” *Id.* (“The main feature of these systems is the absence of a global timing reference and the use of several

distinct local clocks (or clock domains), possibly running at different frequencies.”). Iyer’s design discloses “five clock domains”—in other words, a plurality of domains. Ex. 1005 at §4.2 (“To simulate the GALS machine, we made use of the event-driven simulation engine described above. We have set up five clock domains in our simulator and in the first set of experiments, had all the clocks running at the same speed. The starting phase of each clock was set to a random value at runtime.”). This plurality of domains can be seen, *e.g.*, each denoted with dotted lines on the right side of Iyer’s Figure 1:



Ex. 1005 at 4, Fig. 1.

c. Limitation 1[b]

- i. **“for each of the plurality of domains, a clock for separately generating a clock signal at a frequency for that domain ...”**

81. In my opinion, Iyer teaches this limitation. For instance, Iyer teaches “several independent synchronous blocks which operate with their own local clocks.” Ex. 1005 at §1 (“GALS systems contain several independent synchronous blocks which operate with their own local clocks and communicate asynchronously with each other.”); *see also id.* at §3.2 (“In the GALS model, various regions are clocked using different clock signals independent of each other.”); *id.* at §3 (“Local clock generation: Each clock domain in a GALS system needs its own local clock generator; ring oscillators have been proposed as a viable clock generation scheme [4, 7]. We assume that we can use ring oscillators in each synchronous block in the GALS processor.”); *id.* §1 (“The main feature of these systems is the absence of a global timing reference and the use of several distinct local clocks (or clock domains), possibly running at different frequencies.”). Thus, for each of Iyer’s plurality of domains—alternately referred to as synchronous blocks—the clock is separately generated. The primary reason for partitioning the original system into distinct clock domains with their own independent different frequencies is to avoid all the design inefficiencies associated with distributing a common clock across the IC. Moreover, the notion of GALS allows for more fine-grained control of the power

consumption by adjusting each clock domain to meet the computation and power requirements of the target application.

- ii. **“... the frequency being dynamically changeable independently of the frequencies of the clock signals generated for others of the plurality of domains; and”**

82. In my opinion, Iyer teaches this limitation. As discussed with respect to element 1[b](i), the frequencies in each of Iyer’s domains are independent of the other domains. Iyer further expressly discloses that each domain is dynamically changeable. For instance, Iyer teaches that “the speed and voltage of each synchronous block are adapted dynamically.” Ex. 1005 at §3.4 (“We propose a multiple-clock dynamic-voltage core processor where the speed and voltage of each synchronous block are adapted dynamically based on the application profile.”). Iyer teaches both slowing down and increasing each domain’s frequency independently. *Compare* Ex. 1005 at §3.4 (“In addition, if we have a balanced pipeline design to start with, we can slow down synchronous blocks that are off the critical path of the design while keeping the others running at full speed.”) *with id.* (“If the occupancy rises above a certain threshold for these N consecutive cycles, we switch the clock back to full speed.”). This makes sense, because generally, the purpose of altering the frequency and voltage in each domain is to maximize performance while reducing energy. In some cases, that will require speeding up one domain, while slowing another down. By having different clock domains that can be controlled

independently, it is possible to adjust and ultimately tune the resource allocation mechanism to meet the specified power performance envelope.

- d. Limitation 1[c]: “for each of the plurality of domains, a voltage input for receiving a voltage which is dynamically changeable independently of the voltages applied to said others of the plurality of domains.”**

83. In my opinion, Iyer teaches this limitation. Iyer states that “[d]ynamic voltage scaling (DVS) has been widely studied and implemented commercially.” Ex. 1005 at §1.1. Iyer teaches that each “synchronous block” contains “dynamically variable voltages.” Ex. 1005 at §1 (“To this end, we propose a novel modeling and simulation environment for multiple clock cores with static or dynamically variable voltages for each synchronous block.”). In Iyer’s section dedicated to “Dynamic Clock and Voltage Management,” Iyer further discloses that “slower clock domains” can “operate at a smaller supply voltage.” This means that voltage is dynamically changeable on an independent—that is, domain by domain—basis. Ex. 1005 at §3.4 (“The slower clock domains could also operate at a smaller supply voltage, thus producing additional power savings.... Our method for dynamic clock management involves changing the operating frequencies (and supply voltages) of the integer, memory and floating-point clock domains in the GALS processor (indicated as clock domains 3, 4 and 5 in Figure 1).”). Dynamically changing both frequency and voltage for each domain independently is a key component of Iyer’s teachings. *See, e.g.,* Ex. 1005 at §3.4 (“Since usage profiles and communication patterns among

pipeline stages vary within and across applications, such MCDV cores should be able to dynamically adapt the speed and corresponding voltage for each synchronous block such that overall performance is kept within certain limits.”).

2. Claim 3: “The microprocessor of claim 1, wherein there are at least four of said domains.”

84. In my opinion, Iyer teaches this limitation. Iyer’s representative example of its system teaches using “five clock domains in the GALS design.” Ex. 1005 at §3.2 (“To compare GALS with synchronous microarchitecture organization, we have chosen to have five clock domains in the GALS design.”). Iyer also notes, for example, that the 21264 Alpha processor has six clock domains. *Id.* (“The 21264 has the following major clocks [21]: (1) instruction fetch and branch predict (2) bus interface unit (3) integer issue and execution units (4) floating point issue and execution units (5) load/store unit (6) pad ring. Guided by this, we have chosen synchronous blocks for our GALS design, which we describe below.”) Ultimately, as a POSITA would understand and as Iyer expresses, the number of domains is a design choice, at the discretion of the designer. Thus, Iyer discloses this claim and, in any event, does not add any non-obvious elements.

3. Claim 4

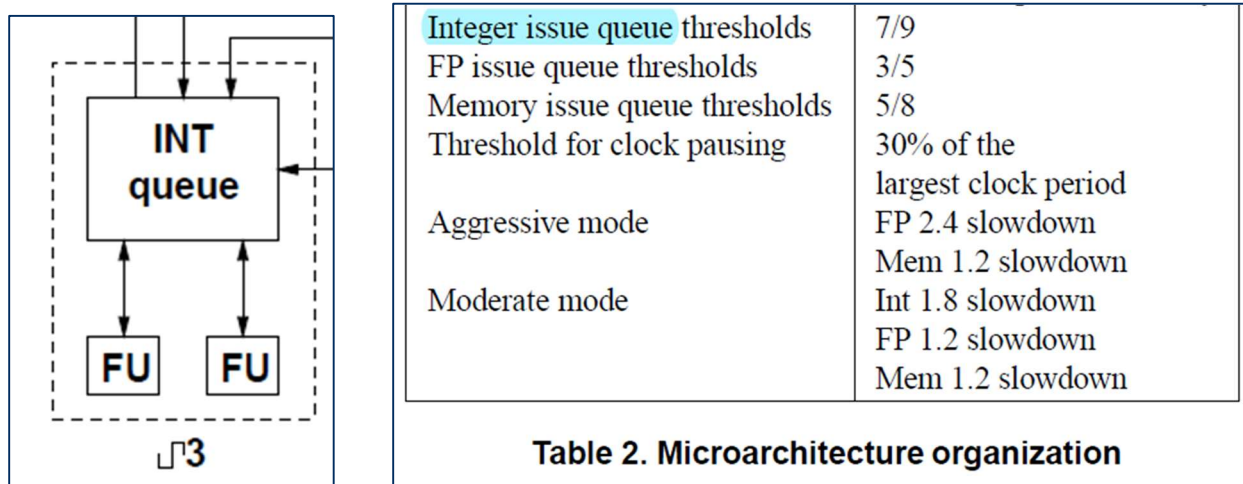
- a. Limitation 4[a]: “The microprocessor of claim 1, wherein the microprocessor is programmed to determine a slack in processing in one of the domains and . . .”**

85. In my opinion, Iyer teaches this limitation. Iyer's core invention is directed to the this concept. Iyer discusses "an adaptive approach for GALS processors which is able to tune the operating voltage of different clock domains, according to the usage of that clock domain." Ex. 1005 at §2.4. When Iyer discusses tuning the operating voltage according to the usage, in other words, Iyer is principally focused on reducing underutilization, or "slack."

86. In a GALS system, slack may be determined by monitoring the occupancy of each issue queue, where lower occupancy indicates the presence of slack. Occupancy simply means the amount of entries made in the queue structure. So, if one domain is overworking (pulling from the queue when it is below a threshold value), or underworking (allowing the queue to fill above a threshold value), one could monitor that behavior and adjust according. Iyer teaches determining slack in this way, disclosing that "we monitor the occupancy of each issue queue and we reduce the frequency to a lower value only when the queue occupancy drops below a low threshold for N consecutive cycles." Ex. 1005 at §3.4.

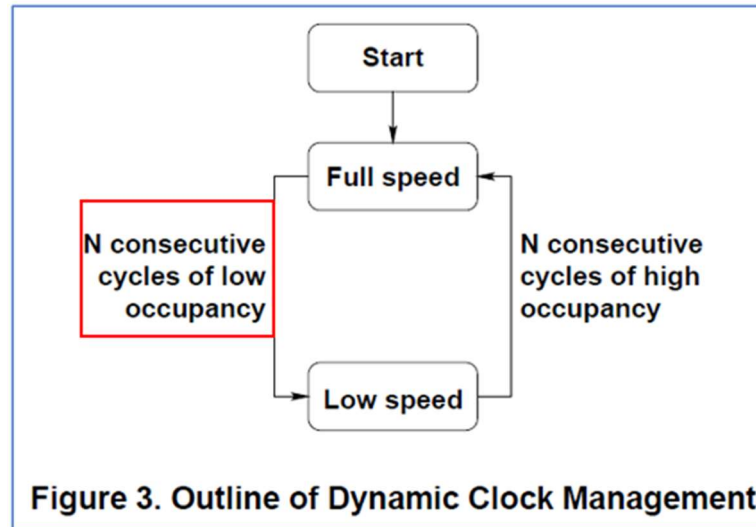
87. It should be noted that Claim 4 only claims the determination of slack in "one of the domains". Ex. 1001 at 17:27-31 ("The microprocessor of claim 1, wherein the microprocessor is programmed to determine a slack in processing in one of the domains and to reduce the clock frequency and the voltage in said one of the

domains to reduce the slack.”). Considering, for example, one domain in Iyer—the Iyer reference teaches an integer issue queue as part of Domain 3:



Ex. 1005 at Fig. 1 (cropped), Table 2 (cropped and annotated) (“Table 2 shows some details of the microarchitecture. The low and high occupancy thresholds for the three queues were chosen close to half of their maximum size, and less in the case of FP queue where a higher threshold may trigger a larger performance loss.”).

88. Iyer then teaches monitoring occupancy in each issue queue—including this domain, to determine slack:



Ex. 1005, Fig. 3 (annotated), §3.4. (“Figure 3 shows the outline of our method. The processor starts up with all clocks running at full speed. To minimize the performance impact of clock speed reduction, we monitor the occupancy of each issue queue and we reduce the frequency to a lower value only when the queue occupancy drops below a low threshold for N consecutive cycles. Since the execution characteristics may be different even within different parts of the same application, we continue monitoring the issue queues for a number of N consecutive cycles. If the occupancy rises above a certain threshold for these N consecutive cycles, we switch the clock back to full speed.”)

89. Monitoring slack is a threshold prerequisite to the operation described in Figure 3. Specifically, the system must monitor and first register a low occupancy before it would reduce the domain’s clock frequency to a lower speed. (“To minimize the performance impact of clock speed reduction, we monitor the

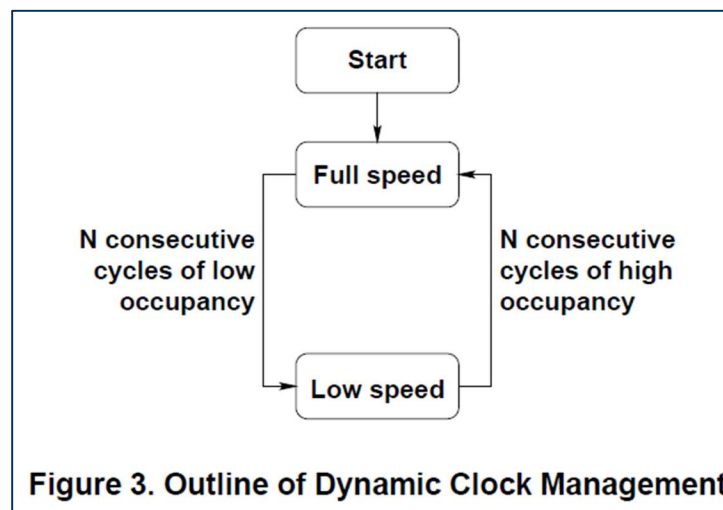
occupancy of each issue queue and we reduce the frequency to a lower value only when the queue occupancy drops below a low threshold for N consecutive cycles.”). As shown above, if N consecutive cycles of low occupancy occur (because the domain is pulling entries too fast), then the system lowers the domain’s speed, in order to increase energy-efficiency. *Id.*, §3.4. (“Since the execution characteristics may be different even within different parts of the same application, we continue monitoring the issue queues for a number of N consecutive cycles.”). Thus, due to this monitoring, Iyer discloses a microprocessor programmed to determine slack in one of the domains.

b. Limitation 4[b]: “... to reduce the clock frequency and the voltage in said one of the domains to reduce the slack.”

90. In my opinion, Iyer teaches this limitation. As noted above with respect to element 4[a], Iyer teaches monitoring slack in order to determine whether a reduction in clock frequency is appropriate. And as Iyer notes and a POSITA would know, “slower clock domains” can “operate at a smaller supply voltage.” Ex. 1005 at §3.4 (“The slower clock domains could also operate at a smaller supply voltage, thus producing additional power savings.”). For this reason, Iyer’s “method for dynamic clock management involves changing the operating frequencies (and supply voltages).” Ex. 1005 at §3.4; *see also id.* at §2.3 (“Such synchronous blocks

whose speed may be gracefully scaled down, can also run at a lower voltage, thus providing additional power savings.”).

91. Additionally, while Claim 4 appears to separately claim frequency reduction and voltage reduction, as discussed above, the two are relational one another. Iyer gives the formula $f_i \propto \frac{(V_{dd,i} - V_t)^\alpha}{V_{dd,i}}$ where the voltage and clock speed associated with each clock domain i are $V_{dd,i}$ and f_i , the threshold voltage is V_t , and α (sensitivity) is “is a technology-dependent factor, which is 1.2 to 1.6 for current technologies.” *Id.* Ex. 1005 at §2.4. In the simplest iteration, Iyer discloses reducing clock frequency from full speed f_{high} to low speed f_{low} in response to low occupancy (slack):



Ex. 1005 at Fig. 3, §3.4 (“The processor starts up with all clocks running at full speed. To minimize the performance impact of clock speed reduction, we monitor the occupancy of each issue queue and we reduce the frequency to a lower

value only when the queue occupancy drops below a low threshold for N consecutive cycles.”)

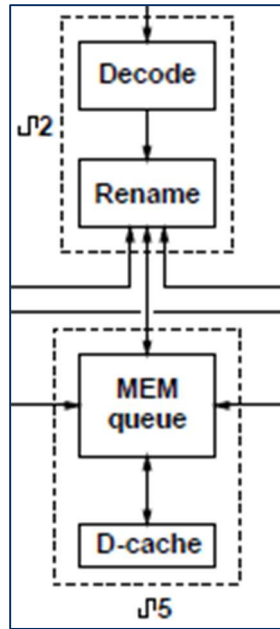
92. Thus, the relationship at full speed is $f_{high} \propto \frac{(V_{dd,high}-V_t)^\alpha}{V_{dd,high}}$. At low speed, the relationship is $f_{low} \propto \frac{(V_{dd,low}-V_t)^\alpha}{V_{dd,low}}$. Given that $f_{low} < f_{high}$ it follows that

$\frac{(V_{dd,low}-V_t)^\alpha}{V_{dd,low}} < \frac{(V_{dd,high}-V_t)^\alpha}{V_{dd,high}}$. Therefore, $V_{dd,low} < V_{dd,high}$. A POSITA would also

have understood that in the context of dynamic voltage and frequency scaling $V \propto \sqrt{f}$ such that $V_{low} < V_{high}$. Finally, a POSITA would understand that the purpose of Iyer’s invention is to enable—wherever possible—a reduction in a domain’s frequency, and consequentially that domain’s voltage. Therefore, voltage will be reduced when frequency is reduced. Conversely, a POSITA would also know that frequency cannot be substantially increased without a corresponding increase in supply voltage, or the signal will deteriorate. Thus, Iyer implicitly and expressly teaches scaling both voltage and frequency in response to slack.

4. Claim 5: “The microprocessor of claim 1, further comprising a queue for communication between at least two of the domains.”

93. In my opinion, Iyer teaches this limitation. It should be noted again that this limitation only claims one queue between domains. Iyer teaches, for instance, a queue between domains 2 and 5:



Ex. 1005 at Fig. 1 (cropped), §3.2 (“There are three issue queues in the design: one for integer instructions (clock domain 3), one for floating-point instructions (clock domain 4) and one for loads and stores (clock domain 5).”)

94. Iyer discloses several “issue queues in the design” including “one for loads and stores (clock domain 5).” Ex. 1005 at §3.4 (“There are three issue queues in the design: one for integer instructions (clock domain 3), one for floating-point instructions (clock domain 4) and one for loads and stores (clock domain 5).”). Iyer also teaches other integer queues, including “one for integer instructions (clock domain 3)” and “one for floating-point instructions (clock domain 4).” *Id.* Iyer further teaches that their “approach uses asynchronous FIFOs [] between clock domains.” Ex. 1005 at §3 (“Our approach uses asynchronous FIFOs [6] between clock domains and this in turn relies on synchronizers. For comparison, we have also

included models for FIFOs using arbiters for clock pausing whenever metastable conditions occur [9, 13].”). As a POSITA would have understood, a first-in-first-out system (FIFO) is a well-known type of queue.

5. Claim 8

a. Limitation 8[pre]: “A method of operating a microprocessor, the method comprising:”

95. In my opinion, as discussed with respect to Claim element 1[pre], Iyer teaches a microprocessor. §VII.A.1. Iyer further discloses a “method” of operating said microprocessor. *See, e.g.*, Ex. 1005 at §3.4 (“Figure 3 shows the outline of our method.”).

b. Limitation 8[a]: “providing a plurality of domains in the microprocessor;”

96. In my opinion, as discussed with respect to Claim element 1[a], Iyer teaches providing a plurality of domains in the microprocessor. §VII.A.1.

c. Limitation 8[b]: “clocking each of the domains separately at a clock frequency”

97. In my opinion, as discussed with respect to Claim element 1[b], Iyer teaches clocking each of the domains separately at a clock frequency. §VII.A.1.

d. Limitation 8[c]: “applying a voltage to each of the domains separately;”

98. In my opinion, as discussed with respect to Claim element 1[c], Iyer teaches applying a voltage to each of the domains separately. §VII.A.1.

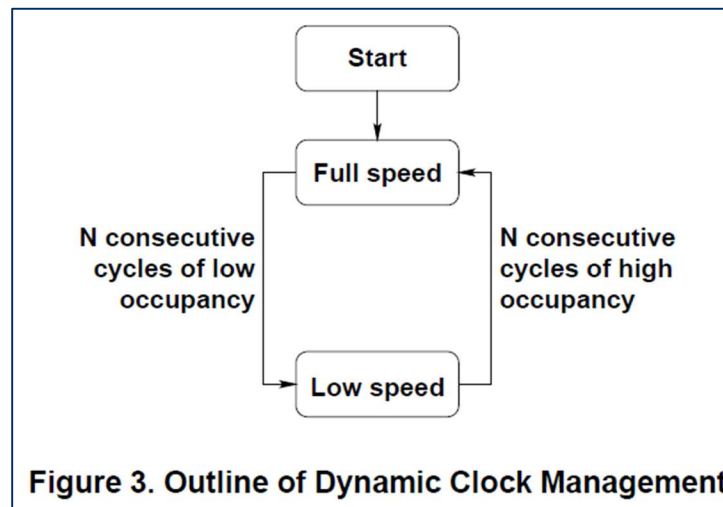
- e. **Limitation 8[d]: “operating the microprocessor such that each domain operates synchronously, while the domains operate asynchronously relative to one another; and”**

99. In my opinion, Iyer teaches this limitation. Iyer teaches “using a Globally Asynchronous Locally Synchronous (GALS) organization,” Ex. 1005 at Abstract, and focuses on the design of “a microprocessor in a GALS style.” *Id.* at §1 (“If we design a microprocessor in a GALS style with multiple clock domains (also called Multiple Clock, Single Voltage, or MCSV core), how much performance overhead will it incur over a fully synchronous machine?”). Notably, operating a processor in this manner was already well known in the art for almost two decades. Ex. 1005 at §1.1 (“GALS systems have been studied in detail by Chapiro in his 1984 thesis [4].”); *see, e.g.*, Ex. 1012 at § 2.1 (“The problems of synchronous design arc, in general, well understood. For our locally-synchronous modules, it is convenient to borrow the structure and notation of some synchronous discipline. We will use the strict two-phase clocking discipline of Noice [36] (although we could as easily use any other reasonable scheme) because although we will focus more on the global non-synchronous interaction, we will still need a consistent notation to describe the interactions between the non-synchronous machinery and the synchronous components, and among the synchronous components themselves.”).

- f. **Limitation 8[e]: “dynamically controlling the clock frequency and the voltage in each of the plurality of**

domains independently of the clock frequencies and the voltages in others of the plurality of domains.”

100. In my opinion, Iyer teaches this limitation. As discussed with respect to claim elements 1[b] and 1[c], Iyer teaches the ability to dynamically control the clock frequency and voltage in each domain independently of the other domains. §VII.A.1. Iyer also teaches the method of dynamically changing frequency and voltage. *See* Ex. 1005 at §1 (“How can we exploit the extra flexibility offered by independent clock domains in a GALS processor and what is the impact of using a Multiple Clock Dynamic Voltage (MCDV) scheme on power/performance metrics?”). Iyer also teaches that “Figure 3 shows the outline of our method.” Ex. 1005 at §3.4. Figure 3 teaches dynamically controlling clock frequency:



Ex. 1005, Fig. 3; *see also id.* at §3.4 (“Our method for dynamic clock management involves changing the operating frequencies (and supply voltages) of the integer, memory and floating-point clock domains in the GALS processor (indicated

as clock domains 3, 4 and 5 in Figure 1).”).

6. **Claim 10: “The method of claim 8, wherein step (d) comprises providing a queue for communication between at least two of the domains.”**

101. In my opinion, as discussed with respect to Claim 5, Iyer teaches providing a queue for communication between at least two of the domains. §VII.A.4.

7. **Claim 12: “The microprocessor of claim 8, wherein there are at least four of said domains.”**

102. In my opinion, as discussed with respect to Claim 3, Iyer teaches the method of operating the microprocessor wherein there are at least four domains. §VII.A.2.

B. Ground II.A: Claims 1, 3, 4, 5, 8, 10, and 12 Are Rendered Obvious By Njolstad (Ex. 1009)

1. Claim 1

- a. **Limitation 1[pre]: “A multiple clock domain microprocessor comprising:”**

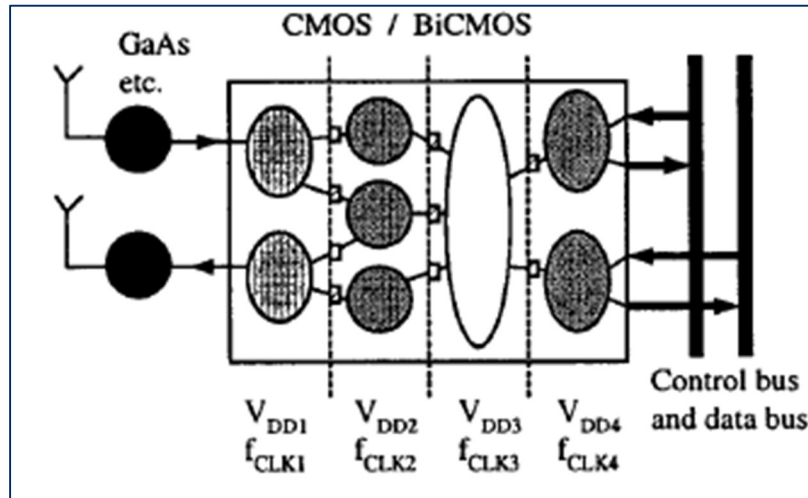
103. In my opinion, to the extent the preamble is limiting, Njolstad teaches this limitation. For instance, Njolstad teaches a digital signal processor (DSP) module, which a POSITA would have understood to be a type of microprocessor. *See* Ex. 1009 at §IV.D (“Figure 4: The processing element (e.g. a DSP module) executes its calculations fast enough to satisfy the actual performance requirement.”). Njolstad further discloses that its teachings are intended to further existing work on scaling in a “microprocessor.” Ex. 1009 at §III (“Use of complete

systems with variable supply voltages has been demonstrated in academia, e.g. [8][19][20]. Recently, voltage scaling is introduced in several consumer products for mobile computing based on e.g. Transmeta's Crusoe microprocessor [26]."). Njolstad also discloses multiple clock domains. For instance, Njolstad teaches a "globally-asynchronous, locally-synchronous paradigm." Ex. 1009 at Abstract ("A low-power amendment to the globally-asynchronous, locally-synchronous paradigm (GALS) is presented. A proposed socket interface provides locally dynamic voltage scaling (LDVS) adapted to the actual processing rate requirements for each module."). As a POSITA would have understood, a GALS paradigm contains multiple clock domains. Njolstad discloses, for instance, "local clock generation" in each domain. Ex. 1009 at Abstract ("Local clock generation by ring oscillators tracks the local device speed at the local supply voltage."); *see also id.* at §IV.D ("The energy per operation will therefore be regulated to the lowest possible value for the actual rate-requirement, the given technology parameters and temperature, since the local clock is generated by a ring oscillator which tracks these local conditions.").

b. Limitation 1[a]: "a plurality of domains;"

104. In my opinion, Njolstad teaches this limitation. As discussed with respect to limitation 1[pre], Njolstad's GALS paradigm includes a plurality of

domains. §VII.B.1.a. This plurality of clock domains can be seen, for instance, in Njolstad's Figure 3:



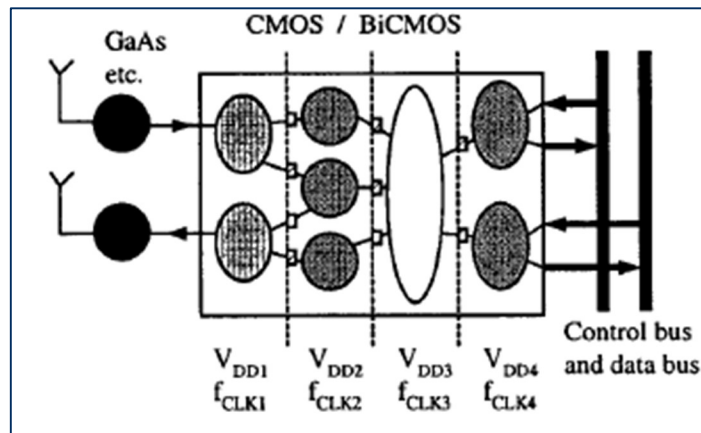
Ex. 1009, Fig. 3 (“A model of a wideband radio access system using multiple supply voltages and logic level shifters.”); IV.A (“An integrated digital transceiver may be partitioned as in Figure 3, where all the modules with their individual performance and latency requirements are clustered into separate supply voltage regions, thus giving potentially the lowest energy per operation while satisfying the performance requirements.”).

c. Limitation 1[b]

- i. **“for each of the plurality of domains, a clock for separately generating a clock signal at a frequency for that domain ...”**

105. In my opinion, Njolstad teaches this limitation. Njolstad discloses a “locally generated clock rate” for each domain, noting that the “local clock is generated by a ring oscillator.” Ex. 1009 at §IV.D (“The control signal *RATE-*

REQUIREMENT_i can be used to scale down the local supply voltage such that the locally generated clock rate is just high enough to perform the actual processing task and to produce output data at the given rate.”). The clocks for separately generating a clock signal at a frequency for each domain are again visible in Figure 3, as f_{CLK1} , 2, 3, and 4, respectively:

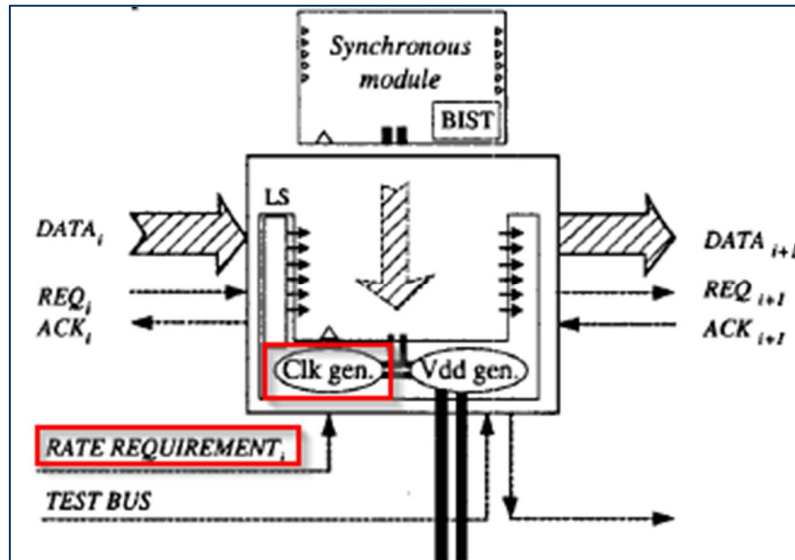


Ex. 1009, Fig. 3.

- ii. **“...the frequency being dynamically changeable independently of the frequencies of the clock signals generated for others of the plurality of domains; and”**

106. In my opinion, Njolstad teaches this limitation. In each domain, the local frequency (“Clk gen.”) is based on the local processing demands (“RATE REQUIREMENT_i”) and scaled accordingly. Ex. 1009 at §IV.D (“A clock signal with the correct frequency given by the logical depth of the synchronous module and by the actual supply voltage level, is active for a predefined number of clock periods

for the actual task.”). For instance, a given domain or “module” is shown in Njolstad’s Figure 5:



Ex. 1009, Fig. 5 (annotated) (“A synchronous version of the actual processing element (DSP module) is inserted into the Socket Interface.”).

107. The local “rate requirement can dynamically be configured” such that such that “the locally generated clock rate is just high enough to perform the actual processing task.” Ex. 1009 at §IV.D (“The fixed rate requirement or the fixed average rate requirement can dynamically be configured, by the special control signal $RATE-REQUIREMENT_i$ The control signal $RATE-REQUIREMENT_i$ can be used to scale down the local supply voltage such that the locally generated clock rate is just high enough to perform the actual processing task and to produce output data at the given rate.”). As Njolstad notes, from “a system point of view, faster operation of the processing element than given by the rate requirement does not give

any better system performance, while slower operation than the requirement can of course not be tolerated.” Ex. 1009 at §IV.D. For this reason, Njolstad implements a dynamically changeable frequency for each domain.

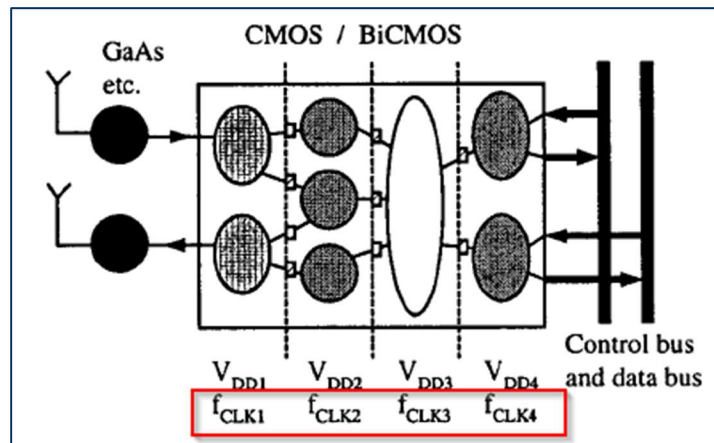
d. Limitation 1[c]: “for each of the plurality of domains, a voltage input for receiving a voltage which is dynamically changeable independently of the voltages applied to said others of the plurality of domains.”

108. In my opinion, Njolstad teaches this limitation. As a POSITA would have understood, and as Njolstad expressly discloses, voltage and frequency are varied together. They are mathematically related. For instance, frequency cannot be substantially increased without increasing voltage, or the clock signal integrity will deteriorate. And decreasing clock frequency without decreasing voltage would defeat the energy-saving focus of the technology. Ex. 1009 at §III.C (“The candidate methods for local supply voltage generation should be explored further including their suitability for future technologies, because it is only by local supply voltage selection rate-optimal energy saving can be attained.”). For this reason, Njolstad’s core focus relates to “locally dynamic voltage scaling (LDVS) adapted to the actual processing rate requirements” for each domain. Ex. 1009 at Abstract (“A proposed socket interface provides locally dynamic voltage scaling (LDVS) adapted to the actual processing rate requirements for each module.”). Njolstad dedicates an entire section this concept. *See* Ex. 1009 at §IV.C. (“C. Means to generate different local supply voltages”). As Njolstad’s concludes, “[w]e have described GALS-LDVS”

with “locally rate-adapted dynamic voltage scaling.” Ex. 1009 at §V; *id.* at §IV.D (“The control signal $RATE-REQUIREMENT_i$ can be used to scale down the local supply voltage such that the locally generated clock rate is just high enough to perform the actual processing task and to produce output data at the given rate.”).

2. Claim 3: “The microprocessor of claim 1, wherein there are at least four of said domains.”

109. In my opinion, Njolstad teaches this limitation. Njolstad’s Figure 3, for instance, shows four clock domains:



Ex. 1009, Fig. 3.

3. Claim 4

a. Limitation 4[a]: “The microprocessor of claim 1, wherein the microprocessor is programmed to determine a slack in processing in one of the domains and . . .”

110. In my opinion, Njolstad teaches this limitation. Njolstad teaches that its “rate requirement can dynamically be configured, by the special control signal $RATE-REQUIREMENT_i$ ” in order to ensure that there is not “faster operation of the

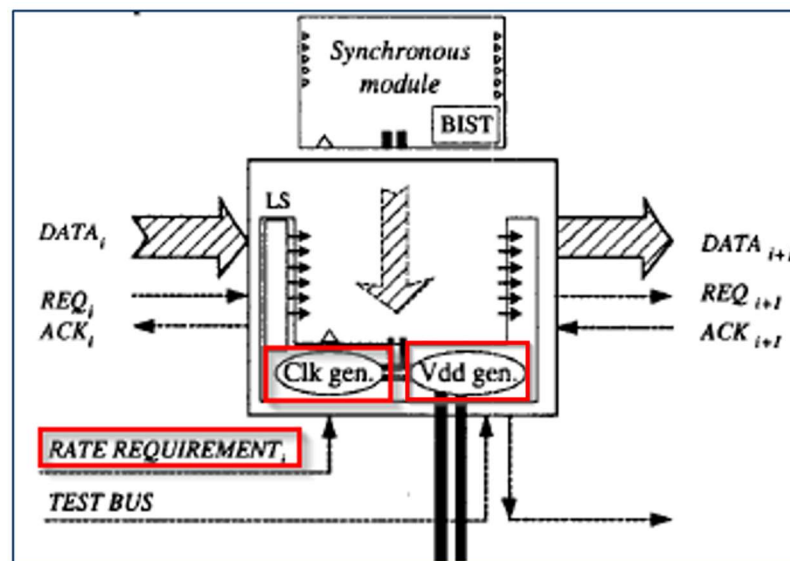
processing element than” required. Ex. 1009 at §IV.D (“The fixed rate requirement or the fixed average rate requirement can dynamically be configured, by the special control signal *RATE-REQUIREMENT_i*. From a system point of view, faster operation of the processing element than given by the rate requirement does not give any better system performance, while slower operation than the requirement can of course not be tolerated.”) Determining the rate requirement necessarily entails determining slack. *Id.* This is because the rate requirement is itself targeting the elimination of excess processing capacity in the processor. Because the rate requirement is dynamic, slack must be measured constantly. Njolstad also teaches reducing clock frequency “when no data processing is performed.” *Id.* (“Furthermore, the number of clock periods required per processing operation can be predefined, and the local clock can be stopped after the operation is performed. By stopping the clock when no data processing is performed, the power consumption will be further reduced.”). Thus, as further discussed with respect to limitation 4[b], below, Njolstad expressly and implicitly discloses monitoring slack in a given domain.

b. Limitation 4[b]: “... to reduce the clock frequency and the voltage in said one of the domains to reduce the slack.”

111. In my opinion, Njolstad teaches this limitation. Njolstad teaches monitoring slack, and further teaches dynamically adjusting frequency and voltage

in response to said slack. Njolstad explicitly discloses that “[t]he control signal RATE-REQUIREMENT_i can be used to scale down the local supply voltage such that the locally generated clock rate is just high enough to perform the actual processing task and to produce output data at the given rate.” Ex. 1009 at §IV.D. This disclosure shows that Njolstad teaches (i) monitoring slack and reducing (ii) local frequency and (iii) local voltage in order to reduce or eliminate said slack. A POSITA would understand this from reviewing Njolstad.

112. Moreover, as Njolstad notes, slack monitoring and reduction is executed “locally”, *id.*—that is, independently for each domain:



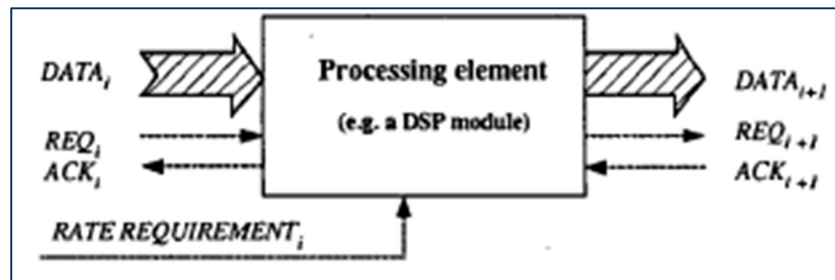
Ex. 1009, Fig. 5., §IV (“We have described GALS-LDVS, a low-power amendment to the GALS paradigm where locally rate- adapted dynamic voltage scaling can be provided by means of the proposed Socket Interface.”).

113. In sum, within a given clock domain, the rate requirement signal is determined by monitoring slack. This signal determines whether scaling is required with respect to the frequency generated (“Clk gen.”) and corollary the supply voltage (“Vdd gen.”). Ex. 1009, Fig. 5, §IV.D (“The control signal *RATE-REQUIREMENT_i* can be used to scale down the local supply voltage such that the locally generated clock rate is just high enough to perform the actual processing task and to produce output data at the given rate.”).

4. Claim 5: “The microprocessor of claim 1, further comprising a queue for communication between at least two of the domains.”

114. In my opinion, Njolstad renders obvious this limitation. As would have been obvious to a POSITA, and as Njolstad expressly notes, inter-domain communication in a GALS paradigm requires “some given protocol, such that *e.g.*, the processing does not start before the module has input data ready.” Ex. 1009 at §IV.D (“The input and output data sets may be passed along with handshake signals which implement some given protocol, such that *e.g.* the processing does not start before the module has input data ready.”). The simplest and most well-known mechanism for enabling asynchronous inter-domain communication is the use of buffer—such as a queue—such that the producer and consumer domains may operate at distinct frequencies. Njolstad, in particular, teaches toward the straightforward implementation of a queue, because the data transfer shown is

unidirectional, as shown in Figure 4. The processing element of Figure 4 must incorporate a temporary memory storage such as a buffer, register or a queue, for processing the received data. By using a buffer or queue, POSITA would understand that the design disclosed in this prior art will be more efficient by having a multiple entry buffer or queue to store and manage pending data waiting to be processed



Ex. 1009, Fig. 4. (“The processing element (e.g. a DSP module) executes its calculations fast enough to satisfy the actual performance requirement.”).

115. Thus, as a POSITA would understand, a basic FIFO queue would manage the asynchronous operation of producer and consumer domains by simply buffering data until the consumer domain is ready to process it. The ‘443 Patent acknowledges that this implementation would be obvious to a POSITA, noting that its own purported invention simply coopts existing queue designs. Ex. 1001 at 5:1 (“Our design uses existing queue structures in a superscalar processor core to isolate the different clock domains in a way that minimizes the need for inter-domain synchronization.”). Queues had been the standard mechanism for enabling asynchronous communication in processors for over a decade. For instance, a 1989

paper titled *Micropipelines*, which Iyer refers to as “the classic introduction to asynchronous design,” illustrates a FIFO queue approach in detail. Ex. 1005 at §1.1 (“Sutherland’s paper on Micropipelines [1] contains the classic introduction to asynchronous design.”); Ex. 1014 (Micropipelines) at 722 (“FIFOs often connect senders and receivers that have separate clocks.”). Thus, a POSITA would be well aware of this approach and motivated to adopt it in accordance with Njolstad’s teaching that “some given protocol” is required for asynchronous communication. Ex. 1009 at §IV.D (“The input and output data sets may be passed along with handshake signals which implement some given protocol, such that e.g. the processing does not start before the module has input data ready.”). And given the widespread and longstanding implementation of such queues, a POSITA would reasonably expect to be successful in implementing said queue.

5. Claim 8

a. Limitation 8[pre]: “A method of operating a microprocessor, the method comprising:”

116. In my opinion, to the extent the preamble is limiting, Njolstad teaches this limitation. As discussed with respect to limitation 1[pre], Njolstad teaches a microprocessor. §VII.B.1. Njolstad further teaches a “method” and “means” for operating said microprocessor. *See, e.g.*, Ex. 1009 at §IV.C (“This method is considered to be a viable alternative to the dynamically programmable generator described above, but performance degradation due to IR drop, and other practical

aspects of this method, have to be studied further.”); *id.* (“means to generate different local supply voltages.”).

b. Limitation 8[a]: “providing a plurality of domains in the microprocessor;”

117. In my opinion, Njolstad teaches this limitation. As discussed with respect to Claim element 1[a], Njolstad teaches providing a plurality of domains in the microprocessor. §VII.B.1.

c. Limitation 8[b]: “clocking each of the domains separately at a clock frequency”

118. In my opinion, Njolstad teaches this limitation. As discussed with respect to Claim element 1[b], Njolstad teaches clocking each of the domains separately at a clock frequency. §VII.B.1.

d. Limitation 8[c]: “applying a voltage to each of the domains separately;”

119. In my opinion, Njolstad teaches this limitation. As discussed with respect to Claim element 1[c], Njolstad teaches applying a voltage to each of the domains separately. §VII.B.1.

e. Limitation 8[d]: “operating the microprocessor such that each domain operates synchronously, while the domains operate asynchronously relative to one another; and”

120. In my opinion, Njolstad teaches this limitation. Njolstad discloses a method of operating a microprocessor in a “globally-asynchronous locally-synchronous paradigm.” Ex. 1009 at §I (“For increased design productivity,

simplified system design, higher yield, lower energy consumption and higher performance, it is imperative to depart from the fully synchronous chip design paradigm, and replace it with a globally-asynchronous locally-synchronous paradigm (GALS).”).

f. Limitation 8[e]: “dynamically controlling the clock frequency and the voltage in each of the plurality of domains independently of the clock frequencies and the voltages in others of the plurality of domains.”

121. In my opinion, Njolstad teaches this limitation. As discussed with respect to Claim element 1[b] and 1[c], Njolstad discloses dynamically controlling the frequency and voltage in each domain independently. §VII.B.1.

6. Claim 10: “The method of claim 8, wherein step (d) comprises providing a queue for communication between at least two of the domains.”

122. In my opinion, Njolstad renders obvious this limitation. As discussed with respect to Claim 5, Njolstad renders obvious providing a queue for communication between at least two of the domains. §VII.B.4.

7. Claim 12: “The microprocessor of claim 8, wherein there are at least four of said domains.”

123. As discussed with respect to Claim 3, Njolstad teaches the method of operating the microprocessor wherein there are at least four domains. §VII.B.2.

C. Ground IIB: Claims 5 and 10 Are Rendered Obvious By Njolstad (Ex. 1009) in view of Liljeberg (Ex. 1010)

1. Motivation to Combine Njolstad and Liljeberg

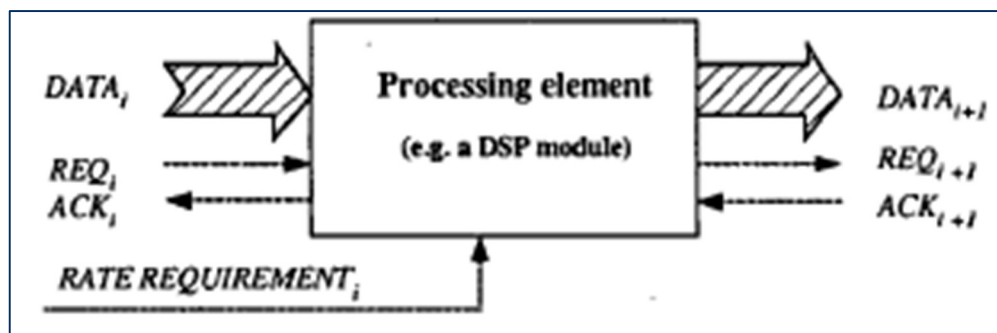
124. To the extent that it would not already have been obvious to a POSITA to implement Njolstad's teachings with a queue for asynchronous communication, this element is expressly disclosed by Liljeberg. Liljeberg is focused on solving the very same issues Njolstad addresses. *Compare* Ex. 1009 at §I ("However, as the semiconductor technology scales down and complexity increases, fully synchronous designs will eventually not be feasible on the chip level."), *with* Ex. 1010 at §1 ("Also, the system-on-chip design strategy brings new challenges to manage system level timing and to arrange an efficient interaction between separate modules."). Njolstad and Liljeberg also offer aligned solutions. *Compare* Ex. 1009 at §I ("For increased design productivity, simplified system design, higher yield, lower energy consumption and higher performance, it is imperative to depart from the fully synchronous chip design paradigm, and replace it with a globally-asynchronous locally-synchronous paradigm (GALS)."), *with* Ex. 1010 at §1 ("A viable alternative to overcome the problems of a globally clocked system is the globally asynchronous locally synchronous approach [1,4], where each distinct module of the system is synchronized to an optimized local clock, and communication between these modules is arranged asynchronously."). Furthermore, both references are focused on implementation in a CMOS integrated circuit. *See* Ex. 1009 at Fig. 3 (depicting a CMOS model); Ex. 1010 at §4 ("4. CMOS Implementation"). Thus, a POSITA would have reasonably expected to be successful in integrating the teachings of

Njolstad and Liljeberg. Furthermore, whereas Njolstad is focused on dynamically scaling independent domains, Liljeberg is dedicated to inter-domain communication. Both focuses must be addressed to enable a coherent and energy efficient GALS design, and thus a POSITA would have been motivated to combine these teachings. As a POSITA would understand, as a matter of flow control, implementing buffers in addition to a full handshake mechanism between producer and consumer domains is advantageous in minimizing latency and optimizing throughput. While the handshake ensures data integrity and synchronization, a queue allows the producer to continue sending data without waiting for the consumer to be immediately ready, thus reducing latency. Buffers also help manage bursty traffic by absorbing spikes in data generation and decoupling the producer and consumer, allowing each to operate at their own optimal speeds. By way of analogy, in a bank with tellers (consumers) and customers (producers), a full handshake would require each customer to wait for a signal from a teller before approaching, i.e., waiting outside until called in. This creates apparent inefficiencies. However, with a waiting area (buffer) inside the bank, customers can queue up and be ready to step forward as soon as a teller is available, ensuring a smooth and continuous flow of service without unnecessary waiting times. Similarly, a POSITA would be motivated to employ a queue irrespective of whether there is a full handshake protocol in place.

Moreover, the main objective of both teachings is to address design and provide feasibility studies for efficient design of GALS based systems.

2. Claim 5: “The microprocessor of claim 1, further comprising a queue for communication between at least two of the domains.”

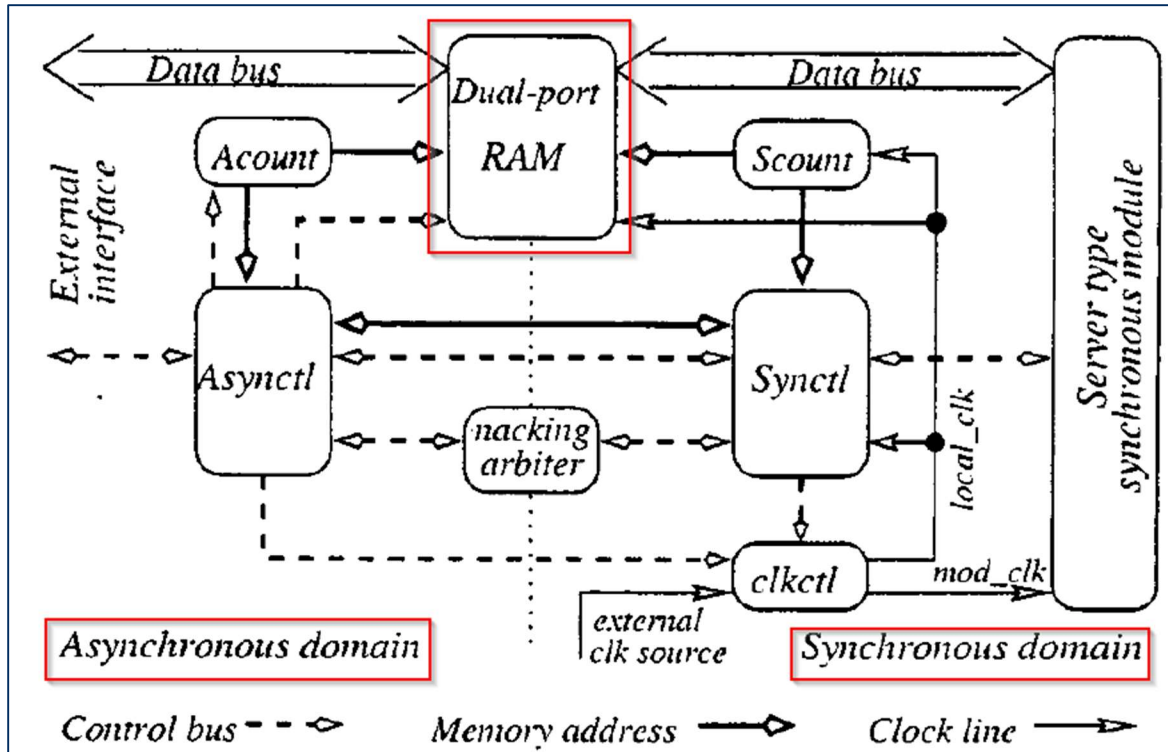
125. In my opinion, Njolstad and Liljeberg teach or render obvious this limitation. As would have been obvious to a POSITA, and as Njolstad expressly notes, inter-domain communication in a GALS paradigm requires “some given protocol, such that *e.g.*, the processing does not start before the module has input data ready.” Ex. 1009 at §IV.D (“The input and output data sets may be passed along with handshake signals which implement some given protocol, such that *e.g.* the processing does not start before the module has input data ready.”). The simplest and most well-known mechanism for enabling asynchronous inter-domain communication is the use of a buffer—*i.e.*, a queue—such that the producer and consumer domains may operate at distinct frequencies. Njolstad, in particular, teaches toward the straightforward implementation of a queue, because the data transfer shown is unidirectional, as shown in Figure 4:



Ex. 1009, Fig. 4. (“The processing element (e.g. a DSP module) executes its calculations fast enough to satisfy the actual performance requirement.”).

126. Thus, a queue may manage the asynchronous operation of producer and consumer domains by simply buffering data until the consumer is ready to process it.

127. To this end, Liljeberg articulates a detailed implementation of the cross-boundary communication “protocol” expressed by Njolstad. Liljeberg teaches transferring data “to the local synchronous model through the FIFO queue.” Ex. 1010 at §3 (“In the (1) *data receiving mode*, data provided by a master-type SAInt unit of another system block, are transferred to the local synchronous module through the FIFO queue.”). This can be seen, for instance, in Liljeberg’s Figure 2, wherein the queue is implemented as dual-ported RAM:



Ex. 1010, Fig. 2 (annotated) (“Server-type SAInt architecture.”).

128. In its own words, Liljeberg presented “an asynchronous design approach to interface locally clocked modules” with “FIFO queue [] interface units.” Ex. 1010 at §6 (“We have presented an asynchronous design approach to interface locally clocked modules in a ULSI system. In this method, data is transferred asynchronously in variable-length packets between concurrently operating synchronous modules. The size of the FIFO queue of the interface units can be easily changed by just replacing the memory and increasing/decreasing the address counter widths, while the control logic can be left untouched.”). Thus, Njolstad in view of Liljeberg teaches a queue for communication between at least two domains.

3. Claim 10: “The method of claim 8, wherein step (d) comprises providing a queue for communication between at least two of the domains.”

129. In my opinion, Njolstad in view of Liljeberg renders obvious this limitation. As discussed with respect to Claim 5, Njolstad in view of Liljeberg teaches providing a queue for communication between at least two of the domains.

§VII.C.2.

D. Ground III.A: Claims 1, 3, 4, 5, 8, 10, and 12 Are Rendered Obvious By Soerensen (Ex. 1006)

1. Claim 1

a. Limitation 1[pre]: “A multiple clock domain microprocessor comprising:”

130. In my opinion, to the extent the preamble is limiting, Soerensen teaches this limitation. Soerensen teaches a processor, and further teaches that “processor may, for instance, be fabricated on a single semiconductor chip,” *i.e.*, a microprocessor. Ex. 1006 at [0028]. (“FIG. 1 illustrates a block diagram of an exemplary processor 20. Processor 20 may, for instance, be fabricated on a single semiconductor chip.”). This block diagram shows all the key components of a bus based standard microprocessor. This is a shared memory design where the DSP and MCU utilize the same bus to access a common memory system (28). *Id.* Further, Soerensen teaches multiple clock domains. *See, e.g.*, Ex. 1006 at [0037] (“The clock signals in each clock domain are provided by various select, divider, and gating logic and distributed to the components of the processor via a plurality of clock trees 50a-

50h.”); *id.* (“For example, clock control 110 includes multiplexers in each of clock domains (e.g., multiplexers 70a, 70b, and 70c, respectively) which may independently select whether clock signal 84 is required in order to operate the components in the corresponding clock domain.”). Soerensen’s Claim 6, for instance, claims the following:

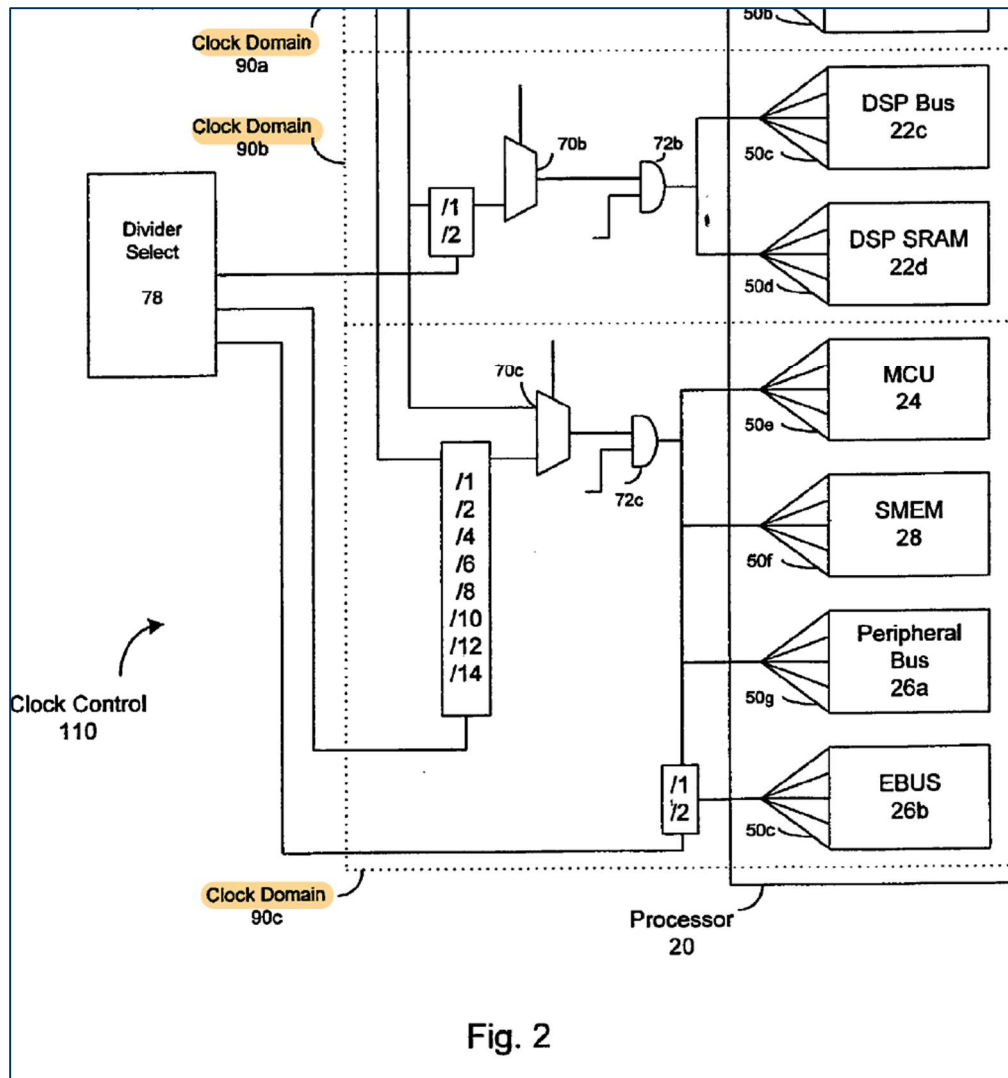
The method of claim 1, wherein the at least one processor has a plurality of clock domains, each clock domain providing a clock signal at a plurality of frequencies to a plurality of components of the at least one processor; and

wherein the act of determining the clock frequency requirement includes determining the minimum clock frequency required in each of the plurality of clock domains.

Ex. 1006 at 8, Cl. 6.

b. Limitation 1[a]: “a plurality of domains;”

131. In my opinion, Soerensen teaches this limitation. As discussed with respect to Claim 1[pre], Soerensen teaches a “plurality of clock domains.” Ex. 1006 at §8, Cl. 6. And Soerensen’s Figure 2, for instance, shows this plurality of clock domains:



Ex. 1006, Fig. 2 (cropped and annotated), ¶11 (“FIG. 2 illustrates a clocking scheme and control to generate and distribute clock signals to the various components of the processor illustrated in FIG. 1.”).

c. Limitation 1[b]

- i. “for each of the plurality of domains, a clock for separately generating a clock signal at a frequency for that domain ...”

132. In my opinion, Soerensen teaches this limitation. Soerensen teaches separately generating clock signals such that each domain may have a separate frequency. Ex. 1006 at [0040] (“For example, clock tree 50a drives the clock signal of clock domain 90a and distributes it to the low level components of DSP 22a. Similarly, clock tree 50e drives the clock signal of clock domain 90c and distributes it to the low level logic components of MCU 24a.”). Soerensen explains that each domain receives its own signal at an independent frequency. *See* Ex. 1006 at [0037] (“For example, clock control 110 includes multiplexers in each of clock domains (e.g., multiplexers 70a, 70b, and 70c, respectively) which may independently select whether clock signal 84 is required in order to operate the components in the corresponding clock domain.”). This generates an independent and asynchronous frequency in each domain. *Id.* To this end, Soerensen’s Claim 6 claims the following:

The method of claim 1, wherein the at least one processor has a plurality of clock domains, each clock domain providing a clock signal at a plurality of frequencies to a plurality of components of the at least one processor; and

wherein the act of determining the clock frequency requirement includes determining the minimum clock frequency required in each of the plurality of clock domains.

Ex. 1006 at 8, Cl. 6.

- ii. **“...the frequency being dynamically changeable independently of the frequencies of the clock signals generated for others of the plurality of domains; and”**

133. In my opinion, Soerensen teaches this limitation. Soerensen discloses, for instance, “dynamically adjust[ing] the frequency of the clock signals provided . . .” Ex. 1006 at [0023] (“Accordingly, one way to reduce the power consumption of a device is to dynamically adjust the frequency of the clock signals provided to a processor to reflect the clock frequency requirement of the processor.”). Soerensen further discloses that this dynamic frequency scaling is executed independently for each domain. Ex. 1006 at [0081] (“At some later instant, the MCU may finish its task and require only a 45 MHz clock in order to meet the computational demands of the current processing state.”). In this instance, Soerensen teaches that “the change in frequency required by the MCU (e.g., 45 MHz) may be generated by adjusting the divider factor of the appropriate clock domain.” *Id.* at [0082]. Thus, the frequency in, for example, the MCU domain is varied dynamically and independent of the frequencies generated for the plurality of other domains. As another example, Soerensen notes that “[a]t some later instant, the DSP may need to process another high priority task.” *Id.* at [0083]. In such an instance, the software will “identify the new clock frequency requirement of the DSP, for instance, 240 MHz” and adjust that domain’s frequency accordingly. *Id.* (“As such, software may identify the new clock frequency requirement of the DSP, for instance, 240 MHz and write this value into the appropriate register.”). Soerensen thus emphasizes that the frequency adjustments within each domain are made independently of the other domains. As a

further example, Soerensen discloses that, for instance, the frequencies within “clock domain 90b” and “clock domain 90c” may each be reduced by disparate amounts in order to provide the requisite frequencies to the subsidiary processor components in each domain. Ex. 1006 at [0038] (“For example, divider 62 may provide the clocked components operating in clock domain 90 b with a clock signal having half the frequency of clock signal 84 when the clock signal from the system PLL multiplier is selected. Similarly, divider 64 may provide the clocked components operating in clock domain 90 c with a clock signal having a frequency reduced by factors of 2, 4, 6, 8, 10, 12, and 14.”).

d. Limitation 1[c]: “for each of the plurality of domains, a voltage input for receiving a voltage which is dynamically changeable independently of the voltages applied to said others of the plurality of domains.”

134. In my opinion, Soerensen teaches this limitation. For instance, Soerensen teaches that “by monitoring the clock frequency requirement” the voltage can be “dynamically controlled” to reduce slack. Ex. 1006 at [0048] (“Applicants have identified and appreciated that by monitoring the clock frequency requirement of a processor, the voltage supplied to the processor can be dynamically controlled such that the processor spends less time supplying a voltage to the processor in excess of the requirement.”). As a POSITA would understand, voltage is varied proportionally to the frequency variation in each domain. Predictably, Soerensen claims “determining the voltage requirement based on the at least one associated

frequency of the clock domains.” Ex. 1006 at 9, Cl. 11 (“The method of claim 10, wherein the act of determining the voltage requirement includes an act of determining the voltage requirement based on the at least one associated frequency of the clock domains that are required to be enabled.”). The relationship between a domain’s frequency and the requisite voltage is also outlined in Soerensen’s Figure 5:

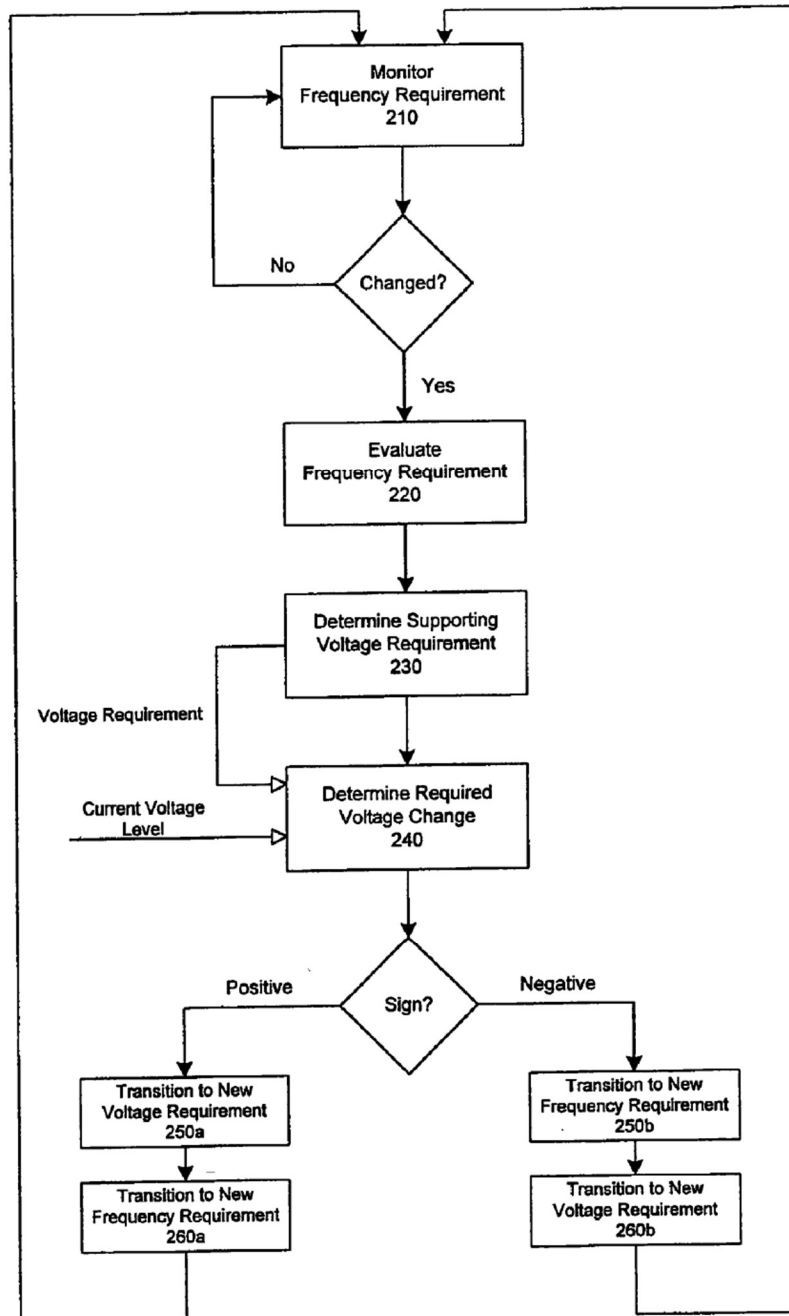


Fig. 5

Ex. 1006, Fig. 5, [0014] (“FIG. 5 illustrates a method of providing dynamic power control to a processor according to one embodiment of the present invention;”).

135. As Soerensen notes, “to avoid such potential problems during power state transitions, when going to a higher power state, the voltage may be stepped up before the clock frequency is increased.” Ex. 1006 at [0064]. Conversely, when “going to a lower power state, frequency may be first decreased before reducing the voltage supplied to the processor.” *Id*; *see also id.* at [0058] (“As such, in step 220, the clock frequency requirement is evaluated in order to determine a voltage requirement necessary to support the new clock frequency requirement.”). Thus, Soerensen teaches the dynamic variance of both frequency—and consequentially, voltage—in each domain.

2. Claim 3: “The microprocessor of claim 1, wherein there are at least four of said domains.”

136. In my opinion, Soerensen teaches this limitation. To the extent that designing the microprocessor of Claim 1 with four domains would not already be obvious to a POSITA, Soerensen expressly discloses such an embodiment. Ex. 1006 at [0043] (“Plot 300 shows exemplary frequency values of four clock domains of a system clocking scheme of a baseband processor as they might vary in time.”).

3. Claim 4

a. Limitation 4[a]: “The microprocessor of claim 1, wherein the microprocessor is programmed to determine a slack in processing in one of the domains and . . .”

137. In my opinion, Soerensen teaches this limitation. As discussed above, Soerensen renders obvious Claim 1. §VII.D.1. Soerensen further teaches that a

“processor is considered to be consuming excess power when the computational tasks, functions, and operations of the processor may be performed, and/or the processing demands met, at a lower power state.” Ex. 1006 at [0017]. The state described by Soerensen—frequency in excess of the requirement—is a state of “slack” and reducing slack is a primary focus of Soerensen’s disclosures. One note, Soerensen’s use of the term “processor” is intended to encompass “any apparatus that performs logic operations, computational tasks, control functions, etc.” Ex. 1006 at [0017] (“The term “processor” refers, generally, to any apparatus that performs logic operations, computational tasks, control functions, etc. A processor may include one or more subsystems, components, and/or other processors.”). A POSITA would have understood this to include, for instance, an instruction unit or execution unit.

138. Soerensen repeatedly discloses determining slack in one domain. *See, e.g.*, Ex. 1006 at [0058] (“[A] DPC monitors the clock frequency requirement of the processor.”); *id.* at [0092] (“For example, a comparator may map information provided about the clock frequency requirement of the processor (e.g., as stored in a register bank, memory, etc.) to an encoded input.”); *id.* at [0027] (“Accordingly, one embodiment according to the present invention includes identifying a clock frequency requirement of a processor and adjusting the voltage supplied to the processor to support the identified clock frequency requirement.”). Where there is a

“delta” between the requirement and the frequency given, “there may be opportunities to reduce both the clock signal frequencies and the voltage.” Ex. 1006 at [0061] (“In step 240, the current voltage level being supplied to the processor is compared to the voltage requirement determined in step 240 to determine the required voltage change, or delta, required to provide the voltage requirement”); *id.* [0027] (“As the computational demands on a processor vary with time, there may be opportunities to reduce both the clock signal frequencies and the voltage level applied to the processor (i.e., the processor may be transitioned into a lower power state).”). This teaching recurs throughout Soerensen’s disclosures. *See, e.g., id.* at [0041] (“As such, by recognizing the clock frequency requirement of the processor, clock control 100 may configure the frequency distribution to reduce the intervals when excess clock frequencies are being generated.”). As a POSITA would have understood, excess clock frequency is a state of slack.

b. Limitation 4[b]: “... to reduce the clock frequency and the voltage in said one of the domains to reduce the slack.”

139. In my opinion, Soerensen teaches this limitation. As Soerensen notes preliminarily, its invention is directed to reducing power consumption in processors. Ex. 1006 at [0002] (“This invention relates to reducing the power consumption in processors and, more particularly, to methods and apparatus for dynamically controlling the voltage level supplied to a processor based on the processing state

and/or computational demands of the processor by determining the clock frequency requirement and a corresponding voltage requirement of the processor.”). And as discussed with respect to Claim element 4[a], Soerensen teaches monitoring slack and reducing frequency and voltage in response to slack. Ex. 1006 at [0027] (“As the computational demands on a processor vary with time, there may be opportunities to reduce both the clock signal frequencies and the voltage level applied to the processor (i.e., the processor may be transitioned into a lower power state).”). A POSITA would recognize that the ultimate point of reducing frequency in response to slack is to ultimately reduce voltage and thus reduce power consumption. Ex. 1006 at [0018] (“Applicant has observed that the power consumption of a device is related to the square of the voltage supplied to the device’s processor and proportional to the frequency at which the processor operates.”). Unsurprisingly, Soerensen teaches that “by monitoring the clock frequency requirement of a processor, the voltage supplied to the processor can be dynamically controlled such that the processor spends less time supplying a voltage to the processor in excess of the requirement.” Ex. 1006 at [0025].

4. Claim 5: “The microprocessor of claim 1, further comprising a queue for communication between at least two of the domains.”

140. In my opinion, Soerensen renders obvious this limitation. As discussed above, Soerensen renders obvious Claim 1. §VII.D.1. And as discussed with respect

to Claim element 1[pre], Soerensen discloses multiple asynchronous clock domains necessitating interdomain data transfer. *Id.*; *see also* Ex. 1006 at [0028] (“The various subsystems of the processor are connected to a system bus 30 in order to communicate and/or transfer data between the various components, such as, operands or instructions, addresses and/or control signals, etc.”). A POSITA would have understood that said system bus requires a mechanism for coherent transfer between domains, including a queue. It would have been obvious for a POSITA to implement a queue for interdomain transfer because queues were well known in the art and Soerensen does not teach toward an incompatible alternative. By way of example, Hemani—a 1999 publication—outlines a POSITA’s approach to the “communication refinement step, for every pair of communicating SBs, communications requirement analysis is made to select the cheapest of the four possible communication modes: send and forget, strobe, handshake, or FIFO.” Ex. 1007 at §2. In other words, in view of Soerensen alone, a POSITA would understand that a FIFO queue should be implemented where it is the “cheapest” option. The ‘443 Patent itself acknowledges that this implementation would be obvious to a POSITA, noting that its own purported invention simply coopts existing queue designs. Ex. 1001 at 5:1 (“Our design uses existing queue structures in a superscalar processor core to isolate the different clock domains in a way that minimizes the need for inter-domain synchronization.”). Thus, a POSITA would

find it obvious and be motivated to implement a queue in Soerensen's design, and further reasonably expect to be successful in doing so.

5. Claim 8

a. Limitation 8[pre]: "A method of operating a microprocessor, the method comprising:"

141. In my opinion, To the extent the preamble is limiting, as discussed with respect to Claim element 1[pre], Soerensen teaches a microprocessor. §VII.D.1. Soerensen further teaches that its invention relates "to methods" for dynamically controlling said microprocessor. *See, e.g.*, Ex. 1006 at [0002] ("This invention relates to reducing the power consumption in processors and, more particularly, to methods and apparatus for dynamically controlling the voltage level supplied to a processor based on the processing state and/or computational demands of the processor by determining the clock frequency requirement and a corresponding voltage requirement of the processor."); *id.* at [0027] ("Accordingly, one embodiment according to the present invention includes identifying a clock frequency requirement of a processor and adjusting the voltage supplied to the processor to support the identified clock frequency requirement.").

b. Limitation 8[a]: "providing a plurality of domains in the microprocessor;"

142. In my opinion, Soerensen teaches this limitation. As discussed with respect to Claim element 1[a], Soerensen teaches providing a plurality of domains. §VII.D.1.

c. Limitation 8[b]: “clocking each of the domains separately at a clock frequency”

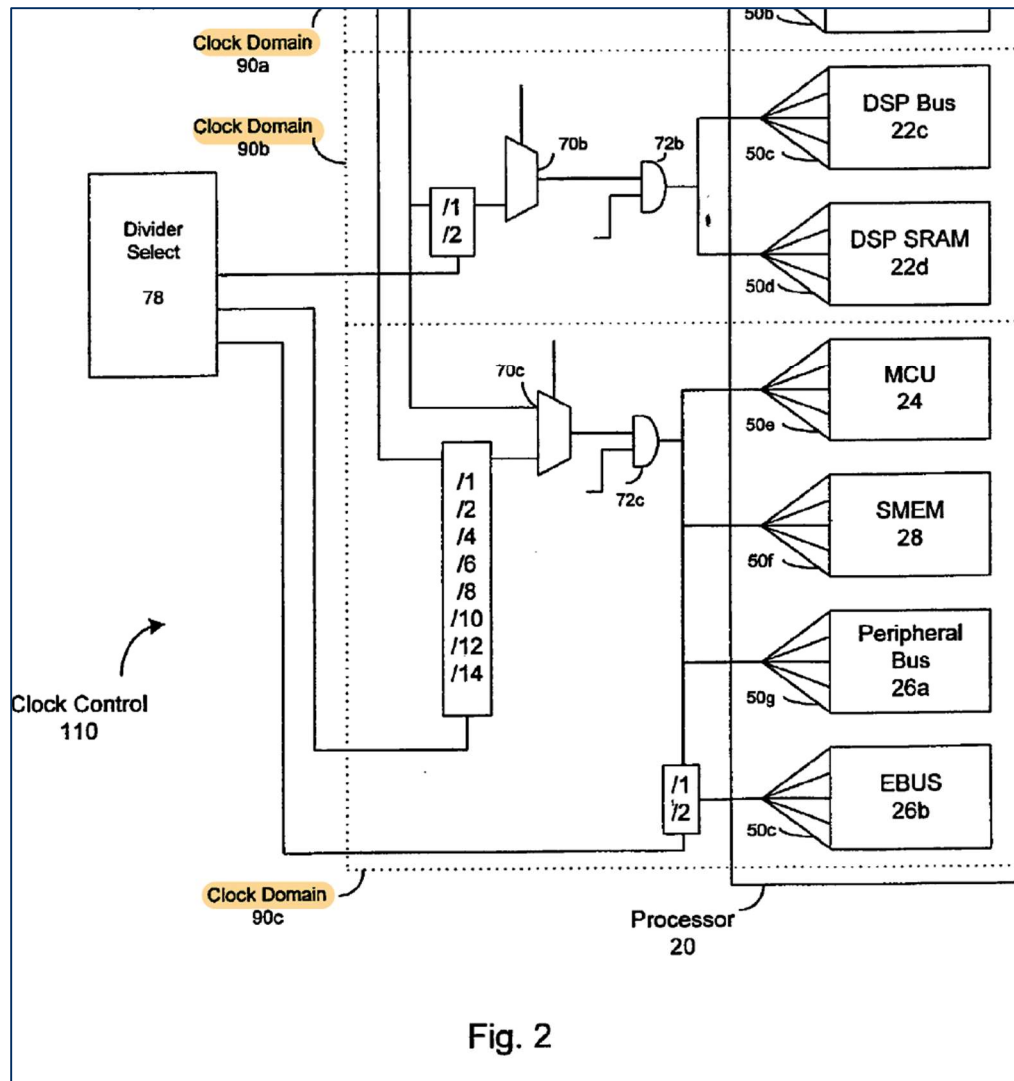
143. In my opinion, Soerensen teaches this limitation. As discussed with respect to Claim element 1[b], Soerensen teaches clocking each of the domains separately. §VII.D.1.

d. Limitation 8[c]: “applying a voltage to each of the domains separately;”

144. In my opinion, Soerensen teaches this limitation. As discussed with respect to Claim element 1[c], Soerensen teaches applying a voltage to each of the domains separately. §VII.D.1.

e. Limitation 8[d]: “operating the microprocessor such that each domain operates synchronously, while the domains operate asynchronously relative to one another; and”

145. In my opinion, Soerensen teaches this limitation. As discussed with respect to Claim element 1[a], Soerensen teaches a plurality of domains. §VII.D.1. Soerensen further teaches that each domain is synchronous, while the domains are asynchronous relative to one another. The dotted lines in Figure 2 outline domains, within which all components are synchronous:



Ex. 1006, Fig 2 (annotated), [0011] (“FIG. 2 illustrates a clocking scheme and control to generate and distribute clock signals to the various components of the processor illustrated in FIG. 1;”), [0042] (“It should be appreciated that the clocking scheme illustrated in FIG. 2 is merely exemplary and is meant to illustrate one scheme by which the various clock signals required by a processor may be controlled and distributed to the components and/or subsystems of the processor.”)

146. In my opinion, Soerensen further teaches that each domain is asynchronous relative to the others, because each of multiplexers 70a, 70b, and 70c may generate a different frequency independently. For instance, Soerensen discloses that “clocked components operating in clock domain 90b” may have “a clock signal having half the frequency of clock signal 84.” Ex. 1006 at [0038] (“For example, divider 62 may provide the clocked components operating in clock domain 90b with a clock signal having half the frequency of clock signal 84 when the clock signal from the system PLL multiplier is selected. Similarly, divider 64 may provide the clocked components operating in clock domain 90c with a clock signal having a frequency reduced by factors of 2, 4, 6, 8, 10, 12, and 14.”). Critically, there is no requirement that the domains maintain a fixed ratio relative to one another. *Id.* (“Dividers 62 and 64 may be, for instance, divide-by-n counters or equivalent components for reducing the frequency of a clock signal. The various divider factors selected for each clock domain may be selected, for example, by divider select 66.”). Thus, each domain in Soerensen is internally synchronous, and fully asynchronous relative to the others.

f. Limitation 8[e]: “dynamically controlling the clock frequency and the voltage in each of the plurality of domains independently of the clock frequencies and the voltages in others of the plurality of domains.”

147. In my opinion, Soerensen teaches this limitation. As discussed above with respect to Claim elements 1[b] and 1[c], Soerensen teaches the ability to

dynamically adjust both frequency and voltage in each domain independently.

§VII.D.1. Soerensen further discloses the method of operating the processor in this manner. Soerensen's disclosures are directed to "a method of dynamic power control." Ex. 1006 at [0006] ("One embodiment according to the present invention includes a method of dynamic power control in a system having at least one processor, the at least one processor including at least one clock signal capable of operating at a plurality of frequencies."). In line with this inventive concept, Soerensen's Figure 5 "illustrates a method of providing dynamic power control to a processor according to one embodiment of the present invention." *Id.* at [0014], Fig. 5.

6. Claim 10: "The method of claim 8, wherein step (d) comprises providing a queue for communication between at least two of the domains."

148. In my opinion, Soerensen renders obvious this limitation. As discussed with respect to Claim 5, Soerensen renders obvious the method of providing a queue for inter-domain communication. §VII.D.4.

7. Claim 12: "The microprocessor of claim 8, wherein there are at least four of said domains."

149. In my opinion, as discussed with respect to Claim 3, Soerensen teaches the method of operating the microprocessor wherein there are at least four domains. §VII.D.2.

E. Ground III.B: Claims 1, 3, 4, 5, 8, 10, and 12 Are Rendered Obvious By Soerensen in view of Hemani

1. Motivation to Combine Soerensen and Hemani

150. A POSITA would be motivated to combine the teachings of Soerensen and Hemani because they are complementary disclosures focused on the same issue. For instance, Hemani is titled *Lowering Power Consumption in Clock by Using Globally Asynchronous Locally Synchronous Design Style*, and Soerensen’s “invention relates to reducing power consumption in processors” by “determining the clock frequency requirement.” Ex. 1007 at Cover; Ex. 1006 at [0002] (“This invention relates to reducing the power consumption in processors and, more particularly, to methods and apparatus for dynamically controlling the voltage level supplied to a processor based on the processing state and/or computational demands of the processor by determining the clock frequency requirement and a corresponding voltage requirement of the processor.”). Both teachings disclose dividing a processor into distinct clock regions which are locally synchronous but asynchronous to one another. Soerensen, like the ‘443 Patent, refers to these blocks as “clock domains” while Hemani refers to these regions as “synchronous blocks.” Ex. 1006 at [0037] (“For example, clock control 110 includes multiplexers in each of clock domains (e.g., multiplexers 70a, 70b, and 70c, respectively) which may independently select whether clock signal 84 is required in order to operate the components in the corresponding clock domain.”); Ex. 1007 at §1 (“GALS

architecture is composed of large synchronous blocks (SBs) which communicate with each other on an asynchronous basis.”). The core means of achieving their respective inventions are similarly uniform. *Compare* Ex. 1006 at [0038] (“Additionally, clock control 110 may divide the frequency of clock signal 84 in order to provide the components of the processor with clocks signals having reduced frequencies in order to meet the clock frequency requirement of the processor.”), *with* Ex. 1007 at §1 (“Additionally, as SBs operate asynchronously with respect to each other, the frequency at which each SB is clocked can be tailored to the local needs, thus reducing the average frequency and the overall power consumption.”).

151. Because Hemani provides a broad overview of means by which a GALS processor may be divided into multiple clock domains, and Soerensen provides a detailed description of the dynamic control of such a processor, a POSITA would be motivated to combine these teachings in developing a dynamically scalable GALS processor. And because both teachings relate to lowering power consumption in globally asynchronous, locally synchronous microprocessors, a POSITA would reasonably expect to be successful in integrating the teachings of each reference.

2. Claim 1

a. Limitation 1[pre]: “A multiple clock domain microprocessor comprising:”

152. In my opinion, to the extent the preamble is limiting, Soerensen and Hemani teach this limitation or render it obvious. As noted above, Soerensen teaches

a processor, and further teaches that “Processor 20 may, for instance, be fabricated on a single semiconductor chip,” *i.e.*, a microprocessor. Ex. 1006 at [0028]; *id.* (“DSP and MCU”). To the extent that Soerensen does not explicitly disclose the “microprocessor” claim language, Hemani discloses that its teachings are directed to “high performance microprocessors.” Ex. 1007 at §1 (“VLSI systems have reached a point in their evolution where they are big enough and are being clocked at high enough frequency that the overhead of the clock in the form of power consumption has become unacceptable. This is confirmed by [12] for high performance microprocessors, see Fig. 1.”). It would have been obvious to a POSITA that Soerensen’s systems would have been implemented on a “microprocessor” such as that taught by Hemani, because Hemani is explicit regarding the mechanical aspects to which GALS systems such as Soerensen were intended to be applied.

153. Further, Soerensen teaches multiple clock domains. *See, e.g.*, Ex. 1006 at [0037] (“The clock signals in each clock domain are provided by various select, divider, and gating logic and distributed to the components of the processor via a plurality of clock trees 50a-50h.”); *id.* (“For example, clock control 110 includes multiplexers in each of clock domains (e.g., multiplexers 70a, 70b, and 70c, respectively) which may independently select whether clock signal 84 is required in order to operate the components in the corresponding clock domain.”). Similarly,

Hemani—which refers to clock domains as *synchronous blocks*—teaches multiple “synchronous blocks” with “independent local clock generators.” Ex. 1007 at §1 (“GALS architecture is composed of large synchronous blocks (SBs) which communicate with each other on an asynchronous basis.... The GALS approach has its own overheads and demands on design methodology. The first overhead concerns the handshake protocol which introduces global control signals, this is discussed in more detail in section 3.3. The second overhead is for the cost of local clock generation for which two schemes exist.”).

b. Limitation 1[a]: “a plurality of domains;”

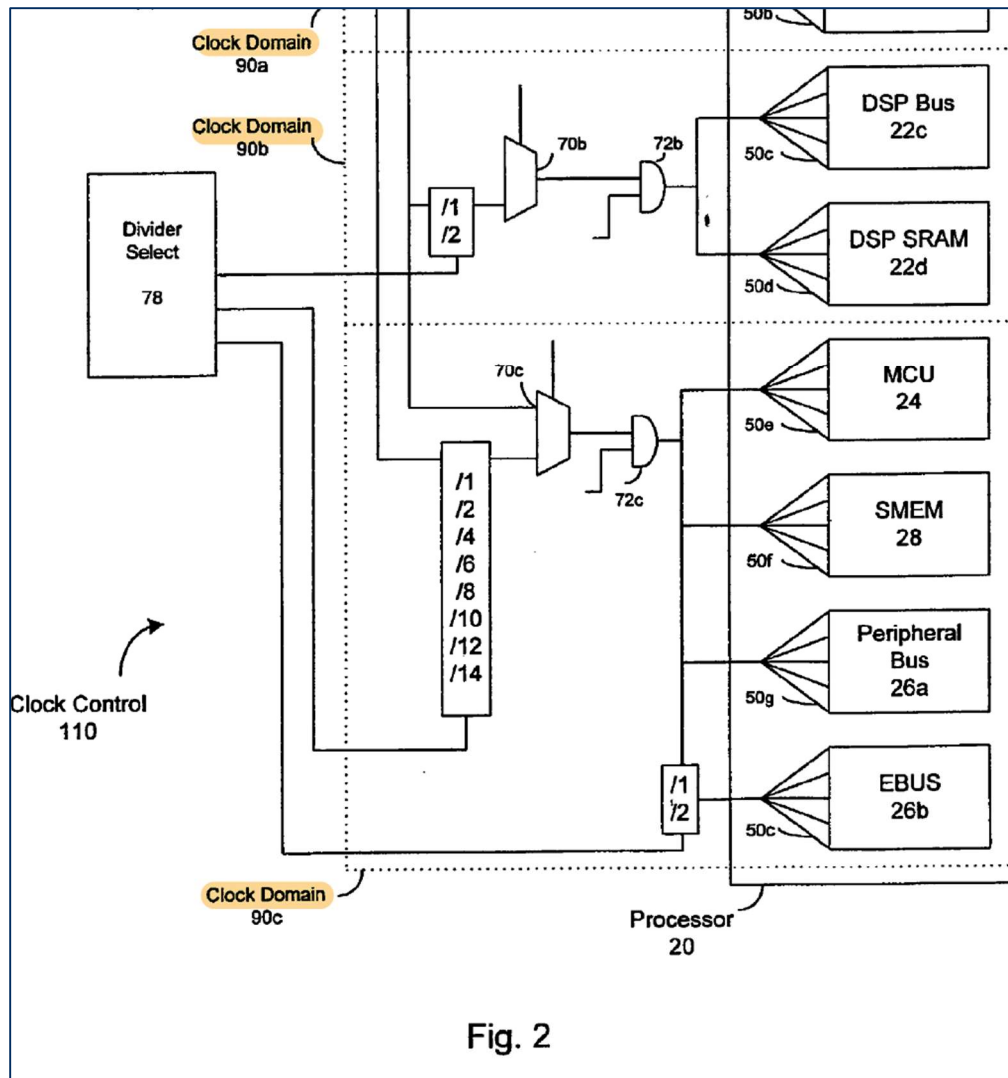
154. In my opinion, Soerensen and Hemani teach this limitation or render it obvious. Soerensen’s Claim 6, for instance, claims the following:

The method of claim 1, wherein the at least one processor has a plurality of clock domains, each clock domain providing a clock signal at a plurality of frequencies to a plurality of components of the at least one processor; and

wherein the act of determining the clock frequency requirement includes determining the minimum clock frequency required in each of the plurality of clock domains.

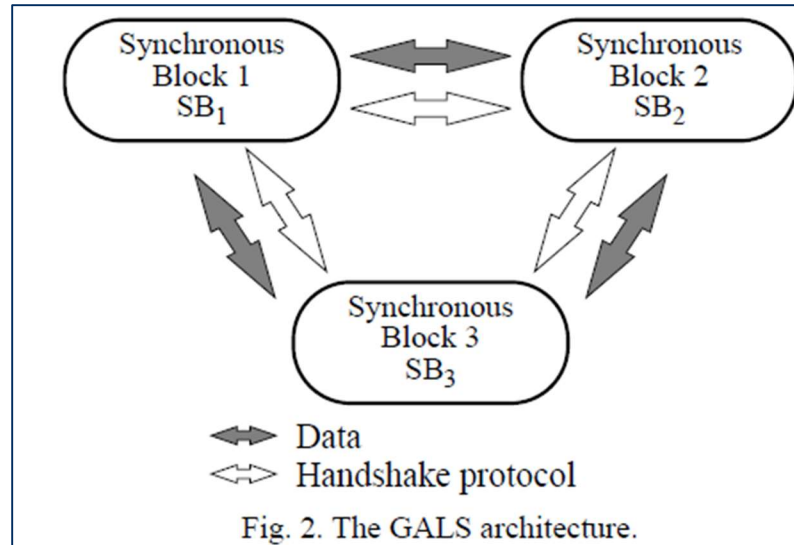
Ex. 1006 at 8, Cl. 6.

155. And Soerensen’s Figure 2 shows this plurality of clock domains:



Ex. 1006, Fig. 2 (cropped and annotated). (“FIG. 2 illustrates a clocking scheme and control to generate and distribute clock signals to the various components of the processor illustrated in FIG. 1.”).

156. Hemani’s Figure 2 similarly shows a plurality of domains in a GALS system, which it again refers to as “synchronous blocks” (SBs):



Ex. 1007, Fig. 2, §1 (“By eliminating the global clock, we eliminate a major source of power consumption and a design bottleneck. Additionally, as SBs operate asynchronously with respect to each other, the frequency at which each SB is clocked can be tailored to the local needs, thus reducing the average frequency and the overall power consumption.”). These complementary teachings confirm the compatibility of these references in combination.

c. Limitation 1[b]

- i. **“for each of the plurality of domains, a clock for separately generating a clock signal at a frequency for that domain ...”**

157. In my opinion, Soerensen and Hemani teach this limitation or render it obvious. Soerensen teaches, for instance, that the “various frequencies required by the processor are often generated from the system clock by a phase locked loop (PLL) multiplier.” Ex. 1006 at [0033] (“The various clock frequencies required by

the processor are often generated from the system clock by a phase locked loop (PLL) multiplier, such that the clock signals distributed to the components of the processor are in-phase with the common reference clock.”). Soerensen notes that this means is “merely exemplary and is meant to illustrate one scheme by which the various clock signals required by a processor may be controlled and distributed to the components and/or subsystems of the processor.” *Id.* Hemani discloses this method, as well as alternatives, teaching “there are two possibilities, either to have an independent local clock generator or to use a global reference clock signal with frequency multiplier in SBs to achieve the desired higher frequency.” Ex. 1007 at §3.4. In the former instance, Hemani teaches “[l]ocal clock generators based on ring oscillators.” Ex. 1007 at §3.4 (“Local clock generators based on ring oscillators have many advantages such as robustness, small size and low power consumption.”). In sum, Soerensen and Hemani disclose multiple means for using a clock to separately generate frequencies for each domain. As Hemani observes, a POSITA would be aware of—and free to choose between—the various disclosed alternatives. Ex. 1007 at §3.4.

- ii. **“... the frequency being dynamically changeable independently of the frequencies of the clock signals generated for others of the plurality of domains; and”**

158. In my opinion, Soerensen and Hemani teach this limitation or render it obvious. As noted above, Soerensen discloses “dynamically adjust[ing] the

frequency of the clock signals provided” Ex. 1006 at [0023] (“Accordingly, one way to reduce the power consumption of a device is to dynamically adjust the frequency of the clock signals provided to a processor to reflect the clock frequency requirement of the processor.”). Soerensen further discloses that this scaling is executed independently for each domain. Ex. 1006 at [0081] (“At some later instant, the MCU may finish its task and require only a 45 MHz clock in order to meet the computational demands of the current processing state. Software running on the MCU, for instance, may identify the new processing state and write the new frequency value into the appropriate register.”).

159. Hemani, too, teaches “independent local clock generators” based on ring oscillators, and the ability to dynamically “manipulate the frequency of the ring oscillator[s].” Ex. 1007 at §3.4 (“There are many ways to manipulate the frequency of the ring oscillator.”). This dynamic scaling achieves the same purpose—it “allows SBs to run at different clock speeds” and thus affects “power consumption and savings.” Ex. 1007 at §5 (“This paper was concerned mainly with power consumption and savings, but there are further consequences of the GALS architecture that deserve closer analysis. The GALS architecture allows SBs to run at different clock speeds.”). Thus, each reference discloses local frequencies which are independently and dynamically changeable.

- d. Limitation 1[c]: “for each of the plurality of domains, a voltage input for receiving a voltage which is**

dynamically changeable independently of the voltages applied to said others of the plurality of domains.”

160. In my opinion, Soerensen and Hemani teach this limitation or render it obvious. As explained above, §II.A.1, Soerensen teaches that voltage can be “dynamically controlled.” Ex. 1006 at [0048] (“Applicants have identified and appreciated that by monitoring the clock frequency requirement of a processor, the voltage supplied to the processor can be dynamically controlled such that the processor spends less time supplying a voltage to the processor in excess of the requirement.”). For instance, Soerensen—which teaches individual and independent frequencies in each domain—notes that “to avoid such potential problems during power state transitions, when going to a higher power state, the voltage may be stepped up before the clock frequency is increased. When going to a lower power state, frequency may be first decreased before reducing the voltage supplied to the processor.” Ex. 1006 at [0064].

161. To the extent that Soerensen does not explicitly disclose this limitation, Hemani does. As a POSITA would understand, the purpose of having independent and dynamically variable clock domains is to enable power savings by reducing the voltage in each domain whenever able, without sacrificing overall performance. For this reason, a POSITA developing a GALS architecture would be motivated to reduce the voltage in each domain whenever the frequency is reduced in that domain. Recall that Hemani discloses that “the frequency at which each SB is clocked can be

tailored to the local needs, thus reducing . . . overall power consumption” and that “[p]ower consumption in VLSI systems is reduced by reducing voltage.” Ex. 1007 at §1 (“Power consumption in VLSI systems is reduced by reducing voltage, switched capacitance and switching activity.... Additionally, as SBs operate asynchronously with respect to each other, the frequency at which each SB is clocked can be tailored to the local needs, thus reducing the average frequency and the overall power consumption.”). To this end, Hemani expressly teaches “dynamic voltage scaling of the power supply voltage at each SB containing a ring oscillator” in order to achieve an “energy efficient design.” Ex. 1007 at §3.4 (“In [11], dynamic voltage scaling of both the oscillators and the SBs, leading to an energy efficient design is proposed. Using a DC/DC-converter, dynamic voltage scaling of the power supply voltage at each SB containing a ring oscillator is made.”).

3. Claim 3: “The microprocessor of claim 1, wherein there are at least four of said domains.”

162. In my opinion, Soerensen and Hemani teach this limitation or render it obvious. To the extent that designing the microprocessor of claim 1 with four domains would not already be obvious to a POSITA, Soerensen expressly discloses such an embodiment. Ex. 1006 at [0043] (“Plot 300 shows exemplary frequency values of four clock domains of a system clocking scheme of a baseband processor as they might vary in time.”). Hemani, for its part, discloses up to 52 domains, which it refers to as synchronous blocks (SBs):

Table 1: Design Data			
	Gate Count in Millions	Die Size (mm)	Number of Initial SBs
Design 1	1.125	7.5 X 7.5	24
Design 2	2.09	10.9 X 10.9	46
Design 3	3.04	12.8 X 12.8	52

Ex. 1007, Table 1, §4 (“The description was cut at the first level of hierarchy, resulting in 24, 46 and 52 partitions for the three designs respectively. These partitions were treated as the initial set of SBs for the three designs.”).

163. In sum, there is nothing inventive or non-obvious about using four (or more) domains.

4. Claim 4

- a. Limitation 4[a]: “The microprocessor of claim 1, wherein the microprocessor is programmed to determine a slack in processing in one of the domains and . . .”**

164. In my opinion, Soerensen and Hemani teach this limitation or render it obvious. As discussed above, Soerensen and Hemani render obvious Claim 1. §VII.E.1. Soerensen notes with respect to slack that a “processor is considered to be consuming excess power when the computational tasks, functions, and operations of the processor may be performed, and/or the processing demands met, at a lower power state.” Ex. 1006 at [0017]. Soerensen’s use of the term “processor” is intended to encompass “any apparatus that performs logic operations, computational tasks,

control functions, etc.” Ex. 1006 at [0017] (“The term “processor” refers, generally, to any apparatus that performs logic operations, computational tasks, control functions, etc. A processor may include one or more subsystems, components, and/or other processors. A processor will typically include various logic components that operate using a clock signal to latch data, advance and/or sequence logic states, synchronize computations and logic operations, and/or provide other timing functions.”). A POSITA would have understood this to include, for instance, domains comprising an instruction unit or execution unit. As discussed above with respect to element 1[b], Soerensen discloses a plurality of clock domains with independent frequencies. §VII.E.1. Soerensen further discloses a “controller adapted to determine a voltage requirement based on the clock frequency requirement.” Ex. 1006 at [0007] (“The processor comprises one or more clocked components receiving the at least one clock signal, and a controller coupled to at least one of the components to determine a clock frequency requirement of the processor, the controller adapted to determine a voltage requirement based on the clock frequency requirement of the processor and to sequence a transition to a power state defined by the clock frequency requirement and the voltage requirement.”). In other words, Soerensen teaches monitoring and adapting to slack in a domain based on the “delta” between the existing frequency and voltage and the frequency and voltage “required.” *Id.* (“Another embodiment according to the present invention includes a

processor including at least one clock signal operating at a plurality of frequencies.”); *id.* at [0061] (“In step 240, the current voltage level being supplied to the processor is compared to the voltage requirement determined in step 240 to determine the required voltage change, or delta, required to provide the voltage requirement.”). As Soerensen further explains, “the clock frequency requirement may represent the minimum frequency that each clock domain requires in order to perform the functions and/or computational tasks of the components operating in that clock domain.” *Id.* [0059].

b. Limitation 4[b]: “... to reduce the clock frequency and the voltage in said one of the domains to reduce the slack.”

165. In my opinion, Soerensen and Hemani teach this limitation or render it obvious. As Soerensen notes preliminarily, its invention is directed to reducing power consumption in processors. Ex. 1006 at [0002] (“This invention relates to reducing the power consumption in processors and, more particularly, to methods and apparatus for dynamically controlling the voltage level supplied to a processor based on the processing state and/or computational demands of the processor by determining the clock frequency requirement and a corresponding voltage requirement of the processor.”). And as discussed with respect to Claim element 4[a], Soerensen teaches monitoring slack and reducing frequency and voltage in response to slack. A POSITA would recognize that the ultimate point of reducing

frequency in response to slack is to ultimately reduce voltage and thus reduce power consumption. Ex. 1006 at [0017] (“In many devices, particularly battery powered devices, it is advantageous to reduce the amount of time in which the device’s processor is consuming excess power.”). Unsurprisingly, Soerensen teaches that “by monitoring the clock frequency requirement of a processor, the voltage supplied to the processor can be dynamically controlled such that the processor spends less time supplying a voltage to the processor in excess of the requirement.” Ex. 1006 at [0025]; *see also id.* at [0041] (“As such, by recognizing the clock frequency requirement of the processor, clock control 100 may configure the frequency distribution to reduce the intervals when excess clock frequencies are being generated.”).

166. As noted throughout, the purpose of monitoring slack is to reduce frequency—and consequentially voltage—whenever able, and thus reduce power consumption. Hemani, which is focused on “lowering power consumption” teaches “dynamic voltage scaling of the power supply voltage at each SB.” Ex. 1007 at §3.4 (“Using a DC/DC-converter, dynamic voltage scaling of the power supply voltage at each SB containing a ring oscillator is made.”). As would be clear to a POSITA, voltage in a domain can only be reduced in response to slack. Reducing voltage without slack in the frequency would impermissibly degrade the clock signal. As Soerensen notes, several “methods of determining a voltage requirement from a

clock frequency requirement of the processor will occur to those skilled in the art and may depend on the clocking scheme of a particular processor implementation.” Ex. 1006 at [0060]. But in any event, “power consumption of a device . . . proportional to the frequency at which the processor operates” and thus Hemani implicitly or expressly teaches monitoring and responding to slack. Ex. 1006 at [0018] (“Applicant has observed that the power consumption of a device is related to the square of the voltage supplied to the device’s processor and proportional to the frequency at which the processor operates.”); *see also* Ex. 1007 at §3.4 (“Local clock generators based on ring oscillators have many advantages such as robustness, small size and low power consumption. The basic ring oscillator consists of an odd number of inverters connected in a circular chain.”).

5. Claim 5: “The microprocessor of claim 1, further comprising a queue for communication between at least two of the domains.”

167. In my opinion, Soerensen and Hemani teach this limitation or render it obvious. As discussed above, Soerensen and Hemani teach or render obvious Claim 1. §VII.E.1. And as discussed with respect to Claim element 1[pre], Soerensen discloses multiple asynchronous clock domains necessitating interdomain data transfer. *See also* Ex. 1006 at [0028] (“The various subsystems of the processor are connected to a system bus 30 in order to communicate and/or transfer data between the various components, such as, operands or instructions, addresses and/or control

signals, etc.”). A POSITA would have understood that said system bus requires a mechanism for coherent transfer between domains, including a queue. It would have been obvious for a POSITA to implement a queue for interdomain transfer because queues were well known in the art and Soerensen does not teach toward an incompatible alternative. The ‘443 Patent itself acknowledges that this implementation would be obvious to a POSITA, noting that its own purported invention simply coopts existing queue designs. Ex. 1001 at 5:1 (“Our design uses existing queue structures in a superscalar processor core to isolate the different clock domains in a way that minimizes the need for inter-domain synchronization.”). Thus, a POSITA would find it obvious and be motivated to implement a queue in Soerensen’s design.

168. To the extent that Soerensen does not expressly disclose this claim language, Hemani teaches this element, and for the reasons noted above, a POSITA would be motivated to employ Hemani’s queue structure. Specifically, Hemani describes multiple interdomain communication methods, including a “FIFO.” Ex. 1007 at §2 (“In the communication refinement step, for every pair of communicating SBs, communications requirement analysis is made to select the cheapest of the four possible communication modes: send and forget, strobe, handshake or FIFO.”). Each communication mode has its own unique advantages and disadvantages associated with its defined operation. For instance, send and forget mode does not require an

acknowledgement signal from the receiver, thereby minimizing the amount of time needed to wait for that signal. Therefore, it has low overhead and latency but is less reliable, since it does not guarantee that the receiver has received the information. This type of communication mode is better suited for certain real-time applications where latency is more important than the guaranteed delivery and data integrity. A strobe based communication mode uses a strobe signal to coordinate the transmission between sender and receiver nodes. It is a low overhead approach that provides simple hardware implementation but may suffer from signal skew when it has to travel on long wires. When a FIFO buffer is incorporated as part of a communication mode, the sender and receiver utilize the buffer for transferring data by writing to and reading from the FIFO buffer, respectively. As discussed with respect to Iyer and Liljeberg, a FIFO is a “first in first out” queue. Of the “existing queue structures” acknowledged by the ‘443 specification, Ex. 1001 at 5:1 (“Our design uses existing queue structures in a superscalar processor core to isolate the different clock domains in a way that minimizes the need for inter-domain synchronization.”), this queue was the most well-known, and thus a POSITA would have been motivated to adapt it and would reasonably expect to be successful in its implementation. As further motivation, Hemani encourages the selection of the most energy efficient—or “cheapest”—structure, and thus where a FIFO is most energy efficiency, a POSITA would be especially motivated to employ such a queue. Ex.

1007 at §2 (“In the communication refinement step, for every pair of communicating SBs, communications requirement analysis is made to select the cheapest of the four possible communication modes: send and forget, strobe, handshake or FIFO.”).

6. Claim 8

a. Limitation 8[pre]: “A method of operating a microprocessor, the method comprising:”

169. In my opinion, to the extent the preamble is limiting, as discussed with respect to Claim element 1[pre], Soerensen and Hemani teach a microprocessor. §VII.E.1. Soerensen further teaches that its invention relates “to methods” for dynamically controlling said microprocessor. *See, e.g.*, Ex. 1006 at [0002] (“This invention relates to reducing the power consumption in processors and, more particularly, to methods and apparatus for dynamically controlling the voltage level supplied to a processor based on the processing state and/or computational demands of the processor by determining the clock frequency requirement and a corresponding voltage requirement of the processor.”); *id.* at [0027] (“Accordingly, one embodiment according to the present invention includes identifying a clock frequency requirement of a processor and adjusting the voltage supplied to the processor to support the identified clock frequency requirement.”). Hemani similarly teaches “suggested methods” of operation. *See* Ex. 1007 at §3.4 (“There are advantages of reducing the local oscillator frequency, using one of the suggested methods, instead of letting each SB operate fast and then be turned off when ready.”).

b. Limitation 8[a]: “providing a plurality of domains in the microprocessor;”

170. In my opinion, Soerensen and Hemani teach this limitation or render it obvious. As discussed with respect to Claim element 1[a], Soerensen and Hemani teach or render obvious providing a plurality of domains. §VII.E.1.

c. Limitation 8[b]: “clocking each of the domains separately at a clock frequency”

171. In my opinion, Soerensen and Hemani teach this limitation or render it obvious. As discussed with respect to Claim element 1[b], Soerensen and Hemani teach or render obvious clocking each of the domains separately. §VII.E.1.

d. Limitation 8[c]: “applying a voltage to each of the domains separately;”

172. In my opinion, Soerensen and Hemani teach this limitation or render it obvious. As discussed with respect to Claim element 1[c], Soerensen and Hemani teach or render obvious applying a voltage to each of the domains separately. §VII.E.1.

e. Limitation 8[d]: “operating the microprocessor such that each domain operates synchronously, while the domains operate asynchronously relative to one another; and”

173. In my opinion, Soerensen and Hemani teach this limitation or render it obvious. For instance, Soerensen’s Figure 2 outlines domains, within which all components are synchronous:

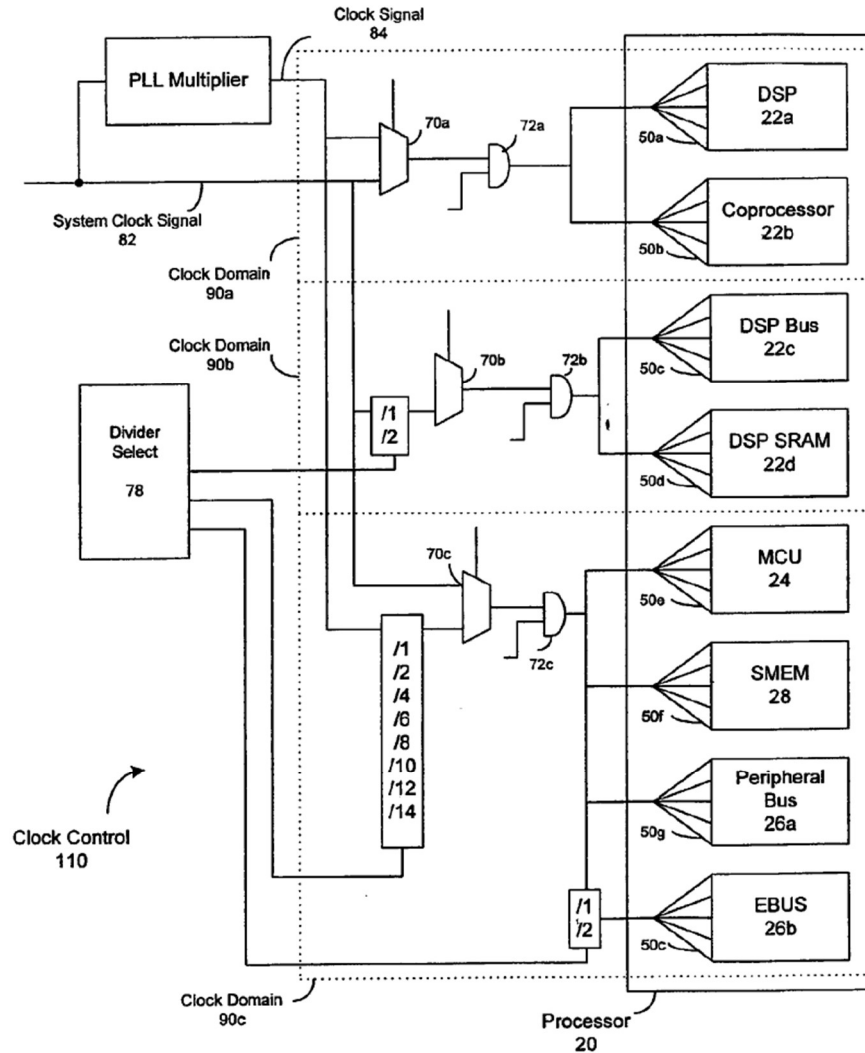


Fig. 2

Ex. 1006, Fig 2, ¶11 (“FIG. 2 illustrates a clocking scheme and control to generate and distribute clock signals to the various components of the processor illustrated in FIG. 1.”).

174. Soerensen further teaches that each domain is asynchronous relative to the others, because each of multiplexers 70a, 70b, and 70c may generate a different frequency independently. *Id.* at [0037] (“For example, clock control 110 includes

multiplexers in each of clock domains (e.g., multiplexers 70a, 70b, and 70c, respectively) which may independently select whether clock signal 84 is required in order to operate the components in the corresponding clock domain.”).

175. Hemani is even more explicit, disclosing a “GALS architecture [] composed of large synchronous blocks (SBs) which communicate with each other on an asynchronous basis.” Ex. 1007 at §3.4 (“GALS architecture is composed of large synchronous blocks (SBs) which communicate with each other on an asynchronous basis. By eliminating the global clock, we eliminate a major source of power consumption and a design bottleneck.”).

f. Limitation 8[e]: “dynamically controlling the clock frequency and the voltage in each of the plurality of domains independently of the clock frequencies and the voltages in others of the plurality of domains.”

176. In my opinion, Soerensen and Hemani teach this limitation or render it obvious. As discussed above with respect to Claim elements 1[b] and 1[c], Soerensen and Hemani teach or render obvious the ability to dynamically adjust both frequency and voltage in each domain independently. §VII.E.1. Soerensen further discloses the method of operating the processor in this manner. Soerensen’s disclosures are directed to “a method of dynamic power control.” Ex. 1006 at [0006] (“One embodiment according to the present invention includes a method of dynamic power control in a system having at least one processor, the at least one processor including at least one clock signal capable of operating at a plurality of

frequencies.”). And Hemani “propose[s] methods” for operating and evaluating their proposed GALS architecture. Ex. 1007 at Abstract (“We propose methods to a) evaluate the benefits of GALS and account for its overheads, which can be used as the basis for partitioning the system into optimal number/size of synchronous blocks, and b) automate the synthesis of the global asynchronous communication.”).

7. Claim 10: “The method of claim 8, wherein step (d) comprises providing a queue for communication between at least two of the domains.”

177. In my opinion, Soerensen and Hemani teach this limitation or render it obvious. As discussed with respect to Claim 5, Soerensen and Hemani teach or render obvious the method of providing a queue for inter-domain communication. §VII.E.5.

8. Claim 12: “The microprocessor of claim 8, wherein there are at least four of said domains.”

178. In my opinion, As discussed with respect to Claim 3, Soerensen and Hemani teach or render obvious the method of operating the microprocessor wherein there are at least four domains. §VII.E.3.

F. Ground III.C: Claims 5 and 10 Are Rendered Obvious By Soerensen (Ex. 1006) in view of Moore (Ex. 1008)

1. Motivation to Combine Soerensen and Moore

179. To the extent that Soerensen alone does not render the subject matter recited by claims 5 and 10 obvious, it is also taught by Moore. As noted above and by the ‘443 Patent itself, queues were well known in the art, and the perhaps most

common form of queue is a first-in-first-out (FIFO) queue. Ex. 1001 at 5:2 (“Our design uses existing queue structures in a superscalar processor core to isolate the different clock domains in a way that minimizes the need for inter-domain synchronization.”).

180. Moore expressly discloses a FIFO queue for communication between synchronous domains which are asynchronous relative to one another. Ex. 1008 at §5 (“Adding an asynchronous FIFO between the producer and consumer (see Figure 5) decouples the req-->sync_ack path. This allows the producer to reliably send just one data item every second clock cycle. For the producer to be able to transmit a data item every clock cycle we have to make the following assumption: if the first element of the FIFO is empty then any data emitted by the synchronous producer will be safely stored before the next clock cycle. This is a very safe timing assumption. From this assumption it is clear that in place of the ack signal from the FIFO we require a *first element of FIFO empty* signal which we will call consumed.”).

181. A POSITA would have been motivated to use Moore’s teachings in combination with Soerensen because Soerensen does not expressly outline a preferred mechanism for enabling asynchronous interdomain communication, and Moore’s disclosures are focused on this very issue. As a POSITA would understand, as a matter of flow control, implementing buffers is advantageous in minimizing

latency and optimizing throughput. A queue allows the producer to continue sending data without waiting for the consumer to be immediately ready, thus reducing latency. Buffers also help manage bursty traffic by absorbing spikes in data generation and decoupling the producer and consumer, allowing each to operate at their own optimal speeds. Therefore, a POSITA would be motivated to employ a queue whenever there are multiple asynchronous domains. Moore itself is principally concerned with “communication between independent clock domains” in “globally asynchronous locally synchronous (GALS) system[s].” *See* Ex. 1008, Abstract (“Reliable low-latency channel communication between independent clock domains may be achieved using a combination of clock pausing techniques, self-calibrating delay lines and an asynchronous interconnect. Such a scheme could be used for point-to-point communication in a globally asynchronous locally synchronous (GALS) system.”). Because Moore is specifically focused on interdomain communication in a GALS design, a POSITA would have reasonably expected to be successful in implementing Moore’s queue in Soerensen’s design.

2. Claim 5: “The microprocessor of claim 1, further comprising a queue for communication between at least two of the domains.”

182. In my opinion, Soerensen in view of Moore renders obvious this limitation. As discussed above, Soerensen renders obvious Claim 1. §VII.D.1. And as discussed with respect to Claim element 1[pre], Soerensen discloses multiple

asynchronous clock domains necessitating interdomain data transfer. *See also* Ex. 1006 at [0028] (“The various subsystems of the processor are connected to a system bus 30 in order to communicate and/or transfer data between the various components, such as, operands or instructions, addresses and/or control signals, etc.”). A POSITA would have understood that said system bus requires a mechanism for coherent transfer between domains, including a queue. It would have been obvious for a POSITA to implement a queue for interdomain transfer because queues were well known in the art and Soerensen does not teach toward an incompatible alternative.

183. Moore describes in detail the implementation of a FIFO queue in a GALS system, noting that “[a]dding an asynchronous FIFO between the producer and consumer (see Figure 5) decouples the [signal] path.” Ex. 1008 at §5 (“Adding an asynchronous FIFO between the producer and consumer (see Figure 5) decouples the req-->sync_ack path. This allows the producer to reliably send just one data item every second clock cycle. For the producer to be able to transmit a data item every clock cycle we have to make the following assumption: if the first element of the FIFO is empty then any data emitted by the synchronous producer will be safely stored before the next clock cycle. This is a very safe timing assumption. From this assumption it is clear that in place of the ack signal from the FIFO we require a first

element of FIFO empty signal which we will call consumed.”). Figure 5 itself is similarly explicit in teaching a queue between clock domains:

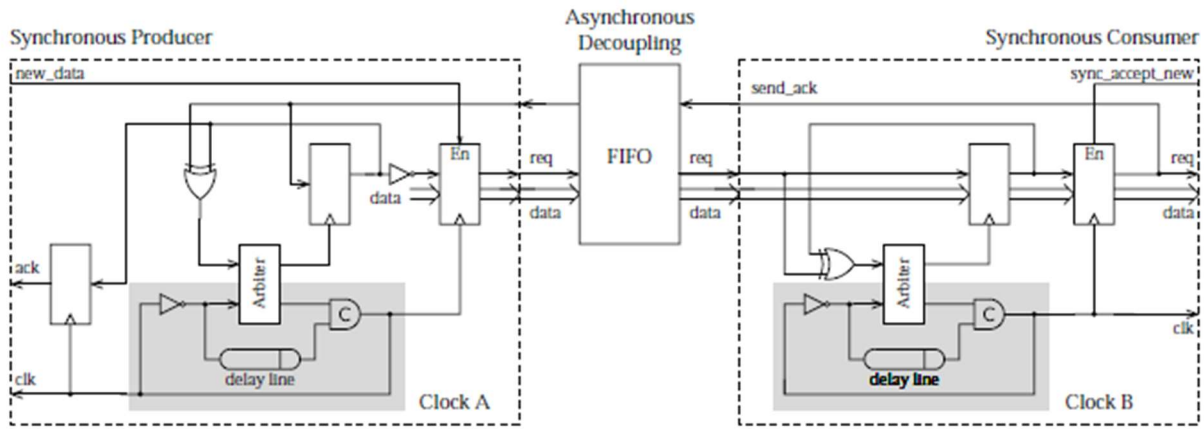


Figure 5: Channel communication with FIFO buffering

Ex. 1008, Fig. 5.

184. Moore’s Figure 5 shows a channel communication system with FIFO buffering, where the FIFO queue is placed between the synchronous producer and the synchronous consumer, effectively decoupling the Request (“req”) and Acknowledge (“ack”) paths and facilitating asynchronous communication between different clock domains. This queue enables interdomain transfer between components operating at different clock frequencies without loss or corruption by temporarily storing data from the producer until the consumer is ready to process it.

185. Therefore, Soerensen in view of Moore renders obvious the microprocessor of Claim 1, further comprising a queue for communication between at least two of the domains.

3. Claim 10: “The method of claim 8, wherein step (d) comprises providing a queue for communication between at least two of the domains.”

186. In my opinion, Soerensen in view of Moore renders obvious this limitation. As discussed with respect to Claim 5, Soerensen in view of Moore renders obvious the method of providing a queue for inter-domain communication. §VII.F.2.

VIII. OBJECTIVE INDICIA OF NON-OBVIOUSNESS

187. I understand that certain factors called “secondary considerations” or “objective indicia” of non-obviousness may support or rebut an assertion of obviousness of a claim. I understand that such secondary considerations include, among other things, commercial success of the alleged invention, skepticism of those having ordinary skill in the art at the time of the alleged invention, unexpected results of the alleged invention, any long-felt but unsolved need in the art that was satisfied by the alleged invention, the failure of others to make the alleged invention, praise of the alleged invention by those having ordinary skill in the art, and copying of the alleged invention by others in the field. I further understand that there must be a nexus—*i.e.*, a connection—between any such secondary considerations and the purported invention.

188. I am not aware of any evidence of objective indicia with a nexus to the Challenged Claims of the ‘443 Patent that would support a finding of non-obviousness or otherwise impact my analysis.

189. I am not aware of any evidence of commercial success of any products due specifically to the alleged invention in the Challenged Claims, as opposed to other factors that may have contributed to any commercial success of those products (if there is any commercial success at all), such as marketing.

190. I am not aware of skepticism by those having ordinary skill in the art at the time of the alleged invention regarding the possibility, utility, or success of the alleged invention of the Challenged Claims, nor am I aware of praise of the alleged invention in the Challenged Claims. Prior to my engagement by Petitioners for purposes of providing my opinion herein, I had never heard of the '443 Patent.

191. I am not aware of any unexpected results achieved by the alleged invention of the Challenged Claims. To the contrary, as explained above, the functional result claimed in the Challenged Claims was a predictable, "straight forward," result of applying standard structures and efficiency steps to microprocessors as shown in the prior art discussed above.

192. I am not aware of any long-felt but unsolved need in the art that was satisfied by the alleged invention of the Challenged Claims.

193. I am not aware of any failure by others to make the alleged invention of the Challenged Claims.

194. I am not aware of any copying of the alleged invention in the Challenged Claims. I understand from counsel that Patent Owner has accused

Petitioners of infringing, but I further understand that Patent Owner has not alleged that Petitioners have specifically copied its patented technology.

IX. CONCLUSION

195. Based on the above, it is my opinion that the Challenged Claims of the '443 Patent are obvious.

196. I hereby declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct, and that all statements made of my own knowledge are true and that all statements made on information and belief are believed to be true. I understand that willful false statements are punishable by fine or imprisonment or both. *See* 18 U.S.C. § 1001.

Date: August 1, 2024

Respectfully submitted,

A handwritten signature in black ink, appearing to read 'N. Bagherzadeh', written in a cursive style.

Nader Bagherzadeh, Ph.D.

APPENDIX OF CHALLENGED CLAIMS

Claim Limitation	Claim Language
1[pre]	A multiple clock domain microprocessor comprising:
1[a]	a plurality of domains;
1[b]i	for each of the plurality of domains, a clock for separately generating a clock signal at a frequency for that domain,
1[b]ii	the frequency being dynamically changeable independently of the frequencies of the clock signals generated for others of the plurality of domains; and
1[c]	for each of the plurality of domains, a voltage input for receiving a voltage which is dynamically changeable independently of the voltages applied to said others of the plurality of domains.
3	The microprocessor of claim 1, wherein there are at least four of said domains.
4[a]	The microprocessor of claim 1, wherein the microprocessor is programmed to determine a slack in processing in one of the domains and
4[b]	to reduce the clock frequency and the voltage in said one of the domains to reduce the slack.
5	The microprocessor of claim 1, further comprising a queue for communication between at least two of the domains.
8[pre]	A method of operating a microprocessor, the method comprising:
8[a]	providing a plurality of domains in the microprocessor;
8[b]	clocking each of the domains separately at a clock frequency;
8[c]	applying a voltage to each of the domains separately;
8[d]	operating the microprocessor such that each domain operates synchronously, while the domains operate asynchronously relative to one another; and
8[e]	dynamically controlling the clock frequency and the voltage in each of the plurality of domains independently of the clock frequencies and the voltages in others of the plurality of domains.

Claim Limitation	Claim Language
10	The method of claim 8, wherein step (d) comprises providing a queue for communication between at least two of the domains.
12	The method of claim 8, wherein there are at least four of said domains.