

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

SILICON MOTION INC.

Petitioner

v.

K. MIZRA LLC

Patent Owner

Case No. IPR2024-01240

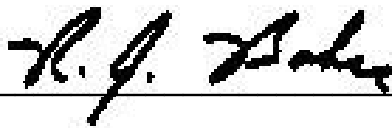
Patent No. 9,160,466

**EXPERT DECLARATION OF R. JACOB BAKER, PH.D., P.E., IN
SUPPORT OF PETITION FOR *INTER PARTES* REVIEW OF
U.S. PATENT NO. 9,160,466**

I declare under penalty of perjury that the following is true and correct. I declare that all statements made herein of my knowledge are true, and that all statements made on information and belief are believed to be true, and that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Executed on August 12, 2024

Boulder, CO

A handwritten signature in black ink, appearing to read "R. J. Baker", is written over a solid black horizontal line.

R. Jacob Baker, Ph.D., P.E.

TABLE OF CONTENTS

I.	Introduction.....	4
II.	Qualifications.....	6
III.	Person Of Ordinary Skill In The Art	14
IV.	Legal Understanding.....	15
V.	The '466 Patent.....	23
VI.	Priority Date of the '466 Patent.....	26
VII.	Claim Construction.....	26
VIII.	State of the Prior Art.....	27
IX.	Ground 1: Meaney in view of the knowledge of a person of ordinary skill in the art	32
X.	Ground 2: Meaney and Nguyen in view of the knowledge of a person of ordinary skill in the art.....	33
XI.	Ground 3: Meaney and Greeff in view of the knowledge of a person of ordinary skill in the art.....	33
XII.	Ground 4: Meaney, Nguyen, and Greeff in view of the knowledge of a person of ordinary skill in the art.....	33
XIII.	Ground 5A: Meaney and Allee in view of the knowledge of a person of ordinary skill in the art.....	33
XIV.	Ground 5B: Meaney, Nguyen, and Allee in view of the knowledge of a person of ordinary skill in the art.....	33
XV.	CONCLUSION.....	34

I. Introduction

1. My name is R. Jacob Baker, Ph.D., P.E. I am a Professor of Electrical and Computer Engineering at the University of Nevada, Las Vegas. I have prepared this declaration as an expert witness on behalf of the Petitioner. I submit this declaration in support of a Petition for *Inter Partes* Review of United States Patent No. 9,160,466 (“the ’466 Patent”), owned by K. Mizra (“K. Mizra” or “Patent Owner”). I have been retained in this matter by Baker Botts LLP (“Counsel”) on behalf of Silicon Motion, Inc. (“SMI” or “Petitioner”).

2. This declaration contains statements of my opinions formed to date and the bases and reasons for those opinions. I may offer additional opinions based on further review of materials in this case, including opinions and/or testimony of other expert witnesses.

3. I make this declaration based upon my personal knowledge. I am over the age of 21 and am competent to make this declaration.

4. The statements herein include my opinions and the bases for those opinions, which relate to at least the following documents of the pending *inter partes* review petition:

1001	U.S. Pat. No. 9,160,466, titled “Periodic calibration for communication channels by drift tracking” (“the ’466 Patent”).
1003	File History for the ’466 Patent

1004	U.S. Pat. Pub. No. 2003/0070123, Meaney <i>et al.</i> , titled “Method and system for optimizing network data transmission using a configurable protocol stack” (“Meaney”).
1005	U.S. Pat. Pub. No. 2003/0151450, Nguyen, <i>et al.</i> , titled “Signaling accommodation” (“Nguyen”).
1006	U.S. Pat. No. 6,356,106, Greeff, <i>et al.</i> , titled “Active termination in a multidrop memory system” (“Greeff”).
1007	U.S. Pat. No. 6,255,979, Allee, <i>et al.</i> , titled “CMOS flash analog to digital converter compensation” (“Allee”).
1010	U.S. Pat. Pub. No. 2002/0066001, Olarig, <i>et al.</i> , titled “Adaptive calibration technique for high speed memory devices,” (“Olarig”).

5. The materials I considered in forming my opinions herein included at least the above-referenced documents. In forming the opinions expressed in this declaration, I relied upon my knowledge, skill, experience, training, and education in the relevant field of the art, and have considered the viewpoint of a person of ordinary skill in the art at the time of the alleged invention (“POSITA”), i.e., a POSITA as of January 28, 2004. My opinion is based on the following references in view of the knowledge of POSITA during this time.:

6. I am being compensated for my time in connection with this IPR at my standard consulting rate for consulting services. My compensation is not affected by the outcome of this matter. Although I am being compensated for my time in preparing this declaration, the opinions herein are my own.

II. Qualifications

7. My qualifications generally are set forth in my Curriculum Vitae, which is attached as Appendix B. Appendix B also includes a list of the publications I have authored and a list of the other cases in which I have testified during the last four years.

A. Educational Background

8. I received a B.S. degree and a M.S. degree in electrical engineering from the University of Nevada, Las Vegas (“UNLV”) in 1986 and 1988, respectively. I received my Ph.D. in Electrical Engineering from the University of Nevada, Reno, in 1993.

9. My doctoral research, culminating in the award of a Ph.D. in Electrical Engineering in 1993, investigated the use of power MOSFETs (metal oxide semiconductor field effect transistors) in the design of very high peak power, and high-speed, instrumentation. I developed techniques to reliably stack power MOSFETs to switch higher voltages, that is, greater than 1,000 V at near 100 Amps of current with nanosecond switching times. This work was reported in the paper entitled “Transformerless Capacitive Coupling of Gate Signals for Series Operation of Power MOSFET Devices,” published in the IEEE Transactions on Power Electronics. The paper received the 2000 Best Paper Award from the Power Electronics Society.

B. Industry Experience

10. I have done technical and expert witness consulting for over 200 companies and their subsidiaries since I started working as an engineer in 1985. From 1985 to 1993, I worked for EG&G Energy Measurements and the Lawrence Livermore National Laboratory designing nuclear diagnostic instrumentation for underground nuclear weapon tests at the Nevada test site. During this time, I designed, and oversaw the fabrication of, over 30 electronic and electro-optic instruments, including high-speed cable and fiber-optic receiver/transmitters, PLLs, frame and bit-syncs, data converters, streak-camera sweep circuits, Pockel's cell drivers, micro-channel plate gating circuits, charging circuits for battery backup of equipment for recording test data, and analog oscilloscope electronics.

11. I am a licensed Professional Engineer and have extensive industry experience in circuit design, fabrication, and manufacture of Dynamic Random Access Memory (DRAM) semiconductor integrated circuit chips, Phase-Change Random Access Memory (PCRAM) chips, and CMOS Image Sensors (CISs) at Micron Technology, Inc. ("MTI") in Boise, Idaho. I am a named inventor on 150 U.S. patents in integrated circuit design including Flash memory, DRAM, and CMOS image sensors. I spent considerable time working on the development of flash memory chips while at MTI. My efforts resulted in more than a dozen patents relating to flash memory. One of my projects at MTI included the development,

design, and testing of circuit design techniques for a multi-level cell (MLC) flash memory using signal processing. This effort resulted in higher-density memories for use in solid-state drives and flash memory cards having an ATA interface that are ubiquitous in consumer electronics, including cameras and data storage systems. Further, the use of higher-density memory can result in fewer changes in the flash translation layer for logical-to-physical addressing, less need for garbage collection, and larger data segments that can improve a computing system's performance. Another project I worked on at MTI focused on the design of buffers for high-speed double-data rate DRAM, which resulted in around 10 U.S. patents in buffer design. Among many other experiences, I led the development of the delay locked loop (DLL) in the late 1990s so that MTI DRAM products could transition to the DDR memory protocol, used in mobile and non-mobile (server, desktop, cell phones, tablets, etc.) computing systems as main computer memory, for addressing and controlling accesses to memory via interprocess communications (IPC) with the memory controller (MC). In addition to work developing these high-speed digital communication systems, I've supervised a half dozen or so master's Theses in high-speed digital communications with focuses ranging from calibration to eliminating slack in digital timing. Further, I provided technical assistance with MTI's acquisition of Photobit during 2001 and 2002, including transitioning the manufacture of CIS products into MTI's process

technology. Further, I did consulting work at Sun Microsystems and then Oracle on the design of memory modules during 2009 and 2010. This work entailed the design of low-power, high-speed communications, and wide interconnection methods with the goal of transmitting data to/from the memory module and the MC at higher speeds.

12. I have extensive experience in the development of instrumentation and commercial products in a multitude of areas including: integrated electrical/biological circuits and systems, array (memory, imagers, and displays) circuit design, CMOS analog and digital circuit design, diagnostic electrical and electro-optic instrumentation for scientific research, CAD tool development and online tutorials, low-power interconnect and packaging techniques, design of communication/interface circuits (to meet commercial standards such as USB, firewire, DDR, PCIe, SPI, etc.), circuit design for the use and storage of renewable energy, and power electronics. For example, a part of my research at Boise State, for many years, focused on the use of Thru-Silicon-Vias (TSVs), aka Thru-Wafer Vias (TWVs), for high-density 3D packaging. These packaging techniques were utilized in the memory module development work I did with Sun Microsystems and Oracle. As another example, I designed circuitry for use in implementing Universal Serial Bus (USB) interface circuits while I did consulting at Tower Semiconductor. I designed PCI communication circuits for IPC between a

Graphics Processor Unit (GPU) and memory while consulting for Rendition, Inc. From 1994 to 1996, I worked on the design of displays at Micron Display and Micron Technology, Inc. (“Micron”) in Boise, Idaho. This work was at a time when cathode ray tubes (CRTs) were still the dominant type of display. Flat panel displays were being developed with the hope of replacing CRTs in the consumer market. I worked on flat panel displays which resulted in 5 patents: 5,598,156, 5,638,085, 5,818,365, 5,894,293, and 5,909,201. I worked on the design of the pixels, both active and passive, as well as the supporting electronics for processing video signals. I was involved with the evaluation of display technologies including liquid crystal displays (LCDs), light-emitting diodes, plasma displays, and organic light emitting diode (OLED) displays; the display technologies that were looking to displace CRTs in the consumer market. I was also involved with the packaging of the displays including the vacuum sealing and deposition of the phosphors for light wavelength conversion. I also taught display design as a topic in my courses and did display design consulting again in industry for Cirque in 2013.

13. Further, I did consulting work in 2009 and 2010 at Sun Microsystems and then Oracle on the design of memory modules. This work entailed the design of low-power, high-speed, and wide interconnection methods with the goal of transmitting data to/from the memory module at higher speeds.

14. My current research work is focused in part on the design of integrated circuits for wireless sensing using LIDAR (LIght Detection And Ranging). I have worked with several companies in the development of these circuits and systems including Freedom Photonics, Aerius Photonics, and FLIR. In the early 1990s, I worked on wireless systems for wideband impulse radar while at Lawrence Livermore Laboratory. Further, part of my research for several years focused on the digitization of IQ channels using delta-sigma modulation. The knowledge and experience gained from this effort are reflected in my textbook CMOS Mixed-Signal Circuit Design and a presentation, which I have presented at several universities and companies,

http://cmosedu.com/jbaker/papers/talks/BP_DSM_talk.pdf.

C. Academic Experience

15. I have been teaching electrical engineering since 1991. From 1991-1993, I was an adjunct faculty member in the electrical engineering department of the University of Nevada, Las Vegas and University of Nevada, Reno. From 1993 to 2000, I served on the faculty at the University of Idaho as an Assistant Professor and then as a tenured Associate Professor of Electrical Engineering. In 2000, I joined a new electrical and computer engineering program at Boise State University (“BSU”) where I served as department chair from 2004 to 2007. At BSU I helped establish graduate programs in electrical and computer engineering

including, in 2006, the university's second Ph.D. degree. In 2012, I re-joined the faculty at UNLV where I am currently a Professor of Electrical and Computer Engineering. Over the course of my career as a professor, I have advised over 100 graduate students.

16. I have been recognized for my contributions as an educator and researcher in electrical and computer engineering. While at Boise State University, I received the President's Research and Scholarship Award (2005), Honored Faculty Member recognition (2003), and Outstanding Department of Electrical Engineering Faculty recognition (2001). In 2007, I received the Frederick Emmons Terman Award (the "Father of Silicon Valley"). The Terman Award is bestowed annually upon an outstanding young electrical/computer engineering educator in recognition of the educator's contributions to the profession. In 2011 I received the IEEE Circuits and Systems Education Award. I have also received the Tau Beta Pi Outstanding Electrical and Computer Engineering Professor Award each year I have been at UNLV.

17. I have authored several books and papers in the electrical and computer engineering area. My published books include CMOS Circuit Design, Layout, and Simulation (Baker, R. J., Wiley-IEEE, ISBN: 978-0470881323 (3rd ed., 2010)) and CMOS Mixed-Signal Circuit Design (Baker, R. J., Wiley-IEEE, ISBN: 978-0470290262 (2nd ed., 2009) and ISBN: 978-0471227540 (1st ed.,

2002)). I co-authored DRAM Circuit Design: Fundamental and High-Speed Topics (Keeth, B., Baker, R. J., Johnson, B., and Lin, F., Wiley-IEEE, ISBN: 978-0-470-18475-2 (2008)), DRAM Circuit Design: A Tutorial (Keeth, B. and Baker, R. J., Wiley-IEEE, ISBN: 0-7803-6014-1 (2001)), and CMOS Circuit Design, Layout and Simulation (Baker, R. J., Li, H.W., and Boyce, D.E., Wiley-IEEE, ISBN: 978-0780334168 (1998)). I contributed as an editor and co-author on several other electrical and computer engineering books.

D. Other Relevant Experience

18. I have performed technical and expert witness consulting for more than 200 companies and their subsidiaries and given more than 50 invited talks at conferences, companies, and universities. I am the author and co-author of more than 100 papers and presentations in the areas of solid-state circuit design and packages. A list of my publications and patents are on my curriculum vitae in Appendix B.

19. I currently serve, or have served, as a volunteer on: the IEEE Press Editorial Board (1999-2004); as editor for the Wiley-IEEE Press Book Series on Microelectronic Systems (2010-2018); as the Technical Program Chair of the 2015 IEEE 58th International Midwest Symposium on Circuits and Systems (MWSCAS 2015); on the IEEE Solid-State Circuits Society (SSCS) Administrative Committee (2011-2016); as a Distinguished Lecturer for the SSCS (2012-2015); the

Technology Editor (2012-2014) and Editor-in-Chief (2015-2020) for IEEE Solid-State Circuits Magazine; IEEE Kirchhoff Award Committee (2020-2023); and advisor for the student branch of the IEEE at UNLV (2013-present). These meetings, groups, and publications are intended to allow researchers to share and coordinate research. In addition to the above, I am an IEEE Fellow and a member of the honor societies Eta Kappa Nu and Tau Beta Pi.

20. My active participation in these meetings, groups, and publications allowed me to see what other researchers in the field have been doing.

21. During the past several years, I have provided expert testimony, depositions, reports, or declarations in the cases shown in Appendix B.

III. Person Of Ordinary Skill In The Art

22. I have been informed by counsel that the content of a patent (including its claims) and prior art should be interpreted the way a POSITA would have interpreted the material at the time of invention. I understand from counsel that the person of ordinary skill is a hypothetical person who is assumed to be aware of all the pertinent information that qualifies as prior art. In addition, the POSITA is a person of ordinary creativity who makes inferences and takes creative steps.

23. I have been informed by counsel that the “time of invention” I should use for purposes of this declaration is the date that the applicant for the ’466 Patent

filed the earliest parent application in the United States Patent and Trademark Office, namely, January 28, 2004. I have not provided an opinion as to whether the claims of the '466 Patent are entitled to that priority date.

24. It is my opinion that a POSITA at the time of the invention would have had a bachelor's degree in electrical engineering, computer engineering, or an equivalent field as well as at least two years of academic or industry experience in design and implementation of high-speed digital communication systems. An individual with an advanced degree in a relevant field, such as computer or electrical engineering, would require less experience in the design and implementation of high-speed digital communication systems, and vice versa.

25. In addition to my testimony as an expert, I am prepared to testify as someone who actually practiced in the field from 1985 to present, who actually possessed at least the knowledge of a POSITA in that time period, and who actually worked with others possessing at least the knowledge of a POSITA in that time period.

IV. Legal Understanding

26. I have a general understanding of validity based on my experience with patents and my discussions with counsel.

27. I have a general understanding of prior art and priority date based on my experience with patents and my discussions with counsel.

28. I understand from my discussions with counsel that inventors may be entitled to a priority date earlier than the date of filing to the extent that they can show complete possession of particular claimed inventions at such an earlier priority date and reasonable diligence to reduce the claims to practice between such an earlier priority date and the date of filing of the patent. I understand from my discussions with counsel that if the patent holder contends that particular claims are entitled to an earlier priority date than the date of filing of the patent, then the patent holder has the burden to prove this contention with specificity.

29. I understand that, subject to the next paragraph, the asserted “priority date” of a patent is the earlier of: (a) the date on which a patent application is filed; or (b) the date on which an earlier-filed patent application is filed if the patent holder claims the benefit of priority to that earlier-filed patent application.

30. I understand that it is not enough for a patent to merely claim the benefit of an earlier-filed application, but that additional criteria must be met. In particular, the prior application itself must describe the claimed invention, and must do so in sufficient detail that one skilled in the art can clearly conclude that the inventor invented the claimed invention as of the filing date sought. First, I understand that a priority-date analysis is on a claim-by-claim basis. Second, I understand that, in order for a patent claim to be entitled to the filing date of an earlier patent application, a “Section 112 analysis” must be conducted. I am

informed that a “Section 112” analysis encompasses looking to the earlier patent application, and ascertaining whether that earlier patent application meets both the written-description and enablement requirements as of the filing date of the earlier application. I understand that it is not enough that the claim would have been obvious from the earlier application, but that application itself must describe the claimed invention.

31. I understand that in order to satisfy the written description requirement the earlier application must reasonably convey to those skilled in the art that the inventors had possession of the subject matter of the patent as of the filing date of the earlier application. I have been informed that it is the disclosure of the earlier patent application that counts, and that while the meaning of terms, phrases, or diagrams in the earlier patent application must be interpreted from the vantage point of one skilled in the art, all of the claimed limitations must appear in the specification. Further, I understand that this analysis is not a question of whether one skilled in the art might be able to construct the claimed invention from the teachings of the disclosure. The question is not whether a claimed invention is an obvious variant of that which is disclosed in the specification; rather, I understand that an earlier application must itself describe each of the claim limitations.

32. I also understand that in order to satisfy the enablement requirement, the earlier application must enable a POSITA to practice the claimed invention without undue experimentation.

33. I understand from my discussions with counsel that an invention by another must be made before the priority date of a particular patent claim in order to qualify as “prior art” under 35 U.S.C. § 102 or § 103, that a printed publication or a product usage must be publicly available before the date of invention of a particular patent claim in order to qualify as “prior art” under 35 U.S.C. § 102(a), that a printed publication or a product usage or offer for sale must be publicly available more than one year prior to the date of the application for patent in the United States in order to qualify as “prior art” under 35 U.S.C. § 102(b), or that the invention by another must be described in an application for patent filed in the United States before the priority date of a particular patent claim in order to qualify as “prior art” under 35 U.S.C. § 102(e). I understand from my discussions with counsel that the Petitioner has the burden of proving that any particular reference or product usage or offer for sale is prior art.

34. I have a general understanding of anticipation based on my experience with patents and my discussions with counsel.

35. I understand that anticipation analysis is a two-step process. The first step is to determine the meaning and scope of the asserted claims. Each claim must

be viewed as a whole, and it is improper to ignore any element of the claim. For a claim to be anticipated under U.S. patent law: (1) each and every claim element must be disclosed, either explicitly or inherently, in a single prior art reference; (2) the claim elements disclosed in the single prior art reference must be arranged in the same way as in the claim; and (3) the invention must be disclosed in the single prior art reference, in as complete detail as set forth in the claim. Where even one element is not disclosed in a reference, the anticipation contention fails. Moreover, to serve as an anticipatory reference, the reference itself must be enabled, i.e., it must provide enough information so that a POSITA can practice the subject matter of the reference without undue experimentation.

36. I have a general understanding of obviousness based on my experience with patents and my discussions with counsel.

37. I understand that a patent claim is invalid under 35 U.S.C. § 103 as being obvious only if the differences between the claimed invention and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person of ordinary skill in that art. An obviousness analysis requires consideration of four factors: (1) scope and content of the prior art relied upon to challenge patentability; (2) differences between the prior art and the claimed invention; (3) the level of ordinary skill in the art at the time of the invention; and (4) the objective evidence of non-obviousness, such as commercial

success, unexpected results, the failure of others to achieve the results of the invention, a long-felt need which the invention fills, copying of the invention by competitors, praise for the invention, skepticism for the invention, or independent development.

38. I understand that a prior art reference is proper to use in an obviousness determination if the prior art reference is analogous art to the claimed invention. I understand that a prior art reference is analogous art if at least one of the following two considerations is met. First, a prior art reference is analogous art if it is from the same field of endeavor as the claimed invention, even if the prior art reference addresses a different problem and/or arrives at a different solution. Second, a prior art reference is analogous art if the prior art reference is reasonably pertinent to the problem faced by the inventor, even if it is not in the same field of endeavor as the claimed invention.

39. I understand that it must be shown that a POSITA at the time of the invention would have had a reasonable expectation that a modification or combination of one or more prior art references would have succeeded. Furthermore, I understand that a claim may be obvious in view of a single prior art reference, without the need to combine references, if the elements of the claim that are not found in the reference can be supplied by the knowledge or common sense of one of ordinary skill in the relevant art. However, I understand that it is

inappropriate to resolve obviousness issues by a retrospective analysis or hindsight reconstruction of the prior art and that the use of “hindsight reconstruction” is improper in analyzing the obviousness of a patent claim.

40. I further understand that the law recognizes several specific guidelines that inform the obviousness analysis. First, I understand that a reconstructive hindsight approach to this analysis, i.e., the improper use of post-invention information to help perform the selection and combination, or the improper use of the listing of elements in a claim as a blueprint to identify selected portions of different prior art references in an attempt to show that the claim is obvious, is not permitted. Second, I understand that any prior art that specifically teaches away from the claimed subject matter, i.e., prior art that would lead a POSITA to a specifically different solution than the claimed invention, points to non-obviousness, and conversely, that any prior art that contains any teaching, suggestion, or motivation to modify or combine such prior art reference(s) points to the obviousness of such a modification or combination. Third, while many combinations of the prior art might be “obvious to try,” I understand that any obvious-to-try analysis will not render a patent invalid unless it is shown that the possible combinations are: (1) sufficiently small in number so as to be reasonable to conclude that the combination would have been selected; and (2) such that the combination would have been believed to be one that would produce predictable

and well understood results. Fourth, I understand that if a claimed invention that arises from the modification or combination of one or more prior art references uses known methods or techniques that yield predictable results, then that factor also points to obviousness. Fifth, I understand that if a claimed invention that arises from the modification or combination of one or more prior art references is the result of known work in one field prompting variations of it for use in the same field or a different one based on design incentives or other market forces that yields predictable variations, then that factor also points to obviousness. Sixth, I understand that if a claimed invention that arises from the modification or combination of one or more prior art references is the result of routine optimization, then that factor also points to obviousness. Seventh, I understand that if a claimed invention that arises from the modification or combination of one or more prior art references is the result of a substitution of one known prior art element for another known prior art element to yield predictable results, then that factor also points to obviousness.

41. I understand from my discussions with counsel that the obviousness analysis should further consider evidence of “secondary considerations” such as commercial success, long-felt but unresolved needs, copying of the invention by competitors, and failures of others.

42. I understand that a dependent claim incorporates each and every limitation of the claim from which it depends. Thus, my understanding is that if a prior art reference fails to anticipate an independent claim, then that prior art reference also necessarily fails to anticipate all dependent claims that depend from the independent claim. Similarly, my understanding is that if a prior art reference or combination of prior art references fails to render obvious an independent claim, then that prior art reference or combination of prior art references also necessarily fails to render obvious all dependent claims that depend from the independent claim.

V. The '466 Patent

43. The '466 Patent, titled "Periodic calibration for communication channels by drift tracking" relates to methods and systems for calibrating a communication channel in a system including a receive component with circuitry to receive a digital signal. Ex. 1001, Abstract, Fig. 1, Fig. 3.

44. Figure 3 (below) depicts a transmit component 100 including a transmitter circuit 103 driving a data signal over link 102 to a receive component 101 including a receiver circuit 104. *Id.*, 7:31-42.

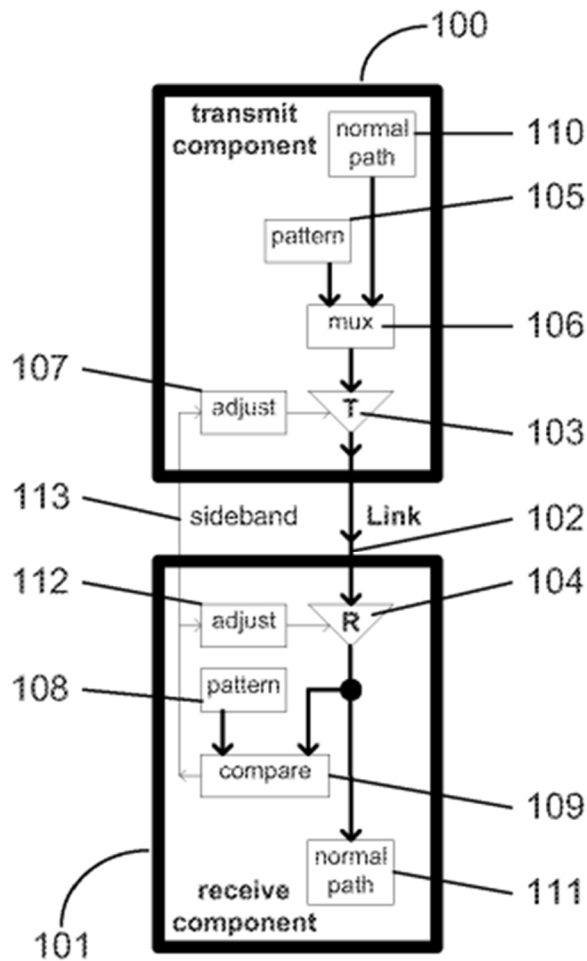


Fig. 3

45. The calibration methods include a first calibration (e.g., during initialization) and a second calibration (e.g., during operation). *Id.*, Abstract. The '466 Patent claims to have invented the use of a simplified second calibration to account for drift in timing, voltage or other parameters of a communication channel caused by changes in operation conditions, without repeating a more exhaustive initial calibration. *Id.*, 2:58-3:5.

46. The '466 Patent explains that the first calibration identifies an initial value for a parameter of the communication channel, and that the first calibration may be exhaustive so that a suitable operation value can be determined for most conditions in which the system is designed to operate. *Id.*, 3:12-16, 3:33-38, 6:25-38. The '466 Patent provides examples of the calibrated parameter as “timing parameters specifying the drive times and sample times, voltage levels for drivers and comparators, resistance values such as link termination resistances, driver strength, adaptive equalization coefficients, noise cancellation coefficients, parameters that cause overshoot and undershoot of signals such as driver switching power or speed, and so on.” *Id.*, 6:28-35.

47. The second calibration of the '466 Patent updates an existing value of the calibrated parameter to account for drift attributable to changes in operating conditions such as voltage or temperature. *Id.*, 2:31-3:11, 3:38-41. The second calibration of the '466 Patent is described as being performed periodically and as using less resources of the communication channel than the first calibration. *Id.*, 10:20-33.

48. The '466 Patent explains that these calibrations may be performed on the receive component of the communication system, which includes circuitry to receive data communicated across a channel by a transmit component. *Id.*, 7:31-42,

10:59-64. The '466 Patent explains that performing the second calibration may involve suspending operations of the receive component. *Id.*, 14:3-5.

49. The '466 Patent provides an example calibration technique as sending first and second data patterns from a transmit component to the receive component and comparing those data patterns to expected patterns. *Id.*, 9:22-43.

50. The '466 Patent includes 19 claims. I have been informed by counsel that Claims 1-19 are the subject of the *Inter Partes* Review petition. Solely purposes of my analysis herein, I have denoted certain elements of the claims as (a), (b), etc. even though such nomenclature does not appear in the '466 Patent.

VI. Priority Date of the '466 Patent

51. I have been informed by counsel that the priority date that I should use for purposes of this declaration is the date that the applicant for the '466 Patent filed the earliest parent application in the United States Patent and Trademark Office, namely, January 28, 2004. I have not provided an opinion as to whether the claims of the '466 Patent are entitled to that priority date.

VII. Claim Construction

52. I understand that claim terms are typically given their ordinary and customary meaning, as would have been understood by a POSITA, at the time of the alleged invention. In considering the meaning of the claims, however, I understand that one must consider the language of the claims, the specification, the

prosecution history of record, and the knowledge of a POSITA at the relevant time.

I have followed these principles in my analysis. I believe all terms in the challenged claims should be given their ordinary and customary meaning, as would have been understood by a POSITA at the time of the purported invention.

VIII. State of the Prior Art

53. As of January 28, 2004, the state of the art in the field of calibration techniques for digital communication systems already fully encompassed the concepts of the methods and systems of the '466 Patent, as evidenced in even the small sample of the art described herein.

A. Calibration of Digital Communication Systems

54. Techniques for calibrating digital communication systems have existed as long as digital communication systems have existed. Like most instruments, systems, or devices, some amount of calibration is needed, at least at initialization, to allow for smooth operation. In digital communication systems, calibration is particularly important to synchronize components in a data channel. For example, parameters of the transmitting and receiving components may need to be calibrated to allow for efficient and effective communication of data.

55. Calibration techniques are particularly well known in “mesochronous” systems, i.e., systems in which the transmitter and receiver use clocks that run with the same frequency, but unknown phases. In such a system, calibration is

performed to adjust the timing for both the transmitter and receiver, such that the receiver knows when to expect (and thus read) the signal from the transmitter.

These calibration techniques are well understood. For example, a known data pattern may be sent across the channel while parameters (e.g., clock delay, offset) are adjusted, to find the parameters that result in the best data transfer. This initial calibration be performed under a variety of conditions (e.g., ambient temperature, system temperature, voltage) to allow the system to adapt to various conditions.

56. External factors such as temperature changes, voltage fluctuations, and aging components may cause the calibrated system to drift out of sync (e.g., the timing for the receiving and transmitted may no longer be synchronized). Digital communication systems account for these factors by recalibrating the system periodically or in response to measured drift.

B. Meaney (Ex. 1004)

57. U.S. Patent Publication No. 2003/0070123, which I refer to as “Meaney,” was filed on September 21, 2001 and published on April 10, 2003. Ex. 1004, cover page.

58. Meaney describes an apparatus and method for recalibrating a source-synchronous pipelined interface with minimal impact on a running system which allows a computer system to remain operational despite environmental drift or degradation. Ex. 1004, [0001].

59. Meaney explains that in symmetric multiprocessing (SMP) systems, managing latencies on cards, wires, or boards that exceed data cycle times is crucial, including source-synchronous pipelined interfaces that capture data within a precise timing window, which requires initial calibration with a known data pattern. Ex. 1004, [0003]-[0004]. Meaney further explains that over time, these interfaces can drift due to environmental changes (e.g., temperature), leading to potential system failures without recalibration. Ex. 1004, [0005]-[0006].

60. Meaney states that it solves this problem by providing periodic recalibration with minimal disruption by putting the system into a wait state, performing fast initialization, and resuming operation. Ex. 1004, [0009]-[0012]. The recalibration is a simplified form of the initial calibration, using one less single clock centering step and skipping a data deskew step. Ex. 1004, [0013]. The system assist processor (SAP) controls the sequence, ensuring the interface is idle before calibration by interacting directly with hardware registers and loading calibration patterns. Ex. 1004, [0036]-[0038]. The recalibration logic re-centers the clock efficiently, optimizing the timing window to adapt to environmental changes. Ex. 1004, [0040]. Upon completion, the system resumes normal operation, maintaining precise timing and continuous functionality. Ex. 1004, [0041].

C. Nguyen (Ex. 1005)

61. U.S. Patent Publication No. 2003/0151450, which I refer to as “Nguyen,” was filed on May 28, 2002 and published on August 14, 2003. Ex. 1005 at cover page.

62. Nguyen describes methods for applying offsets to existing values to determine a calibrated value. Ex. 1005, [0065]. This technique ensures that the reference voltage is accurately set to match the common mode voltage of received signals, thus maintaining signal integrity. Ex. 1005, [0004], [0059]-[0064].

63. Nguyen also discloses circuitry including components for receiving a digital signal and calibrating computer system parameters. Ex. 1005, [0083]-[0086]. A digital calibration component determines high and low compensated voltage failure points and calculates a midpoint value to set the reference voltage, ensuring it matches the common mode voltage of the received signal. Ex. 1005, [0083]-[0086].

64. Additionally, Nguyen details a parameters table for storing calibration data for different transmitting units. Ex. 1005, [0075]-[0078], [0098]-[0100]. This allows the system to dynamically adjust the reference voltage based on the specific characteristics of each unit and environmental conditions. Ex. 1005, [0100], [0147].

D. Greeff (Ex. 1006)

65. U.S. Patent No. 6,356,106, which I refer to as “Greeff,” was filed on September 12, 2000 and issued on March 12, 2002. Ex. 1006 at cover page.

66. Greeff describes a system that addresses the issues of signal reflections and integrity in digital systems through an active termination scheme that integrates termination circuits within the devices connected to a multidrop bus rather than on the system’s printed circuit board (PCB). Ex. 1006, Abstract, 1:5-20. Greeff states that this integration allows for selective enabling or disabling of termination based on the device location and communication traffic, thereby reducing cost and conserving PCB space. Ex. 1006, Abstract, 2:13-34.

67. A key aspect of Greeff is the calibration of termination resistance and driver strength to optimize signal integrity. Ex. 1006, 4:11-14, 6:3-6, 7:52-8:62. Greeff discloses adjusting the termination resistance during a calibration process that can be either static (performed after system configuration) or dynamic (adjusted during system operation) to account for variations in process, voltage, or temperature. Ex. 1006, 7:52-8:3.

E. Allee (Ex. 1007)

68. Another example amongst the numerous prior art references in this field is U.S. Patent No. 6,255,979, which I refer to as “Allee.” Allee was filed on February 24, 1999 and issued on July 3, 2001. Ex. 1007 at cover page.

69. Allee provides a system including a plurality of comparators, each receiving differential input and reference signals to generate output signals. Ex. 1007, Abstract. A self-calibration circuit within the system adjusts these differential reference signals to mitigate noise from digital switching, ensuring accurate signal processing by compensating for component mismatches and non-linearities. *Id.*, Abstract, 2:13-34. This design enhances the performance of the system in environments with significant digital switching noise. *Id.*

70. Each comparator in the system undergoes self-calibration to minimize input offset errors, where the reference voltage is adjusted until the comparator switches states accurately. Ex. 1007, 6:17-40. The calibration involves sequentially fine-tuning each comparator, starting from the lowest quantized level, to achieve accurate threshold points for reliable system performance. Ex. 1007, 6:17-40, 7:24-40.

IX. Ground 1: Meaney in view of the knowledge of a POSITA

71. In my opinion, claims 1, 2, 6-12, 15, and 16 of the '466 Patent are obvious over Meaney in view of the knowledge of a POSITA. I have provided my element-by-element analysis in Appendix A to this declaration.

X. Ground 2: Meaney and Nguyen in view of the knowledge of a POSITA.

72. In my opinion, claims 1, 2, 6-12, 15, and 16 of the '466 Patent are obvious over Meaney and Nguyen in view of the knowledge of a POSITA. I have provided my element-by-element analysis in Appendix A to this declaration.

XI. Ground 3: Meaney and Greeff in view of the knowledge of a POSITA.

73. In my opinion, claims 4, 5, 13, 14, and 17-19 of the '466 Patent are obvious over Meaney and Greeff in view of the knowledge of a POSITA. I have provided my element-by-element analysis in Appendix A to this declaration.

XII. Ground 4: Meaney, Nguyen, and Greeff in view of the knowledge of a POSITA.

74. In my opinion, claims 4, 5, 13, 14, and 17-19 of the '466 Patent are obvious over Meaney, Nguyen, and Greeff in view of the knowledge of a POSITA. I have provided my element-by-element analysis in Appendix A to this declaration.

XIII. Ground 5A: Meaney and Allee in view of the knowledge of a POSITA.

75. In my opinion, claim 3 of the '466 Patent is obvious over Meaney and Allee in view of the knowledge of a POSITA. I have provided my element-by-element analysis in Appendix A to this declaration.

XIV. Ground 5B: Meaney, Nguyen, and Allee in view of the knowledge of a POSITA.

76. In my opinion, claim 3 of the '466 Patent is obvious over Meaney, Nguyen, and Allee in view of the knowledge of a POSITA. I have provided my element-by-element analysis in Appendix A to this declaration.

XV. CONCLUSION

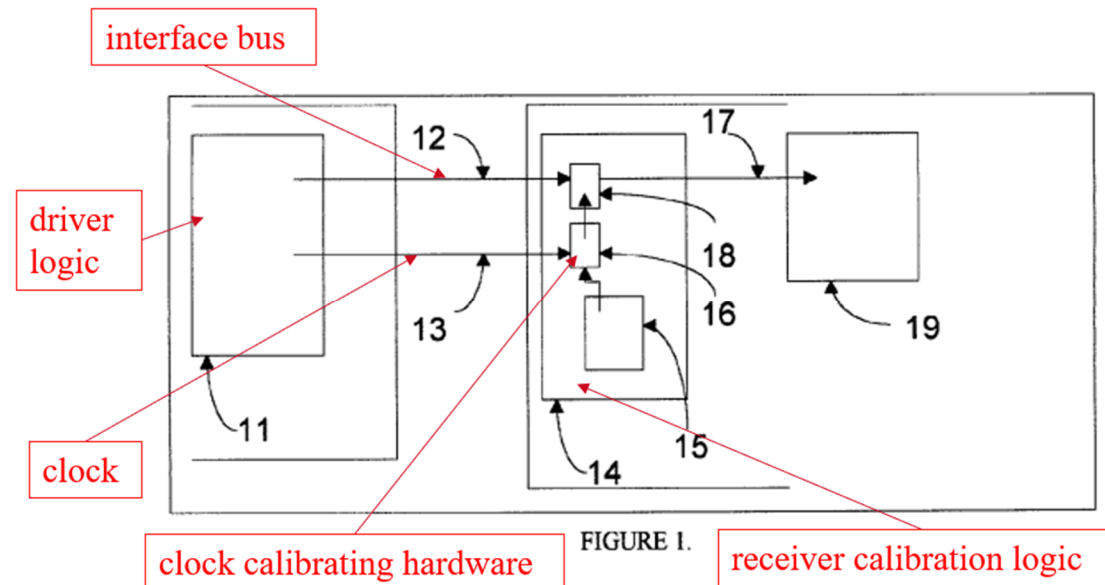
77. In my opinion, claims 1-19 of the '466 Patent are invalid for at least the reasons stated above.

78. I reserve the right to supplement my opinions in the future to respond to any arguments raised by the owner of the '466 Patent and to take into account new information that becomes available to me.

APPENDIX A: LIMITATION-BY-LIMITATION ANALYSIS

GROUND 1: Meaney in view of the knowledge of a POSITA

Claim 1	
1 [pre]: A method of operation in a system that includes a receive component having circuitry to receive a digital signal, the method comprising:	<u>Discussion</u> Meaney discloses this limitation. Meaney discloses “symmetrical computer systems, and particularly to an apparatus and method for recalibrating [a] source-synchronous pipelined interface with minimal impact to a running system.” Ex. 1004, Abstract, [0001]. Meaney’s systems include a “receiver” and “receiver calibration logic 14” for capturing transferred data. <i>Id.</i>, [0003]-[0004], [0023]. As shown in Figure 1 below, Meaney’s interface includes driver logic 11, used to drive interface bus 12, and clock 13, which feeds receiver calibration logic 14. <i>Id.</i>, [0023]. The receiver logic 14 includes state machine logic 15, clock calibrating hardware 16, and data de-skew hardware 17. <i>Id.</i> The output of receiver calibration logic 14, transfers data onto receiver output bus 18, which connects to system logic 19. <i>Id.</i> The receiver calibration logic 14 of Meaney is a receive component having circuitry to receive a digital signal.



Meaney's systems include circuitry for receiving a digital signal. For example, Meaney refers to the hardware and circuitry of its computer systems and the need to calibrate the receiver to compensate changes to the "circuit." *Id.*, [0007], [0015]-[0016], [0020]-[0021], claim 12. A POSITA would have understood that the receiver of Meaney would have circuitry to receive a digital signal. For these reasons, Meaney, in view of the knowledge of a POSITA, renders obvious "[a] method of operation in a system that includes a receive component having circuitry to receive a digital signal."

Other Relevant Citations

Meaney, Ex. 1004 at Abstract: "An SMP computer system has an apparatus and method for recalibrating a self-timed, source-synchronous, pipelined interface while the computer system is running. The apparatus allows for quiescing the interface (ie. idling the processors to allow for no data transfers), raising fences (blocking interfaces), allowing for a quick clock centering recalibration step, and then unfencing and unquiescing to allow for the use of the interface again. The recalibration allows for compensating for drift over time on the interface to compensate for temperature, voltage, cycle time, and end-of-life degradation without bringing down and restarting the system."

	<i>Id.</i> at [0001]: “This invention relates to symmetrical computer systems, and particularly to an apparatus and method for recalibrating an [sic] source-synchronous pipelined interface with minimal impact to a running system, allowing for the system to remain operational, even in the midst of environmental drift or degradation.” <i>Id.</i> at [0003]-[0004]: “In an SMP (symmetric multiprocessing) computer system with high-level packaging, it is often necessary to allow latencies on cards, wires, or boards that exceed the cycle times of the transferred data. In order to allow these transfers to take place, source-synchronous pipelined interfaces have been used in the s/390 machines of IBM. This allows for the proper capturing of the data within a small window, or ‘eye’. In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.” <i>Id.</i> at [0007]: “Also, in a test environment, we often apply environmental stress to a machine as well as frequency changes. We often want to run a hardware/architecture exerciser (like SAK or PCX) under several environments (about 16 voltage corners as well as cycling up or down the frequency), the interface often fails without a re-calibration. If the system is not recalibrated under the new environment, the system must be restarted for every test. Therefore, our test time is jeopardized.” <i>Id.</i> at [0015]-[0016]: “By keeping the interface from being used, the driving circuits can exclusively send the calibration pattern and the calibration logic can re-center the clock under any new environmental conditions or circuit changes. The invention can be triggered periodically. This way, the changes to circuit or environmental characteristics over time do not adversely affect the operation of the interface.” <i>Id.</i> at [0020]-[0021]: “FIG. 2 illustrates the prior art state diagram for performing the initial calibration of the hardware as performed in conjunction with the prior art; FIG. 3 illustrates the preferred embodiment state diagrams for supporting the initial calibration as well as the recalibration of the hardware as performed in conjunction with the invention.” <i>Id.</i> at [0023]: “Turning to FIG. 1, notice that there is driver logic, 11, used to drive interface bus, 12, and clock, 13, which feeds receiver calibration logic, 14. Said receiver logic, 14, consisting of state machine logic, 15, clock calibrating hardware, 16, and data de-skew hardware 17. The output of receiver calibration logic, 14, transfers data onto receiver output bus, 18, which connects to system logic, 19.” <i>Id.</i> at Fig. 1:
--	---

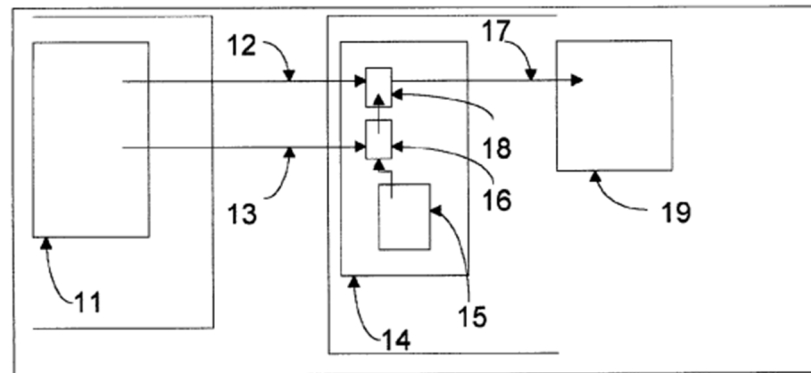


FIGURE 1.

Id. at Fig. 2:

E.I. CLOCK RECEIVER STATE MACHINE DIAGRAM

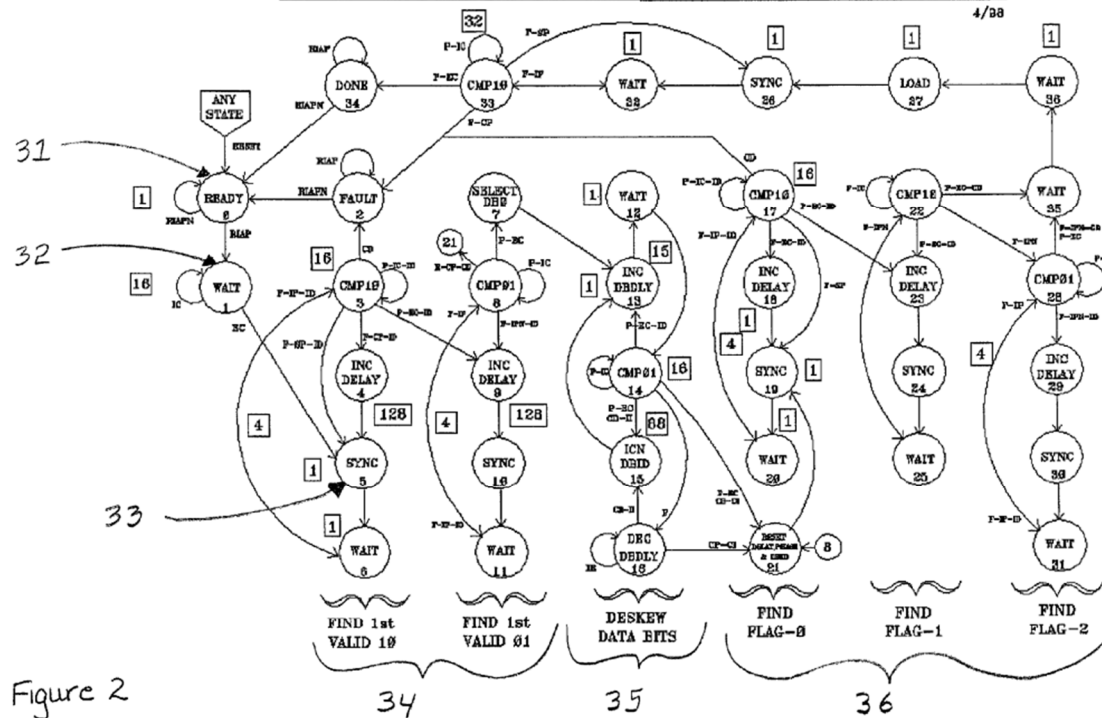


Figure 2

Id. at Fig. 3:

	delay. <i>See</i> Ex. 1004, [0040] (“clock delays are reset as part of the final clock calibration sequence.”); [0051] (“[f]or instance, if it is known that only the frequency changes, the change in frequency can be calculated by additional hardware and half the difference can be applied as a delay shift”); claim 12 (“recalibration []includes . . . re-applying the clock frequency calculation to the clock delay to re-center the clock”) (emphasis added). Thus, Meaney discloses determining the initial value for clock delay during initialization in order to arrive at the original clock delay. <u>Other Relevant Citations</u> Meany, Ex. 1004 at [0004]: “In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.” <i>Id.</i> at [0006]: “The prior s/390 IBM machines only handled the first calibration. There was no way to re-calibrate an interface once it was up and running. Therefore, the cumulative degradations could eventually cause a system failure.” <i>Id.</i> at [0021]: “FIG. 3 illustrates the preferred embodiment state diagrams for supporting the initial calibration as well as the recalibration of the hardware as performed in conjunction with the invention.” <i>Id.</i> at [0024]¹: “In the prior art, the system was always recalibrated from a stopped position. The system would be scan-initialized to a reset state and the calibration would begin when the clocks were started.” <i>Id.</i> at [0038]: “The SAP next signals the receiver calibration logic to perform the fast initialization procedure. Since the data deskew has already been done, it is not necessary to perform this lengthy step during recalibration. The fast initialization procedure consists of starting the state machine at the final clock calibration step.”
--	---

¹ Although *Meaney* refers to this discussion as prior art, *Meaney* is clear that the preferred method is “similar to the prior art calibration sequence” but includes “additional steps.” Ex. 1004 at [0035]. Thus, *Meaney* is clear that the initial calibration of the clock would be part of the preferred method as well.

	<i>Id.</i> at [0040]: “The clock delays are reset as part of the final clock calibration sequence. By adding delay to the clock path, the clock calibration logic is able to find the optimum data capture time for the interface. Since the calibration is done periodically, this window gets reoptimized every time recalibration occurs. Also, if the cycle time, voltage, or other changes are made to the environment for testing, the interface can recalibrate to the new conditions. However, the data deskew sequence does not need to be run since the deskew settings are not likely to change since the data nets will tend to track over time.” <i>Id.</i> at [0051]: “Although the preferred embodiment recalculates the flag0, flag1, and flag2 values, there are other methods that can be used to recalibrate the clock. For instance, if it is known that only the frequency changes, the change in frequency can be calculated by additional hardware and half the difference can be applied as a delay shift. Although this would be faster, we excluded this form of recalibration due to the complexity and the fact that the preferred embodiment was fast enough to avoid the system hangs that the full calibration would cause. Also, the preferred embodiment uses the same sequences used in the original.” <i>Id.</i> at claim 12: “The method according to claim 10 wherein the recalibration stem includes re-calculating the clock frequency of the interface against the current hardware and re-applying the clock frequency calculation to the clock delay to re-center the clock when the machine is being cycled down to failure and the major change needing re-calibration is cycle time.”
1[b]: for a parameter affecting proper reception by the circuitry of the receive component of data communicated across a channel as part of the digital signal;	<u>Discussion</u> Meaney discloses this limitation. For example, the calibrated parameter (clock delay) of Meaney affects proper reception by the circuitry of the receive component of data communicated across a channel as part of the digital signal. Ex. 1004, [0003], [0004], [0040]. Meaney states that in transferring data in its computer system, the data must be captured by the receiver within a small time window (referred to as an ‘eye’). <i>Id.</i>, [0003]. Meaney further explains that the calibration of the clock delay calibration “is able to find the optimum data capture time for the interface. Since the calibration is done periodically, this window gets reoptimized every time recalibration occurs.” <i>Id.</i>, [0040]. This allows the interface to recalibrate to new conditions such as changes to cycle time, voltage, or the computing environment. <i>Id.</i> <u>Other Relevant Citations</u>

	Meany, Ex. 1004 at [0003]: “In an SMP (symmetric multiprocessing) computer system with high-level packaging, it is often necessary to allow latencies on cards, wires, or boards that exceed the cycle times of the transferred data. In order to allow these transfers to take place, source-synchronous pipelined interfaces have been used in the s/390 machines of IBM. This allows for the proper capturing of the data within a small window, or ‘eye’.” <i>Id.</i> at [0004]: “In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.” <i>Id.</i> at [0040]: “The clock delays are reset as part of the final clock calibration sequence. By adding delay to the clock path, the clock calibration logic is able to find the optimum data capture time for the interface. Since the calibration is done periodically, this window gets reoptimized every time recalibration occurs. Also, if the cycle time, voltage, or other changes are made to the environment for testing, the interface can recalibrate to the new conditions. However, the data deskew sequence does not need to be run since the deskew settings are not likely to change since the data nets will tend to track over time.”
1[c]: periodically subjecting the receive component to a second calibration to update an existing value of the parameter for drift attributable to change in at least one of operating voltage or temperature; and	<u>Discussion</u> Meany discloses periodically recalibrating the interface. Ex. 1004, [0008] (“The invention allows for the re-calibration of the interface at periodic intervals.”); [0009]. Claim 1 is unclear whether “an existing value” refers to the initial value or some other value for the parameter. Either way, it is disclosed in Meany because Meany discloses recalibrating both (a) the initial value determined during the first calibration and (b) a value determined during a previous recalibration. Regarding alternative (a), as I discussed above in 1[a], Meany discloses identifying an initial value for clock delay, which discloses the “existing value of the parameter.” Regarding alternative (b), Meany discloses several periodic recalibrations, and the output from a previous recalibration could be the existing value for the next recalibration. Ex. 1004, [0008], [0016], [0040].

	According to Meaney, recalibrating the clock delay allows the receiver to compensate for “drift over time on the interface to compensate for temperature, voltage, cycle time, and end-of-life degradation.” <i>Id.</i>, Abstract, [0015], [0016], [0040]. Specifically, Meaney refers to recalibrating the receiver to update the clock delay to account for “cycle time, voltage, or other changes [] made to the environment for testing,” and states that the recalibration allows the interface to recalibrate to the new conditions. <i>Id.</i>, [0040]. A POSITA would have understood this as rendering obvious “subjecting the receive component to a second calibration to update an existing value of the parameter for drift attributable to change in at least one of operating voltage or temperature.” <u>Other Relevant Citations</u> Meany, Ex. 1004 at [0008]: “The invention allows for the re-calibration of the interface at periodic intervals with minor disruption. This invention is applicable to the SMP computer systems sold by IBM and others.” <i>Id.</i> at [0015]-[0016]: “By keeping the interface from being used, the driving circuits can exclusively send the calibration pattern and the calibration logic can re-center the clock under any new environmental conditions or circuit changes. The invention can be triggered periodically. This way, the changes to circuit or environmental characteristics over time do not adversely affect the operation of the interface.” <i>Id.</i> at [0040]: “The clock delays are reset as part of the final clock calibration sequence. By adding delay to the clock path, the clock calibration logic is able to find the optimum data capture time for the interface. Since the calibration is done periodically, this window gets reoptimized every time recalibration occurs. Also, if the cycle time, voltage, or other changes are made to the environment for testing, the interface can recalibrate to the new conditions. However, the data deskew sequence does not need to be run since the deskew settings are not likely to change since the data nets will tend to track over time.”
1[d]: wherein the existing value is dependent on the initial value	<u>Discussion</u> As I discussed above at 1[c], Meaney discloses two alternative values that can be the claimed “existing value”: the initial value for clock delay from an initial calibration or a clock delay value determined from a recalibration. Ex. 1004 at [0008], [0016], [0040]. Meaney discloses that both of these existing values of clock delay are dependent on the initial value identified by the first calibration.

	Thus, a POSITA would have understood that either alternative of the “existing values” of Meaney are dependent on the initial value. In the first alternative, where the initial value is the existing value, the disclosure of Meaney I cited and discussed above makes it clear that the initial value would be used as the starting point for the next recalibration. In the second alternative, where the existing value is the output of a previous recalibration, that existing value also depends on the initial value because it would be necessarily determined, directly, or indirectly based on one or more recalibrations of the initial value. To illustrate with a specific example for clock delay, for the first recalibration performed after the initial calibration, the initial clock delay value would be used as a starting point in the recalibration to generate a new clock delay, which was generated based on the initial clock delay value. For the next recalibration, the output of the first recalibration (the claimed “existing value”) would be the new clock delay, which was generated from, and thus dependent on, the initial value. This is disclosed in Meaney, which explains that recalibration reoptimizes the window for data capture by recalibrating the existing clock delay to account for changes in conditions, and discloses that this recalibration occurs without resetting the hardware. Ex. 1004, [0033], [0040], [0051]. Additionally, Meaney discloses that the clock can be recalibrated by “adding delay” to the clock path or applying a delay shift to the existing clock delay. Ex. 1004, [0040], [0051]. A POSITA would have understood that these recalibrations would be performed based on the existing clock delay value. To the extent that Patent Owner argues, or the Board concludes, that Meaney does not disclose the existing value being dependent on the initial value, it would have been obvious to a POSITA to modify Meaney to arrive at this limitation. For example, in light of the disclosure of “adding delay” and “delay shifts” in Meaney (as I discussed and cited above), a POSITA would have found it obvious to modify Meaney to use the initial value as the starting point and apply the results of the calibration to that value to determine a calibrated value. A POSITA would have been motivated to make this modification in order to increase the efficiency and speed of recalibration as compared to performing a completely new calibration without using an existing value. Moreover, Meaney’s disclosure of “adding delay” implies an inherent initial value. You cannot add a delay to nothing, so there must necessarily be an initial value, even if it is zero. <u>Other Relevant Citations</u>
--	---

	Meany, Ex. 1004 at [0004]: “In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.” <i>Id.</i> at [0008]: “The invention allows for the re-calibration of the interface at periodic intervals with minor disruption. This invention is applicable to the SMP computer systems sold by IBM and others.” <i>Id.</i> at [0016]: “The invention can be triggered periodically. This way, the changes to circuit or environmental characteristics over time do not adversely affect the operation of the interface.” <i>Id.</i> at [0033]: “The limitation with this sequence is that, once the interfaces are used, there is no means to get the interface recalibrated without resetting the state machine and starting the entire sequence over.” <i>Id.</i> at [0038]: “The SAP next signals the receiver calibration logic to perform the fast initialization procedure. Since the data deskew has already been done, it is not necessary to perform this lengthy step during recalibration. The fast initialization procedure consists of starting the state machine at the final clock calibration step.” <i>Id.</i> at [0039]: “This recalibration can be understood by looking to FIG. 3. Notice that the normal calibration sequences of the initial clock calibration sequence, 34, data deskew sequence, 35, and final clock calibration sequence, 36, still exist. In addition, at the time of recalibration, a new wait state, 41, is used, which allows the calibration logic to proceed directly to the final clock calibration sequence, 36.” <i>Id.</i> at [0040]: “The clock delays are reset as part of the final clock calibration sequence. By adding delay to the clock path, the clock calibration logic is able to find the optimum data capture time for the interface. Since the calibration is done periodically, this window gets reoptimized every time recalibration occurs. Also, if the cycle time, voltage, or other changes are made to the environment for testing, the interface can recalibrate to the new conditions. However, the data deskew sequence does not need to be run since the deskew settings are not likely to change since the data nets will tend to track over time.”
--	---

	<i>Id.</i> at [0051]: “Although the preferred embodiment recalculates the flag0, flag1, and flag2 values, there are other methods that can be used to recalibrate the clock. For instance, if it is known that only the frequency changes, the change in frequency can be calculated by additional hardware and half the difference can be applied as a delay shift. Although this would be faster, we excluded this form of recalibration due to the complexity and the fact that the preferred embodiment was fast enough to avoid the system hangs that the full calibration would cause. Also, the preferred embodiment uses the same sequences used in the original.”
1[e]: and wherein the second calibration is constrained to occur during a time period that is shorter than a time period of the first calibration.	<u>Discussion</u> A POSITA would have understood that Meaney discloses “the second calibration is constrained to occur during a time period that is shorter than a time period of the first calibration.” Meaney discloses that its recalibration is faster and has fewer steps than the initial calibration performed during initialization. Ex. 1004, [0008], [0013], [0038]-[0039]. Meaney explains that recalibration is performed using a “fast initialization” process that can occur with minor disruption to the computer system by skipping certain steps (e.g., a data deskew and a clock centering step) that are performed during the original initialization process. Ex. 1004, [0008], [0013], [0038], Fig. 3. Also, the initial calibration of Meaney requires stopping operation completely (or starting calibration from a stopped position), but the recalibration of Meaney allows the system to be put into a “wait” state for the duration of the recalibration. Ex. 1004, [0024], [0035], [0039], [0041]. For these reasons a POSITA would have understood that the recalibration of Meaney would occur during a time period that is shorter than the time period of Meaney’s initial calibration. <u>Other Relevant Citations</u> Meaney, Ex. 1004, at [0008]: “The invention allows for the re-calibration of the interface at periodic intervals with minor disruption. This invention is applicable to the SMP computer systems sold by IBM and others.” <i>Id.</i> at [0013]: “The fast initialization process in the preferred embodiment is similar to the prior art, except that there is only one clock centering step, instead of two. Also, the data deskew step is skipped since that step has already been performed as part of the original initialization process.”

	<i>Id.</i> at [0024] : “In the prior art, the system was always recalibrated from a stopped position. The system would be scan-initialized to a reset state and the calibration would begin when the clocks were started.” <i>Id.</i> at [0035]: “The preferred method of recalibration is similar to the prior art calibration sequence. However, it includes the additional steps of putting the system into a wait state, running a modified recalibration, and restoring the system from a wait state to a running state.” <i>Id.</i> at [0038]: “The SAP next signals the receiver calibration logic to perform the fast initialization procedure. Since the data deskew has already been done, it is not necessary to perform this lengthy step during recalibration. The fast initialization procedure consists of starting the state machine at the final clock calibration step.” <i>Id.</i> at [0039]: “This recalibration can be understood by looking to FIG. 3. Notice that the normal calibration sequences of the initial clock calibration sequence, 34, data deskew sequence, 35, and final clock calibration sequence, 36, still exist. In addition, at the time of recalibration, a new wait state, 41, is used, which allows the calibration logic to proceed directly to the final clock calibration sequence, 36.” <i>Id.</i> at [0041]: “When the calibration is complete, the SAP turns off the driver calibration pattern, resets the interface fences and takes the processors out of the wait state.”
--	---

Claim 2	
2: The method of claim 1, wherein periodically subjecting includes calculating an interval and repeatedly (a) performing the second calibration at each expiration of the interval and (b) resetting the interval.	<u>Discussion</u> Meaney discloses performing recalibration in “periodic intervals” and states that the recalibration method can be “triggered periodically.” Ex. 1004, [0008], [0015], claim 5. For these reasons, a POSITA would have understood the disclosure of Meaney to disclose “calculating an interval and repeatedly (a) performing the second calibration at each expiration of the interval and (b) resetting the interval.” To be triggered in periodic intervals, Meaney necessarily must involve calculating the interval, repeating the recalibration when the interval is expired, and resetting the interval between recalibrations. <i>Id.</i>

	To the extent Patent Owner argues this limitation is not taught by Meaney, it would have been obvious to a POSITA to modify Meaney to have the interval be calculated and repeatedly reset between recalibrations. Meaney discloses that periodic recalibration allows the system to ensure that changes to the circuitry or environmental characteristics do not adversely affect performance over time. Ex. 1004, [0016]. To achieve this objective, it would have been obvious to a POSITA to calculate an interval based on, for example, an expected time it would take an environmental characteristic (e.g., change in temperature) to affect performance, and then repeatedly recalibrate the system based on that calculated interval. Periodic recalibration based on intervals using timers were well known at the time of the ‘466 Patent. For example, U.S. Patent Publication No. 2002/0066001 (“Olarig”) discloses calibration timers that include time intervals for conducting a calibration. Olarig, Ex. 1010 at [0042]. When the timer of Olarig expires, the calibration is performed. <i>Id.</i> <u>Other Relevant Citations</u> Meany, Ex. 1004, at [0008]: “The invention allows for the re-calibration of the interface at periodic intervals with minor disruption. This invention is applicable to the SMP computer systems sold by IBM and others.” <i>Id.</i> at [0015]: “By keeping the interface from being used, the driving circuits can exclusively send the calibration pattern and the calibration logic can re-center the clock under any new environmental conditions or circuit changes.” <i>Id.</i> at [0016]: “The invention can be triggered periodically. This way, the changes to circuit or environmental characteristics over time do not adversely affect the operation of the interface.” <i>Id.</i> at claim 5: “In an SMP computer system having an source-synchronous interface, the method for re-calibration of the interface at periodic intervals comprising the steps of: a. putting the system of the interface into a wait state, b. performing a fast initialization process for calibration, c. taking the system of the interface out of said wait state.”
--	---

Claim 6	
6: The method of claim 1, wherein the second calibration uses less resources of the channel than the first calibration.	<u>Discussion</u> As I discussed above in Section 1[e], Meaney discloses that the recalibration is faster and has fewer steps than the first calibration, i.e., the initial calibration performed during the original initialization process. Ex. 1004, [0008], [0013], [0038]-[0039]. Specifically, Meaney discloses that recalibration is performed using a “fast initialization” process that can occur with minor disruption to the computer system by skipping certain steps (e.g., a data deskew and a clock centering step) that are performed during the original initialization process. Ex. 1004, [0008], [0013], [0038], Fig. 3. Because the recalibration of Meaney is faster than the initial calibration, it occupies the data channel for less time. Thus, a POSITA would have understood that Meaney discloses “second calibration uses less resources of the channel than the first calibration.” <u>Other Relevant Citations</u> Meany at [0008]: “The invention allows for the re-calibration of the interface at periodic intervals with minor disruption. This invention is applicable to the SMP computer systems sold by IBM and others.” <i>Id.</i> at [0013]: “The fast initialization process in the preferred embodiment is similar to the prior art, except that there is only one clock centering step, instead of two. Also, the data deskew step is skipped since that step has already been performed as part of the original initialization process.” <i>Id.</i> at [0038]: “The SAP next signals the receiver calibration logic to perform the fast initialization procedure. Since the data deskew has already been done, it is not necessary to perform this lengthy step during recalibration. The fast initialization procedure consists of starting the state machine at the final clock calibration step.” <i>Id.</i> at [0039]: “This recalibration can be understood by looking to FIG. 3. Notice that the normal calibration sequences of the initial clock calibration sequence, 34, data deskew sequence, 35, and final clock calibration sequence, 36, still exist. In addition, at the time of recalibration, a new wait state, 41, is used, which allows the calibration logic to proceed directly to the final clock calibration sequence, 36.”

Claim 7	
7[a]: The method of claim 1, wherein subjecting the receive component to the first calibration comprises receiving a first set of operations at the circuitry to exhaustively determine the initial value of the parameter, and	<u>Discussion</u> As I discussed above in Section VIII.A.1.ii, Meaney discloses subjecting the circuitry of the receive component to the first calibration by calibration of clock delay during initialization of the computer system. Meaney explains that the initial calibration is an exhaustive calibration, which involves sending a known data pattern across a data transfer interface to receive circuitry. Ex. 1004, [0004], [0024]-[0033], Fig. 3. This triggers the receiver circuitry to use exhaustive calibration techniques to determine an initial clock delay value that “compensates for the various package tolerances.” <i>Id.</i> Meaney explains that this calibration technique requires stopping the system and involves two separate clock calibrations and deskewing the data. <i>Id.</i> For these reasons, a POSITA would have understood that Meaney discloses “subjecting the receive component to the first calibration comprises receiving a first set of operations at the circuitry to exhaustively determine the initial value of the parameter.” <u>Other Relevant Citations</u> Meany at [0004]: “In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.” <i>Id.</i> at [0006]: “The prior s/390 IBM machines only handled the first calibration. There was no way to re-calibrate an interface once it was up and running. Therefore, the cumulative degradations could eventually cause a system failure.” <i>Id.</i> at [0021]: “FIG. 3 illustrates the preferred embodiment state diagrams for supporting the initial calibration as well as the recalibration of the hardware as performed in conjunction with the invention.” <i>Id.</i> at [0024]-[0032]: “In the prior art, the system was always recalibrated from a stopped position. The system would be scan-initialized to a reset state and the calibration would begin when the clocks were started. The high-level sequence for the prior art calibration (described in referenced Docket AT998212, Dynamic Wave-pipelined Interface Apparatus and Methods) included the steps of: 1. Setting interface fences (block interface from being used by system operations) 2. Resetting interface state machine 3. Turning on the driver calibration pattern

	4. Calibrating clock 5. Deskewing the data, one bit at a time 6. Recalibrating clock 7. Turning off driver calibration pattern 8. Resetting interface fences” <i>Id.</i> at [0033]: “The limitation with this sequence is that, once the interfaces are used, there is no means to get the interface recalibrated without resetting the state machine and starting the entire sequence over.” <i>Id.</i> at [0038]: “The SAP next signals the receiver calibration logic to perform the fast initialization procedure. Since the data deskew has already been done, it is not necessary to perform this lengthy step during recalibration. The fast initialization procedure consists of starting the state machine at the final clock calibration step.” <i>Id.</i> at [0040]: “The clock delays are reset as part of the final clock calibration sequence. By adding delay to the clock path, the clock calibration logic is able to find the optimum data capture time for the interface. Since the calibration is done periodically, this window gets reoptimized every time recalibration occurs. Also, if the cycle time, voltage, or other changes are made to the environment for testing, the interface can recalibrate to the new conditions. However, the data deskew sequence does not need to be run since the deskew settings are not likely to change since the data nets will tend to track over time.” <i>Id.</i> at [0051]: “Although the preferred embodiment recalculates the flag0, flag1, and flag2 values, there are other methods that can be used to recalibrate the clock. For instance, if it is known that only the frequency changes, the change in frequency can be calculated by additional hardware and half the difference can be applied as a delay shift. Although this would be faster, we excluded this form of recalibration due to the complexity and the fact that the preferred embodiment was fast enough to avoid the system hangs that the full calibration would cause. Also, the preferred embodiment uses the same sequences used in the original.”
--	---

Id. at claim 12: “The method according to claim 10 wherein the recalibration stem includes re-calculating the clock frequency of the interface against the current hardware and re-applying the clock frequency calculation to the clock delay to re-center the clock when the machine is being cycled down to failure and the major change needing re-calibration is cycle time.”

Id. at Fig. 3:

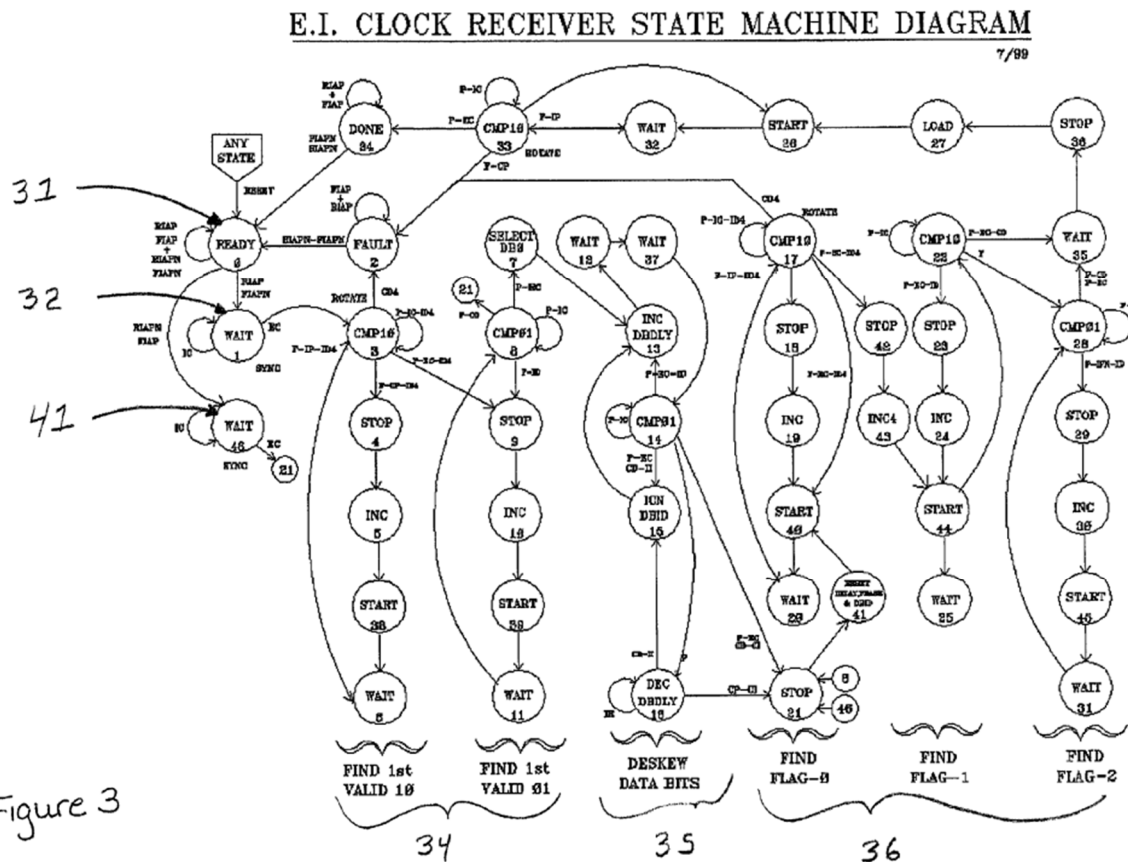


Figure 3

7[b]: wherein periodically subjecting the receive component to the second calibration comprises receiving a second set of operations at the circuitry to update the existing value if a desired value of the parameter has drifted from the existing value by more than a delta value.	<u>Discussion</u> As I discussed above with respect to 1[d], Meaney discloses subjecting the receive component to a second calibration (Meaney’s recalibration) to update an existing value of the parameter (clock delay) for drift. Meaney further discloses the receive component circuitry receiving a set of operations to perform the recalibration if a desired clock delay value has drifted from the existing value by more than a delta value. Ex. 1004, Abstract, [0001], [0005]. Meaney discloses that recalibration may be performed in response to a trigger event that suggests that the clock delay has drifted and the data capture time window has changed and is no longer accurate. Ex. 1004, [0017]. Meaney explains that this trigger event can be indicated by, for example, when error correction (ECC) is performed on the data and a correctable error is found. <i>Id.</i> Errors in data caused by clock delay drift would indicate that the clock delay has drifted by more than a delta value, i.e., by more than the amount that the system could tolerate without introducing errors into the data. For these reasons, a POSITA would have understood that Meaney, in view of the knowledge of a POSITA, renders obvious “wherein periodically subjecting the receive component to the second calibration comprises receiving a second set of operations at the circuitry to update the existing value if a desired value of the parameter has drifted from the existing value by more than a delta value.” To the extent Patent Owner argues that Meaney does not disclose recalibrating “a desired value of the parameter has drifted from the existing value by more than a delta value,” it would have been obvious to a POSITA to modify Meaney to include this feature. Meaney emphasizes the need to recalibrate clock delay to maintain accurate data capture despite drift to circuitry changes or changes in environmental conditions, and discloses recalibrating in response to a trigger event. Ex. 1004, Abstract, [0004]-[0005]. It would have been obvious to a POSITA to trigger recalibration in response to clock delay, or other parameters indicative of clock delay, drifting from an existing value by an amount beyond the tolerance of the computer system (the claimed “delta value”). <u>Other Relevant Citations</u>
--	--

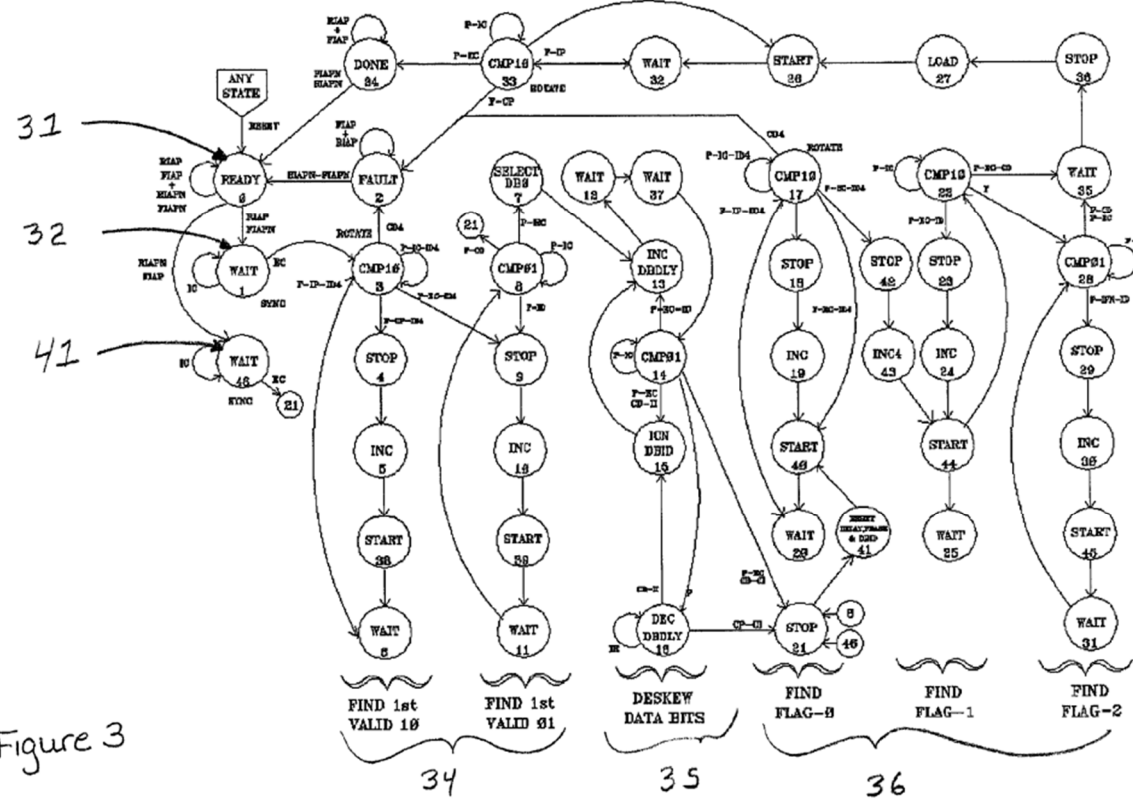
	Meany at Abstract: “An SMP computer system has an apparatus and method for recalibrating a self-timed, source-synchronous, pipelined interface while the computer system is running. The apparatus allows for quiescing the interface (ie. idling the processors to allow for no data transfers), raising fences (blocking interfaces), allowing for a quick clock centering recalibration step, and then unfencing and unquiescing to allow for the use of the interface again. The recalibration allows for compensating for drift over time on the interface to compensate for temperature, voltage, cycle time, and end-of-life degradation without bringing down and restarting the system.” <i>Id.</i> at [0001]: “This invention relates to symmetrical computer systems, and particularly to an apparatus and method for recalibrating an source-synchronous pipelined interface with minimal impact to a running system, allowing for the system to remain operational, even in the midst of environmental drift or degradation.” <i>Id.</i> at [0004]: “In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.” <i>Id.</i> at [0005]: “However, over time, these large-latency interfaces can drift due to environmental changes like voltage, temperature, end-of-life degradation, etc.” <i>Id.</i> at [0017]: “Also, the preferred embodiment allows for the proactive running of the recalibration sequence based on a trigger event. For instance, when ECC is used on an interface and a correctable error (CE) is detected, the data is corrected. The CE can trigger an event to recalibrate the interface. This should protect against future CEs, if the original CE was based on a degraded interface.”
--	--

Claim 8	
8: The method of claim 1, wherein periodically subjecting the receive component to the second calibration comprises suspending transmit and receive operations of the circuitry of the receive component.	<u>Discussion</u> As I discussed above with respect to 1[c], Meaney discloses periodically recalibrating the receiver. Meaney further discloses that the first step of the recalibration process is putting the computer system or data transfer interface into a “wait state” to keep it from being used for anything other than recalibration. Ex. 1004, [0009]-[0015], [0041]-[0042], Fig. 3, claim 7. Additionally, Meaney discloses setting interface fences to block the interface from being used by any system operations other than recalibration. <i>Id.</i>, Abstract, [0037], [0044]. In this way, the system of Meaney suspends transit and receive operations of the receiver circuitry as part of the recalibration process. For these reasons, Meaney discloses “wherein periodically subjecting the receive component to the second calibration comprises suspending transmit and receive operations of the circuitry of the receive component.” <u>Other Relevant Citations</u> Meany at [0009]-[0015]: “The steps for re-calibration involve: 1. Putting the system of the interface into a wait state, 2. Performing a fast initialization process for calibration, and 3. Taking the system of the interface out of a wait state[.] The fast initialization process in the preferred embodiment is similar to the prior art, except that there is only one clock centering step, instead of two. Also, the data deskew step is skipped since that step has already been performed as part of the original initialization process. The step of putting the system into the wait state can be done through several methods as described in the referenced prior art. One way is to idle all the processors while having one processor oversee the recalibration sequence. This same processor may take the system out of the wait state as well. By keeping the interface from being used, the driving circuits can exclusively send the calibration pattern and the calibration logic can re-center the clock under any new environmental conditions or circuit changes.” <i>Id.</i> at [0037]: “Then, the SAP uses a hardware interface protocol to read and write registers in the interface logic directly. The SAP writes the Fence registers to make sure the interfaces are truly unused. Then, it loads the driver side calibration flag. This flag signals hardware to drive a repeating pattern across the interface. The pattern for the preferred embodiment is 100010001000” <i>Id.</i> at [0041]: “When the calibration is complete, the SAP turns off the driver calibration pattern, resets the interface fences and takes the processors out of the wait state.”

	<i>Id.</i> at [0042]-[0044]: “To summarize, the preferred method includes the steps of: 1. Putting the interface or system into a wait state, 2. Setting interface fences (block interface from being used by system operations) . . .” <i>Id.</i> at claim 7: “The method according to claim 6 wherein said wait state keeps the interface from being used for processing steps other than re-calibration and sending a calibration pattern and allowing claibration [sic] logic to re-center the clock applicable to the interface to compensate for new environmental conditions and circuit changes.” <i>Id.</i> at Fig. 3:
--	--

E.I. CLOCK RECEIVER STATE MACHINE DIAGRAM

7/89



Claim 9	
9[a]: The method of claim 8, wherein subjecting the receive component to the first calibration comprises receiving at the circuitry a first pattern transmitted by a transmit component and comparing, in the receive component, the first pattern with a first expected pattern, and	<u>Discussion</u> As I discussed above with respect to 1[c], Meany discloses an initial calibration of the system clock during initialization to determine a clock delay. Meany further discloses performing the initial calibration by sending a “known data pattern” across the interface to the receiver circuitry, which is used to calibrate the clock using calibration techniques. Ex. 1004, [0004], [0027]-[0031], Fig. 2. A POSITA would have understood that this discloses performing calibration by comparing, in the receive component, the received data pattern with the known data pattern to determine if they match, and if not, adjusting the clock delay. <u>Other Relevant Citations</u> Meany at [0004]: “In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.” <i>Id.</i> at [0024]-[0031]: “In the prior art, the system was always recalibrated from a stopped position. The system would be scan-initialized to a reset state and the calibration would begin when the clocks were started. The high-level sequence for the prior art calibration (described in referenced Docket AT998212, Dynamic Wave-pipelined Interface Apparatus and Methods) included the steps of: 1. Setting interface fences (block interface from being used by system operations) 2. Resetting interface state machine 3. Turning on the driver calibration pattern 4. Calibrating clock 5. Deskewing the data, one bit at a time 6. Recalibrating clock 7. Turning off driver calibration pattern” <i>Id.</i> at Fig. 2:

E.I. CLOCK RECEIVER STATE MACHINE DIAGRAM

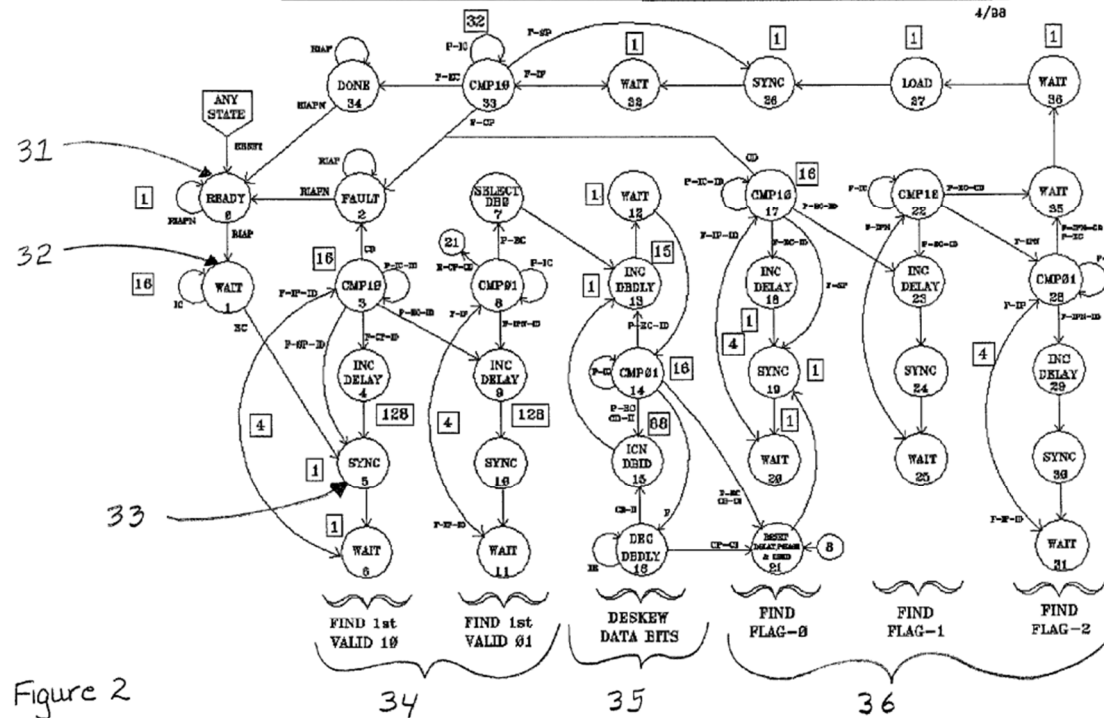


Figure 2

9[b]: wherein periodically subjecting the receive component to the second calibration comprises receiving at the circuitry a second pattern transmitted by the transmit component and comparing in the receive component the second pattern with a second expected pattern.	<u>Discussion</u> As I discussed above with respect to 1[c], Meaney discloses periodically recalibrating the receiver. As I discussed above with respect to 9[a], Meaney discloses initially calibrating a clock by transmitting a known data pattern and comparing the data pattern received by the receiver component with the known data pattern. Ex. 1004, [0004], [0027]-[0031], Fig. 3. Meaney further discloses that the same “known data pattern” techniques are used for recalibration (the claimed “second calibration”). Ex. 1004, [0037]-[0038], [0042]-[0048]. Meaney explains that recalibration includes turning on a known “driver calibration pattern” (e.g., a repeating bit pattern), which is used to re-calibrate the clock. <i>Id.</i> A POSITA would have understood that the re-calibration would be performed by comparing, in the receive component, the received data pattern with the known data pattern to determine if they match, and if not, to re-calibrate the clock. <u>Other Relevant Citations</u> Meany at [0004]: “In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.” <i>Id.</i> at [0024]-[0031]: “In the prior art, the system was always recalibrated from a stopped position. The system would be scan-initialized to a reset state and the calibration would begin when the clocks were started. The high-level sequence for the prior art calibration (described in referenced Docket AT998212, Dynamic Wave-pipelined Interface Apparatus and Methods) included the steps of: 1. Setting interface fences (block interface from being used by system operations) 2. Resetting interface state machine 3. Turning on the driver calibration pattern 4. Calibrating clock 5. Deskewing the data, one bit at a time 6. Recalibrating clock
---	---

	7. Turning off driver calibration pattern” <i>Id.</i> at [0037]: “Then, the SAP uses a hardware interface protocol to read and write registers in the interface logic directly. The SAP writes the Fence registers to make sure the interfaces are truly unused. Then, it loads the driver side calibration flag. This flag signals hardware to drive a repeating pattern across the interface. The pattern for the preferred embodiment is 100010001000” <i>Id.</i> at [0038]: “The SAP next signals the receiver calibration logic to perform the fast initialization procedure. Since the data deskew has already been done, it is not necessary to perform this lengthy step during recalibration. The fast initialization procedure consists of starting the state machine at the final clock calibration step.” <i>Id.</i> at [0042]-[0048]: “To summarize, the preferred method includes the steps of: 1. Putting the interface or system into a wait state, 2. Setting interface fences (block interface from being used by system operations) 3. Resetting interface state machine 4. Turning on the driver calibration pattern 5. Calibrating clock 6. Turning off driver calibration pattern”
--	--

Claim 10	
10[pre]: A system, comprising:	<u>Discussion</u> Meany discloses “[a]n SMP computer system has an apparatus and method for recalibrating a self-timed, source-synchronous, pipelined interface while the computer system is running.” Ex. 1004, Abstract, [0001]. <u>Other Relevant Citations</u> Meany at Abstract: “An SMP computer system has an apparatus and method for recalibrating a self-timed, source-synchronous, pipelined interface while the computer system is running. The apparatus allows for quiescing the interface (ie. idling the processors to allow for no data transfers), raising fences (blocking interfaces), allowing for a quick clock centering recalibration step, and then unfencing and unquiescing to allow for the use of the interface again. The recalibration allows for compensating for drift over time on the interface to compensate for temperature, voltage, cycle time, and end-of-life degradation without bringing down and restarting the system.”

	<i>Id.</i> at [0001]: “This invention relates to symmetrical computer systems, and particularly to an apparatus and method for recalibrating an source-synchronous pipelined interface with minimal impact to a running system, allowing for the system to remain operational, even in the midst of environmental drift or degradation.”
10[a]: a receive component, the receive component having circuitry to receive data communicated across a channel by a transmit component;	<u>Discussion</u> As I discussed above regarding claim 1[pre], Meaney discloses computer systems that include a receive component having circuitry to receive data. Ex. 1004, [0003]-[0004], [0007], [0015]-[0016], [0020]-[0021], [0023]. Meaney further discloses data communicated across a channel by a transmit component. Specifically, the computer systems of Meaney include the transfer of data across a source-synchronous pipelined interface, which a POSITA would have understood is a data channel. Ex. 1004, [0003]. Meaney explains that data is communicated “across the interface.” Ex. 1004, [0004]. For these reasons, a POSITA would have understood Meaney to disclose “a transmit component.” <u>Other Relevant Citations</u> Meaney at Abstract: “An SMP computer system has an apparatus and method for recalibrating a self-timed, source-synchronous, pipelined interface while the computer system is running. The apparatus allows for quiescing the interface (ie. idling the processors to allow for no data transfers), raising fences (blocking interfaces), allowing for a quick clock centering recalibration step, and then unfencing and unquiescing to allow for the use of the interface again. The recalibration allows for compensating for drift over time on the interface to compensate for temperature, voltage, cycle time, and end-of-life degradation without bringing down and restarting the system.” <i>Id.</i> at [0001]: “This invention relates to symmetrical computer systems, and particularly to an apparatus and method for recalibrating an [sic] source-synchronous pipelined interface with minimal impact to a running system, allowing for the system to remain operational, even in the midst of environmental drift or degradation.”

	<i>Id.</i> at [0003]-[0004]: “In an SMP (symmetric multiprocessing) computer system with high-level packaging, it is often necessary to allow latencies on cards, wires, or boards that exceed the cycle times of the transferred data. In order to allow these transfers to take place, source-synchronous pipelined interfaces have been used in the s/390 machines of IBM. This allows for the proper capturing of the data within a small window, or ‘eye’. In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.” <i>Id.</i> at [0007]: “Also, in a test environment, we often apply environmental stress to a machine as well as frequency changes. We often want to run a hardware/architecture exerciser (like SAK or PCX) under several environments (about 16 voltage corners as well as cycling up or down the frequency), the interface often fails without a re-calibration. If the system is not recalibrated under the new environment, the system must be restarted for every test. Therefore, our test time is jeopardized.” <i>Id.</i> at [0015]-[0016]: “By keeping the interface from being used, the driving circuits can exclusively send the calibration pattern and the calibration logic can re-center the clock under any new environmental conditions or circuit changes. The invention can be triggered periodically. This way, the changes to circuit or environmental characteristics over time do not adversely affect the operation of the interface.” <i>Id.</i> at [0020]-[0021]: “FIG. 2 illustrates the prior art state diagram for performing the initial calibration of the hardware as performed in conjunction with the prior art; FIG. 3 illustrates the preferred embodiment state diagrams for supporting the initial calibration as well as the recalibration of the hardware as performed in conjunction with the invention.” <i>Id.</i> at [0023]: “Turning to FIG. 1, notice that there is driver logic, 11, used to drive interface bus, 12, and clock, 13, which feeds receiver calibration logic, 14. Said receiver logic, 14, consisting of state machine logic, 15, clock calibrating hardware, 16, and data de-skew hardware 17. The output of receiver calibration logic, 14, transfers data onto receiver output bus, 18, which connects to system logic, 19.” <i>Id.</i> at Fig. 1:
--	---

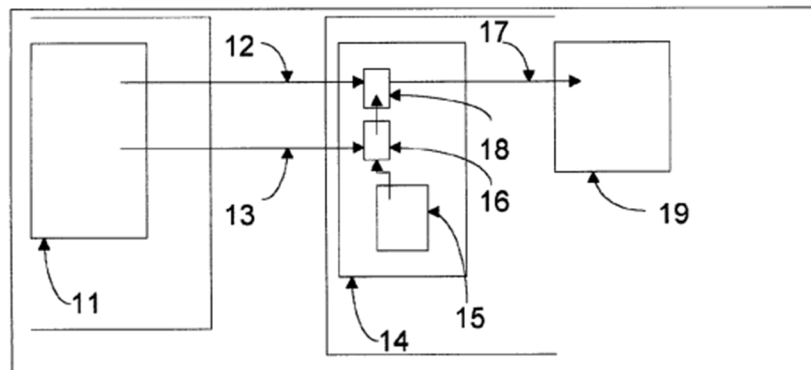


FIGURE 1.

Id. at Fig. 2:

E.I. CLOCK RECEIVER STATE MACHINE DIAGRAM

7/89

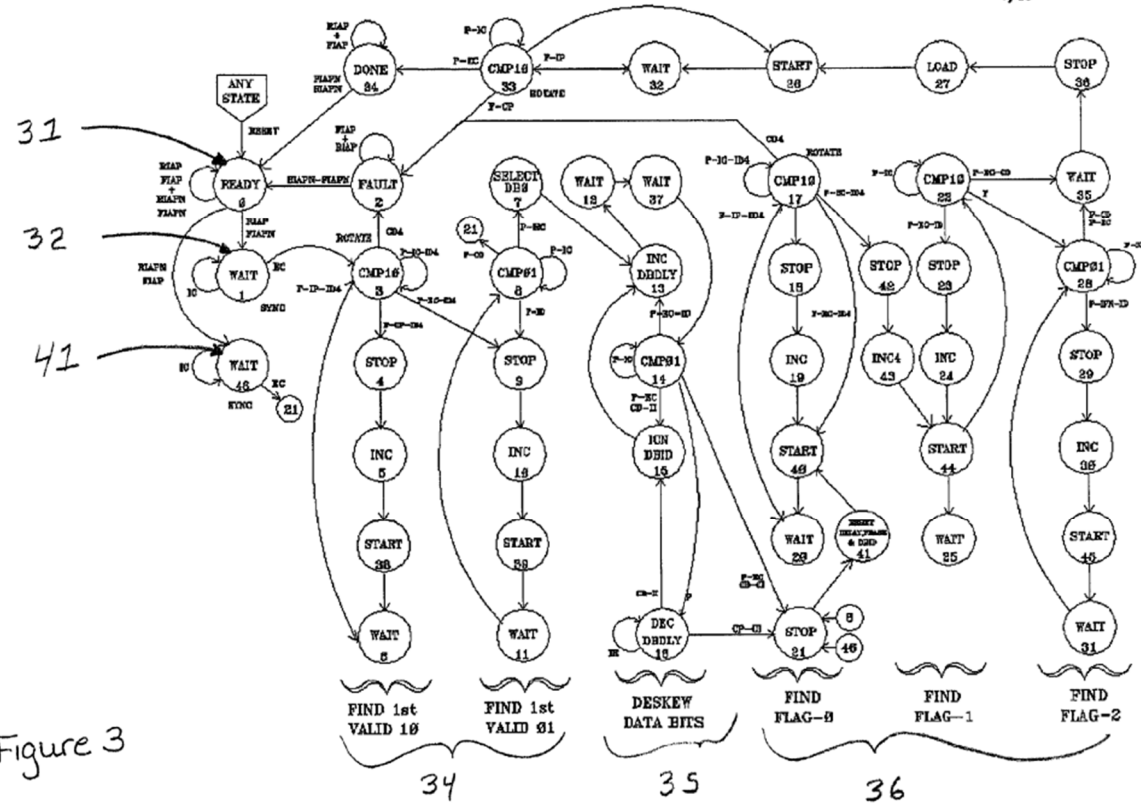


Figure 3

10[b]: the receive component to perform at system initialization a first calibration, the first calibration to identify an initial value for a parameter affecting proper reception by the circuitry of the data communicated across the channel, and	<u>Discussion</u> As I discussed above regarding 1[a] and 1[b], Meaney discloses subjecting the receive component to a first calibration during initialization to identify an initial value for a parameter affecting proper reception by the circuitry of the receive component of data communicated across a channel as part of the digital signal. For the same reasons, Meaney discloses “the receive component to perform at system initialization a first calibration, the first calibration to identify an initial value for a parameter affecting proper reception by the circuitry of the data communicated across the channel.” <u>Other Relevant Citations</u> Meany at [0003]: “In an SMP (symmetric multiprocessing) computer system with high-level packaging, it is often necessary to allow latencies on cards, wires, or boards that exceed the cycle times of the transferred data. In order to allow these transfers to take place, source-synchronous pipelined interfaces have been used in the s/390 machines of IBM. This allows for the proper capturing of the data within a small window, or ‘eye’.” <i>Id.</i> at [0004]: “In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.” <i>Id.</i> at [0006]: “The prior s/390 IBM machines only handled the first calibration. There was no way to re-calibrate an interface once it was up and running. Therefore, the cumulative degradations could eventually cause a system failure.” <i>Id.</i> at [0021]: “FIG. 3 illustrates the preferred embodiment state diagrams for supporting the initial calibration as well as the recalibration of the hardware as performed in conjunction with the invention.” <i>Id.</i> at [0024] : “In the prior art, the system was always recalibrated from a stopped position. The system would be scan-initialized to a reset state and the calibration would begin when the clocks were started.”²: “
---	--

² Although *Meaney* refers to this discussion as prior art, *Meaney* is clear that the preferred method is “similar to the prior art calibration sequence” but includes “additional steps.” Ex. 1004 at [0035]. Thus, *Meaney* is clear that the initial calibration of the clock would be part of the preferred method as well.

	<i>Id.</i> at [0038]: “The SAP next signals the receiver calibration logic to perform the fast initialization procedure. Since the data deskew has already been done, it is not necessary to perform this lengthy step during recalibration. The fast initialization procedure consists of starting the state machine at the final clock calibration step.” <i>Id.</i> at [0040]: “The clock delays are reset as part of the final clock calibration sequence. By adding delay to the clock path, the clock calibration logic is able to find the optimum data capture time for the interface. Since the calibration is done periodically, this window gets reoptimized every time recalibration occurs. Also, if the cycle time, voltage, or other changes are made to the environment for testing, the interface can recalibrate to the new conditions. However, the data deskew sequence does not need to be run since the deskew settings are not likely to change since the data nets will tend to track over time.” <i>Id.</i> at [0051]: “Although the preferred embodiment recalculates the flag0, flag1, and flag2 values, there are other methods that can be used to recalibrate the clock. For instance, if it is known that only the frequency changes, the change in frequency can be calculated by additional hardware and half the difference can be applied as a delay shift. Although this would be faster, we excluded this form of recalibration due to the complexity and the fact that the preferred embodiment was fast enough to avoid the system hangs that the full calibration would cause. Also, the preferred embodiment uses the same sequences used in the original.” <i>Id.</i> at claim 12: “The method according to claim 10 wherein the recalibration stem includes recalculating the clock frequency of the interface against the current hardware and re-applying the clock frequency calculation to the clock delay to re-center the clock when the machine is being cycled down to failure and the major change needing re-calibration is cycle time.”
--	---

10[c]: perform on a periodic basis a second calibration, the second calibration to update an existing value of the parameter for drift attributable to change in at least one of voltage or temperature,	<u>Discussion</u> As I discussed above regarding 1[c], Meaney discloses periodically subjecting the receive component to a second calibration to update an existing value of the parameter for drift attributable to change in at least one of operating voltage or temperature. For the same reasons, a POSITA with the benefit of Meaney would find it obvious to “perform on a periodic basis a second calibration, the second calibration to update an existing value of the parameter for drift attributable to change in at least one of voltage or temperature.” <u>Other Relevant Citations</u> Meany at [0008]: “The invention allows for the re-calibration of the interface at periodic intervals with minor disruption. This invention is applicable to the SMP computer systems sold by IBM and others.” <i>Id.</i> at [0009]-[0016]: “The steps for re-calibration involve: 1. Putting the system of the interface into a wait state, 2. Performing a fast initialization process for calibration, and 3. Taking the system of the interface out of a wait state[.] The fast initialization process in the preferred embodiment is similar to the prior art, except that there is only one clock centering step, instead of two. Also, the data deskew step is skipped since that step has already been performed as part of the original initialization process. The step of putting the system into the wait state can be done through several methods as described in the referenced prior art. One way is to idle all the processors while having one processor oversee the recalibration sequence. This same processor may take the system out of the wait state as well. By keeping the interface from being used, the driving circuits can exclusively send the calibration pattern and the calibration logic can re-center the clock under any new environmental conditions or circuit changes. The invention can be triggered periodically. This way, the changes to circuit or environmental characteristics over time do not adversely affect the operation of the interface.” <i>Id.</i> at [0040]: “The clock delays are reset as part of the final clock calibration sequence. By adding delay to the clock path, the clock calibration logic is able to find the optimum data capture time for the interface. Since the calibration is done periodically, this window gets reoptimized every time recalibration occurs. Also, if the cycle time, voltage, or other changes are made to the environment for testing, the interface can recalibrate to the new conditions. However, the data deskew sequence does not need to be run since the deskew settings are not likely to change since the data nets will tend to track over time.”
--	---

10[d]: wherein a time duration of the second calibration is constrained to be shorter than a time duration of the first calibration; and	<u>Discussion</u> As I discussed above regarding 1[e], Meaney discloses wherein the second calibration is constrained to occur during a time period that is shorter than a time period of the first calibration. For the same reasons, a POSITA with the benefit of Meaney would find this limitation obvious: “wherein a time duration of the second calibration is constrained to be shorter than a time duration of the first calibration.” <u>Other Relevant Citations</u> Meany at [0008]: “The invention allows for the re-calibration of the interface at periodic intervals with minor disruption. This invention is applicable to the SMP computer systems sold by IBM and others.” <i>Id.</i> at [0013]: “The fast initialization process in the preferred embodiment is similar to the prior art, except that there is only one clock centering step, instead of two. Also, the data deskew step is skipped since that step has already been performed as part of the original initialization process. <i>Id.</i> at [0024] : “In the prior art, the system was always recalibrated from a stopped position. The system would be scan-initialized to a reset state and the calibration would begin when the clocks were started.” <i>Id.</i> at [0035]: “The preferred method of recalibration is similar to the prior art calibration sequence. However, it includes the additional steps of putting the system into a wait state, running a modified recalibration, and restoring the system from a wait state to a running state.” <i>Id.</i> at [0038]: “The SAP next signals the receiver calibration logic to perform the fast initialization procedure. Since the data deskew has already been done, it is not necessary to perform this lengthy step during recalibration. The fast initialization procedure consists of starting the state machine at the final clock calibration step.” <i>Id.</i> at [0039]: “This recalibration can be understood by looking to FIG. 3. Notice that the normal calibration sequences of the initial clock calibration sequence, 34, data deskew sequence, 35, and final clock calibration sequence, 36, still exist. In addition, at the time of recalibration, a new wait state, 41, is used, which allows the calibration logic to proceed directly to the final clock calibration sequence, 36.” <i>Id.</i> at [0041]: “When the calibration is complete, the SAP turns off the driver calibration pattern, resets the interface fences and takes the processors out of the wait state.”
--	--

10[e]: circuitry to store the existing value of the parameter,	<u>Discussion</u> Meaney discloses that the computer system includes clock calibrating hardware. Ex. 1004, [0015]-[0016], [0020]-[0021], [0023]. Meaney further discloses a system assist processor (SAP) that uses a hardware interface protocol to read and write registers in the logic of the interface. Ex. 1004, [0037]. A POSITA would have understood that the clock calibrating hardware would include circuitry and would at least temporarily store the existing value of the clock delay in order to perform the recalibration. Additionally, a POSITA would have understood that the SAP would write the clock delay into a register in the interface logic. For these reasons, Meaney, in view of the knowledge of a POSITA, renders obvious “circuitry to store the existing value of the parameter.” Additionally, to the extent Patent Owner argues that Meaney does not disclose circuitry to store the existing clock delay parameter, it would have been obvious to a POSITA to modify Meaney to do so. Meaney has clock calibration hardware and discloses performing periodic recalibration of clock delay in a manner that depends on the existing clock delay. <i>Id.</i>; Ex. 1004, [0033], [0040], [0051]. A POSITA would have been motivated to store the existing clock delay locally in the computer system so that the existing clock delay would be readily accessible in performing the next clock recalibration. <u>Other Relevant Citations</u> Meany at [0015]: “By keeping the interface from being used, the driving circuits can exclusively send the calibration pattern and the calibration logic can re-center the clock under any new environmental conditions or circuit changes.” <i>Id.</i> at [0016]: “The invention can be triggered periodically. This way, the changes to circuit or environmental characteristics over time do not adversely affect the operation of the interface.” <i>Id.</i> at [0020]: “FIG. 2 illustrates the prior art state diagram for performing the initial calibration of the hardware as performed in conjunction with the prior art; while” <i>Id.</i> at [0021]: “FIG. 3 illustrates the preferred embodiment state diagrams for supporting the initial calibration as well as the recalibration of the hardware as performed in conjunction with the invention.” <i>Id.</i> at [0023]: “Turning to FIG. 1, notice that there is driver logic, 11, used to drive interface bus, 12, and clock, 13, which feeds receiver calibration logic, 14. Said receiver logic, 14, consisting of state machine logic, 15, clock calibrating hardware, 16, and data de-skew hardware 17. The output of receiver calibration logic, 14, transfers data onto receiver output bus, 18, which connects to system logic, 19.”
--	---

	<i>Id.</i> at [0033]: “The limitation with this sequence is that, once the interfaces are used, there is no means to get the interface recalibrated without resetting the state machine and starting the entire sequence over.” <i>Id.</i> at [0037]: “Then, the SAP uses a hardware interface protocol to read and write registers in the interface logic directly. The SAP writes the Fence registers to make sure the interfaces are truly unused. Then, it loads the driver side calibration flag. This flag signals hardware to drive a repeating pattern across the interface. The pattern for the preferred embodiment is 100010001000” <i>Id.</i> at [0040]: “The clock delays are reset as part of the final clock calibration sequence. By adding delay to the clock path, the clock calibration logic is able to find the optimum data capture time for the interface. Since the calibration is done periodically, this window gets reoptimized every time recalibration occurs. Also, if the cycle time, voltage, or other changes are made to the environment for testing, the interface can recalibrate to the new conditions. However, the data deskew sequence does not need to be run since the deskew settings are not likely to change since the data nets will tend to track over time.” <i>Id.</i> at [0051]: “Although the preferred embodiment recalculates the flag0, flag1, and flag2 values, there are other methods that can be used to recalibrate the clock. For instance, if it is known that only the frequency changes, the change in frequency can be calculated by additional hardware and half the difference can be applied as a delay shift. Although this would be faster, we excluded this form of recalibration due to the complexity and the fact that the preferred embodiment was fast enough to avoid the system hangs that the full calibration would cause. Also, the preferred embodiment uses the same sequences used in the original.”
10[f]: the existing value of the parameter dependent on the initial value and any updates from the second calibration.	<u>Discussion</u> As I discussed above regarding 1[d], Meaney discloses the existing value of clock delay is dependent on the initial clock delay value. For the same reasons, Meaney, in view of the knowledge of a POSITA, renders obvious “the existing value of the parameter dependent on the initial value.” And, as I discussed above in 1[c], Meaney discloses updating the clock delay based on the recalibration. <u>Other Relevant Citations</u> Meany at [0004]: “In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.”

	<i>Id.</i> at [0008]: “The invention allows for the re-calibration of the interface at periodic intervals with minor disruption. This invention is applicable to the SMP computer systems sold by IBM and others.” <i>Id.</i> at [0015]: “By keeping the interface from being used, the driving circuits can exclusively send the calibration pattern and the calibration logic can re-center the clock under any new environmental conditions or circuit changes.” <i>Id.</i> at [0016]: “The invention can be triggered periodically. This way, the changes to circuit or environmental characteristics over time do not adversely affect the operation of the interface.” <i>Id.</i> at [0033]: “The limitation with this sequence is that, once the interfaces are used, there is no means to get the interface recalibrated without resetting the state machine and starting the entire sequence over.” <i>Id.</i> at [0038]: “The SAP next signals the receiver calibration logic to perform the fast initialization procedure. Since the data deskew has already been done, it is not necessary to perform this lengthy step during recalibration. The fast initialization procedure consists of starting the state machine at the final clock calibration step.” <i>Id.</i> at [0039]: “This recalibration can be understood by looking to FIG. 3. Notice that the normal calibration sequences of the initial clock calibration sequence, 34, data deskew sequence, 35, and final clock calibration sequence, 36, still exist. In addition, at the time of recalibration, a new wait state, 41, is used, which allows the calibration logic to proceed directly to the final clock calibration sequence, 36.” <i>Id.</i> at [0040]: “The clock delays are reset as part of the final clock calibration sequence. By adding delay to the clock path, the clock calibration logic is able to find the optimum data capture time for the interface. Since the calibration is done periodically, this window gets reoptimized every time recalibration occurs. Also, if the cycle time, voltage, or other changes are made to the environment for testing, the interface can recalibrate to the new conditions. However, the data deskew sequence does not need to be run since the deskew settings are not likely to change since the data nets will tend to track over time.”
--	---

	<i>Id.</i> at [0051]: “Although the preferred embodiment recalculates the flag0, flag1, and flag2 values, there are other methods that can be used to recalibrate the clock. For instance, if it is known that only the frequency changes, the change in frequency can be calculated by additional hardware and half the difference can be applied as a delay shift. Although this would be faster, we excluded this form of recalibration due to the complexity and the fact that the preferred embodiment was fast enough to avoid the system hangs that the full calibration would cause. Also, the preferred embodiment uses the same sequences used in the original.”
--	---

Claim 11	
11: The system of claim 10, wherein the second calibration is to be performed at expiration of an interval of time, at which time the circuitry is to suspend receive operations.	<u>Discussion</u> For the same reasons I discussed above for claim 2, Meaney, in view of the knowledge of a POSITA, renders obvious “the second calibration is to be performed at expiration of an interval of time.” And for the same reasons I discussed above for claim 8, Meaney, in view of the knowledge of a POSITA, renders obvious performing the second calibration at expiration of an interval of time, “at which time the circuitry is to suspend receive operations.” Additionally, periodic recalibration based on intervals using timers were well known at the time of the ‘466 Patent. For example, U.S. Patent Publication No. 2002/0066001 (“Olarig”) discloses calibration timers that include time intervals for conducting a calibration. Olarig, Ex. 1010 at [0042]. When the timer of Olarig expires, the calibration is performed. <i>Id.</i> <u>Other Relevant Citations</u> Meany at [0008]: “The invention allows for the re-calibration of the interface at periodic intervals with minor disruption. This invention is applicable to the SMP computer systems sold by IBM and others.”

Id. at [0009]-[0016]: “The steps for re-calibration involve: 1. Putting the system of the interface into a wait state, 2. Performing a fast initialization process for calibration, and 3. Taking the system of the interface out of a wait state[.] The fast initialization process in the preferred embodiment is similar to the prior art, except that there is only one clock centering step, instead of two. Also, the data deskew step is skipped since that step has already been performed as part of the original initialization process. The step of putting the system into the wait state can be done through several methods as described in the referenced prior art. One way is to idle all the processors while having one processor oversee the recalibration sequence. This same processor may take the system out of the wait state as well. By keeping the interface from being used, the driving circuits can exclusively send the calibration pattern and the calibration logic can re-center the clock under any new environmental conditions or circuit changes. The invention can be triggered periodically. This way, the changes to circuit or environmental characteristics over time do not adversely affect the operation of the interface.”

Id. at [0037]: “Then, the SAP uses a hardware interface protocol to read and write registers in the interface logic directly. The SAP writes the Fence registers to make sure the interfaces are truly unused. Then, it loads the driver side calibration flag. This flag signals hardware to drive a repeating pattern across the interface. The pattern for the preferred embodiment is 100010001000”

Id. at [0041]: “When the calibration is complete, the SAP turns off the driver calibration pattern, resets the interface fences and takes the processors out of the wait state.”

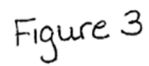
Id. at [0042]-[0044]: “To summarize, the preferred method includes the steps of: 1. Putting the interface or system into a wait state, 2. Setting interface fences (block interface from being used by system operations)”

Id. at claim 5: “In an SMP computer system having an source-synchronous interface, the method for re-calibration of the interface at periodic intervals comprising the steps of: a. putting the system of the interface into a wait state, b. performing a fast initialization process for calibration, c. taking the system of the interface out of said wait state.”

Id. at claim 7: “The method according to claim 6 wherein said wait state keeps the interface from being used for processing steps other than re-calibration and sending a calibration pattern and allowing claibration [sic] logic to re-center the clock applicable to the interface to compensate for new environmental conditions and circuit changes.”

Id. at Fig. 3:

7/89



Claim 12	
12: The system of claim 11, wherein passage of the interval of time is to be tracked by the transmit component and the second calibration is to be initiated by the transmit component.	<u>Discussion</u> As I discussed above for 10[a], Meaney discloses a transmit component. A POSITA would have understood that Meaney discloses the hardware of the transmit component tracking the periodic intervals and the second calibration being initiated by the transmit component. For example, Meaney discloses performing recalibration in “periodic intervals” and states that the recalibration method can be “triggered periodically.” Ex. 1004, [0008]. Meaney further discloses that the passage of the interval of time is tracked by the transmit component and the recalibration is initiated by the transmit component periodically. The recalibration of Meaney is initiated by a driver side calibration flag signaling hardware to drive a repeating pattern across the interface to the receiver. Ex. 1004, [0037]. To the extent Patent Owner argues that Meaney does not disclose the transmit component tracking the passage of the interval of time, it would have been obvious to a POSITA to modify Meaney to do so. A POSITA would have been motivated to have the transmit component track the periodic time intervals so it could initiate the recalibration directly without needing further instructions or signals (e.g., from the receiving component or other hardware). A POSITA would be further motivated to make this modification by Meaney’s disclosure that periodic recalibration allows the system to ensure that changes to the circuitry or environmental characteristics do not adversely affect performance over time, and discloses that the transmit component (the driver side hardware) initiates the recalibration. Ex. 1004, [0016], [0037]. <u>Other Relevant Citations</u> Meaney at Abstract: “An SMP computer system has an apparatus and method for recalibrating a self-timed, source-synchronous, pipelined interface while the computer system is running. The apparatus allows for quiescing the interface (ie. idling the processors to allow for no data transfers), raising fences (blocking interfaces), allowing for a quick clock centering recalibration step, and then unfencing and unquiescing to allow for the use of the interface again. The recalibration allows for compensating for drift over time on the interface to compensate for temperature, voltage, cycle time, and end-of-life degradation without bringing down and restarting the system.”

	<i>Id.</i> at [0001]: “This invention relates to symmetrical computer systems, and particularly to an apparatus and method for recalibrating an [sic] source-synchronous pipelined interface with minimal impact to a running system, allowing for the system to remain operational, even in the midst of environmental drift or degradation.” <i>Id.</i> at [0003]-[0004]: “In an SMP (symmetric multiprocessing) computer system with high-level packaging, it is often necessary to allow latencies on cards, wires, or boards that exceed the cycle times of the transferred data. In order to allow these transfers to take place, source-synchronous pipelined interfaces have been used in the s/390 machines of IBM. This allows for the proper capturing of the data within a small window, or ‘eye’. In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.” <i>Id.</i> at [0015]-[0016]: “By keeping the interface from being used, the driving circuits can exclusively send the calibration pattern and the calibration logic can re-center the clock under any new environmental conditions or circuit changes. The invention can be triggered periodically. This way, the changes to circuit or environmental characteristics over time do not adversely affect the operation of the interface.” <i>Id.</i> at [0023]: “Turning to FIG. 1, notice that there is driver logic, 11, used to drive interface bus, 12, and clock, 13, which feeds receiver calibration logic, 14. Said receiver logic, 14, consisting of state machine logic, 15, clock calibrating hardware, 16, and data de-skew hardware 17. The output of receiver calibration logic, 14, transfers data onto receiver output bus, 18, which connects to system logic, 19.” <i>Id.</i> at [0037]: “Then, the SAP uses a hardware interface protocol to read and write registers in the interface logic directly. The SAP writes the Fence registers to make sure the interfaces are truly unused. Then, it loads the driver side calibration flag. This flag signals hardware to drive a repeating pattern across the interface. The pattern for the preferred embodiment is 100010001000” <i>Id.</i> at Fig. 1:
--	--

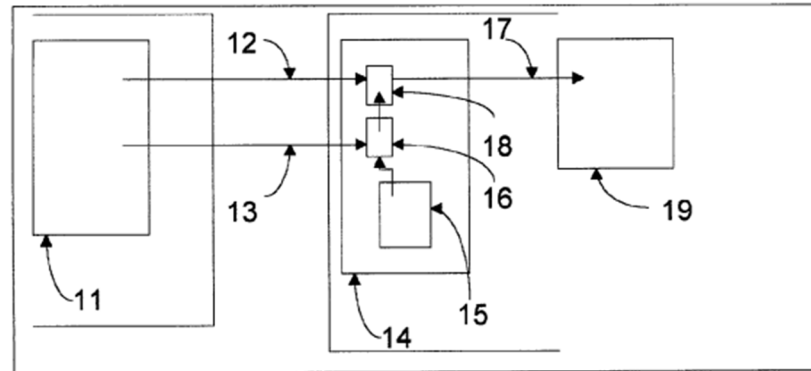


FIGURE 1.

Claim 15

15: The system of claim 10, wherein the receive component is to perform, responsive to the second calibration, one of an increment operation or a decrement operation upon the existing value if a desired value of the parameter has drifted from the existing value by more than a delta value.

Discussion

As I discussed above with respect to 1[d], Meaney discloses subjecting the receive component to a second calibration (Meaney's recalibration) to update an existing value of the parameter (clock delay) for drift.

Meaney further discloses that the receive component performs, responsive to the second calibration, one of an increment operation or a decrement operation upon the existing clock delay if a desired value of the clock delay has drifted from the existing clock delay by more than a delta value. Meaney discloses that recalibration may be performed in response to a trigger event that suggests that the clock delay has drifted and the data capture time window has changed and is no longer accurate. Ex. 1004, [0017].

Meaney explains that this trigger event can be indicated by, for example, when error correction is performed on the data and a correctable error is found. *Id.* Errors in data caused by clock delay drift would indicate that the clock delay has drifted by more than a delta value, i.e., by more than the amount that the system could tolerate without introducing errors into the data.

	In response, Meaney discloses that the clock delay may be adjusted by a “delay shift” (i.e., an increment or decrement operation) calculated based on a recalibration and then applied to the existing clock delay value. Ex. 1004, [0051]. For these reasons, a POSITA would have understood that Meaney, in view of the knowledge of a POSITA, renders obvious “the receive component is to perform, responsive to the second calibration, one of an increment operation or a decrement operation upon the existing value if a desired value of the parameter has drifted from the existing value by more than a delta value.” To the extent Patent Owner argues that Meaney does not disclose “perform[ing], responsive to the second calibration, one of an increment operation or a decrement operation upon the existing value if a desired value of the parameter has drifted from the existing value by more than a delta value,” this would have been obvious to a POSITA. It would have been obvious to a POSITA, in response to a recalibration, to perform an increment operation or a decrement operation on the existing clock delay value, to shift the value of the clock delay based on the results of the recalibration (e.g., based on the amount the received data pattern differs from the known data pattern). Ex. 1004, [0004], [0037]-[0038], [0042]-[0048]. A POSITA would have been motivated to do so because increment and decrement operations are well known operations in calibration techniques. A person of ordinary skill would also be motivated, by Meaney’s reference to the need to recalibrate clock delay to maintain accurate data capture despite drift to circuitry changes or changes in environmental conditions, and discloses recalibrating in response to a trigger event, Ex. 1004, Abstract, [0004]-[0005], as well as Meaney’s disclosure of changing the clock delay by applying a “delay shift” to the existing clock delay based on the results of the recalibration. <i>Id.</i> at [0051]. <u>Other Relevant Citations</u> Meany at [0004]: “In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.” <i>Id.</i> at [0005]: “However, over time, these large-latency interfaces can drift due to environmental changes like voltage, temperature, end-of-life degradation, etc.” <i>Id.</i> at [0008]: “The invention allows for the re-calibration of the interface at periodic intervals with minor disruption. This invention is applicable to the SMP computer systems sold by IBM and others.” <i>Id.</i> at [0016]: “The invention can be triggered periodically. This way, the changes to circuit or environmental characteristics over time do not adversely affect the operation of the interface.”
--	---

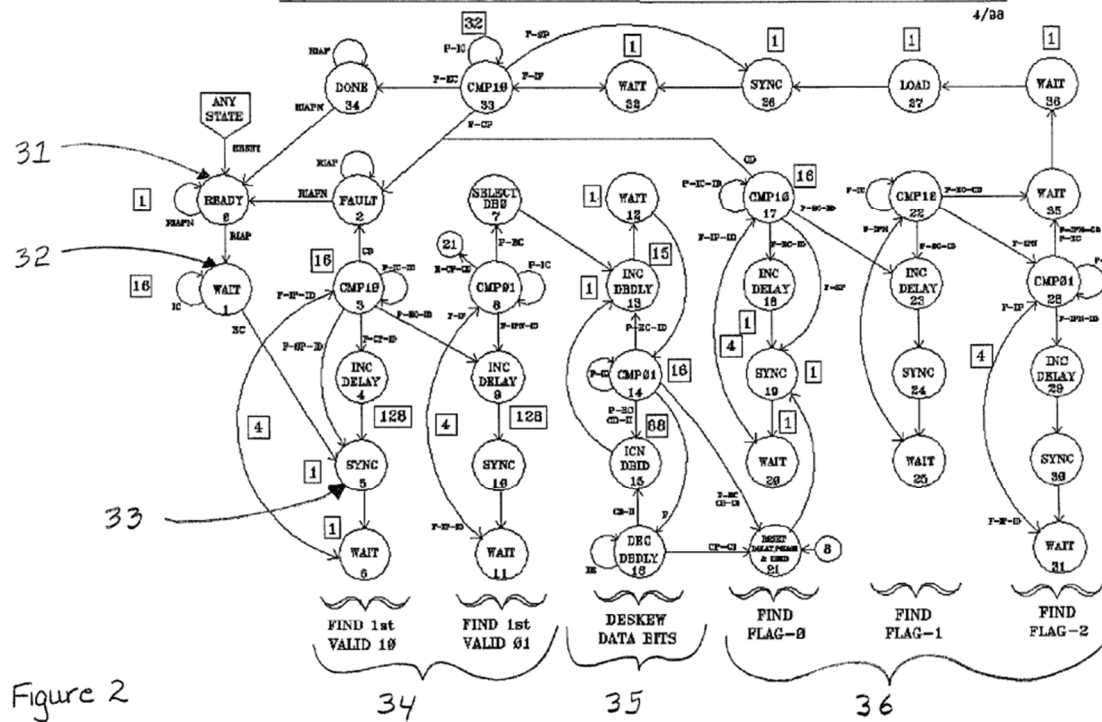
	<i>Id.</i> at [0017]: “Also, the preferred embodiment allows for the proactive running of the recalibration sequence based on a trigger event. For instance, when ECC is used on an interface and a correctable error (CE) is detected, the data is corrected. The CE can trigger an event to recalibrate the interface. This should protect against future CEs, if the original CE was based on a degraded interface.” <i>Id.</i> at [0033]: “The limitation with this sequence is that, once the interfaces are used, there is no means to get the interface recalibrated without resetting the state machine and starting the entire sequence over.” <i>Id.</i> at [0037]: “Then, the SAP uses a hardware interface protocol to read and write registers in the interface logic directly. The SAP writes the Fence registers to make sure the interfaces are truly unused. Then, it loads the driver side calibration flag. This flag signals hardware to drive a repeating pattern across the interface. The pattern for the preferred embodiment is 100010001000” <i>Id.</i> at [0038]: “The SAP next signals the receiver calibration logic to perform the fast initialization procedure. Since the data deskew has already been done, it is not necessary to perform this lengthy step during recalibration. The fast initialization procedure consists of starting the state machine at the final clock calibration step.” <i>Id.</i> at [0039]: “This recalibration can be understood by looking to FIG. 3. Notice that the normal calibration sequences of the initial clock calibration sequence, 34, data deskew sequence, 35, and final clock calibration sequence, 36, still exist. In addition, at the time of recalibration, a new wait state, 41, is used, which allows the calibration logic to proceed directly to the final clock calibration sequence, 36.” <i>Id.</i> at [0040]: “The clock delays are reset as part of the final clock calibration sequence. By adding delay to the clock path, the clock calibration logic is able to find the optimum data capture time for the interface. Since the calibration is done periodically, this window gets reoptimized every time recalibration occurs. Also, if the cycle time, voltage, or other changes are made to the environment for testing, the interface can recalibrate to the new conditions. However, the data deskew sequence does not need to be run since the deskew settings are not likely to change since the data nets will tend to track over time.”
--	--

	<i>Id.</i> at [0042]-[0048]: “To summarize, the preferred method includes the steps of: 1. Putting the interface or system into a wait state, 2. Setting interface fences (block interface from being used by system operations) 3. Resetting interface state machine 4. Turning on the driver calibration pattern 5. Calibrating clock 6. Turning off driver calibration pattern . . .” <i>Id.</i> at [0051]: “Although the preferred embodiment recalculates the flag0, flag1, and flag2 values, there are other methods that can be used to recalibrate the clock. For instance, if it is known that only the frequency changes, the change in frequency can be calculated by additional hardware and half the difference can be applied as a delay shift. Although this would be faster, we excluded this form of recalibration due to the complexity and the fact that the preferred embodiment was fast enough to avoid the system hangs that the full calibration would cause. Also, the preferred embodiment uses the same sequences used in the original.”
--	---

Claim 16	
16[a]: The system of claim 10, further comprising circuitry to store a first expected pattern and a second expected pattern, and	<u>Discussion</u> As I discussed above with respect to claim elements 9[a] and 9[b], Meaney discloses that both the initial calibration and the recalibration may be performed by comparing, in the receive circuitry, the received data pattern with the known data pattern to determine if they match, and if not, adjusting the clock delay. In order to carry out that comparison, the receiver circuitry must, at least temporarily, store the respective known data patterns needed for each calibration or recalibration, and thus those data patterns must be stored in circuitry of the computer system. Additionally, I note that because claim 16[a] does not specify where the circuitry must be, storage of those known data patterns anywhere in the circuitry of Meaney’s computer system is sufficient to meet this element. <u>Other Relevant Citations</u> Meaney at [0004]: “In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.”

	<i>Id.</i> at [0024]-[0031]: “In the prior art, the system was always recalibrated from a stopped position. The system would be scan-initialized to a reset state and the calibration would begin when the clocks were started. The high-level sequence for the prior art calibration (described in referenced Docket AT998212, Dynamic Wave-pipelined Interface Apparatus and Methods) included the steps of: 1. Setting interface fences (block interface from being used by system operations) 2. Resetting interface state machine 3. Turning on the driver calibration pattern 4. Calibrating clock 5. Deskewing the data, one bit at a time 6. Recalibrating clock 7. Turning off driver calibration pattern” <i>Id.</i> at [0037]: “Then, the SAP uses a hardware interface protocol to read and write registers in the interface logic directly. The SAP writes the Fence registers to make sure the interfaces are truly unused. Then, it loads the driver side calibration flag. This flag signals hardware to drive a repeating pattern across the interface. The pattern for the preferred embodiment is 100010001000” <i>Id.</i> at [0038]: “The SAP next signals the receiver calibration logic to perform the fast initialization procedure. Since the data deskew has already been done, it is not necessary to perform this lengthy step during recalibration. The fast initialization procedure consists of starting the state machine at the final clock calibration step.” <i>Id.</i> at [0042]-[0048]: “To summarize, the preferred method includes the steps of: 1. Putting the interface or system into a wait state, 2. Setting interface fences (block interface from being used by system operations) 3. Resetting interface state machine 4. Turning on the driver calibration pattern 5. Calibrating clock 6. Turning off driver calibration pattern” <i>Id.</i> at Fig. 2:
--	--

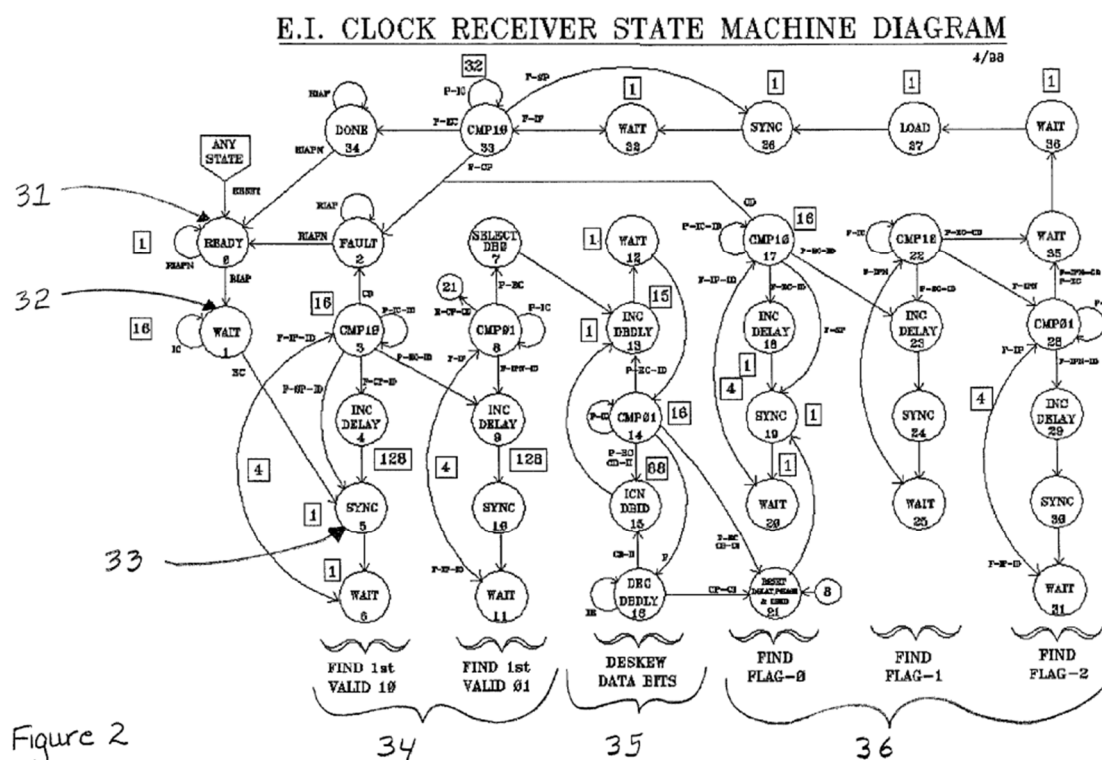
E.I. CLOCK RECEIVER STATE MACHINE DIAGRAM



16[b]: circuitry to compare the first expected pattern with a pattern received from the transmit component in association with the first calibration, and to compare the second expected pattern with a pattern received from the transmit component in association with the second calibration.	<u>Discussion</u> As I discussed above with respect to claim elements 9[a] and 9[b], Meaney discloses that both the initial calibration and the recalibration may be performed by comparing, in the receive circuitry, the received data pattern with the known data pattern to determine if they match, and if not, adjusting the clock delay. For the same reasons, Meaney, in view of the knowledge of a POSITA, renders obvious “circuitry to compare the first expected pattern with a pattern received from the transmit component in association with the first calibration, and to compare the second expected pattern with a pattern received from the transmit component in association with the second calibration.” <u>Other Relevant Citations</u> Meany at [0004]: “In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.” <i>Id.</i> at [0024]-[0031]: “In the prior art, the system was always recalibrated from a stopped position. The system would be scan-initialized to a reset state and the calibration would begin when the clocks were started. The high-level sequence for the prior art calibration (described in referenced Docket AT998212, Dynamic Wave-pipelined Interface Apparatus and Methods) included the steps of: 1. Setting interface fences (block interface from being used by system operations) 2. Resetting interface state machine 3. Turning on the driver calibration pattern 4. Calibrating clock 5. Deskewing the data, one bit at a time 6. Recalibrating clock 7. Turning off driver calibration pattern” <i>Id.</i> at [0037]: “Then, the SAP uses a hardware interface protocol to read and write registers in the interface logic directly. The SAP writes the Fence registers to make sure the interfaces are truly unused. Then, it loads the driver side calibration flag. This flag signals hardware to drive a repeating pattern across the interface. The pattern for the preferred embodiment is 100010001000”
--	---

Id. at [0042]-[0048]: “To summarize, the preferred method includes the steps of: 1. Putting the interface or system into a wait state, 2. Setting interface fences (block interface from being used by system operations) 3. Resetting interface state machine 4. Turning on the driver calibration pattern 5. Calibrating clock 6. Turning off driver calibration pattern . . .”

Id. at Fig. 2:



GROUND 2: Meaney and Nguyen in view of the knowledge of a POSITA.

A. Motivation to Combine Meaney and Nguyen

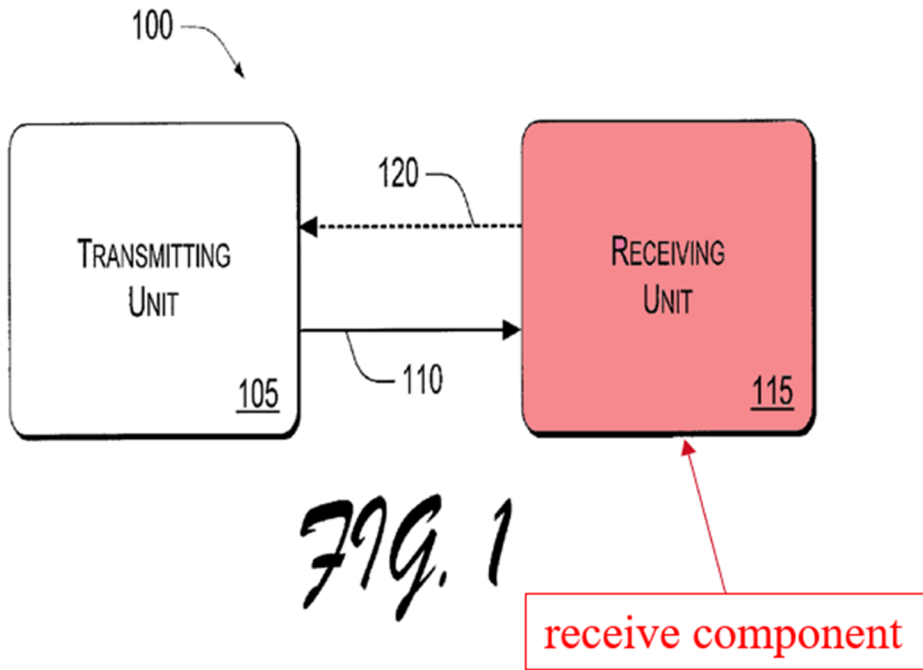
Meaney and Nguyen both pertain to electronic systems and methods for calibration and adjustment of communication system parameters. Meaney discloses a system for calibrating a parameter of a communication system, and Nguyen discloses techniques for applying offsets to existing values to determine calibrated values, as well as specific circuitry for signal reception and parameter calibration. Ex. 1004, Abstract, [0008]-[0017], [0039]-[0040]; Ex. 1005, [0065], [0083-0085], [0099].

A POSITA would have been motivated to modify the calibration techniques of Meaney with the offsets of Nguyen to achieve the benefit the enhancement of calibration accuracy. Nguyen describes a method of applying offsets to existing values to derive calibrated values, which ensures more precise adjustments in system parameters. Ex. 1005, [0099]. Incorporating this method into the system of Meaney would enhance the accuracy of Meaney's calibration process, thus improving overall system performance.

Moreover, a POSITA would have had a reasonable expectation of success in making this modification, at least due to similarities between the calibration techniques. For example, Meaney describes a concept similar to Nguyen's offsets: "adding delay" to the parameters or applying a shift to the existing parameters. Ex. 1004, [0040], [0051]. Alternatively, a POSITA would have understood that the calibration systems of Meaney would be enhanced with the offsets of Nguyen because this improvement represents the use of a known technique to predictably improve a similar system in the same way.

Additionally, Nguyen's disclosure of circuitry for receiving digital signals, calibrating computer system parameters, and storing calibration data offers a streamlined approach to system calibration. Ex. 1005, [0065], [0083-0085], [0099]. A POSITA would have been motivated to integrate this circuitry into the system of Meaney to make the calibration process becomes more efficient. Because Meaney does not specify the particular circuitry that is employed, incorporating the circuitry of Nguyen into the system of Meaney would, be choosing from a finite number of identified, predictable solutions, and a simple substitution of one known element for another to yield predictable results.

A POSITA would, have many reasons to expect to be able to successfully combine Meaney and Nguyen as described above. Both references describe similar digital communication systems and address common challenges in the field of electronic system calibration and optimization. Ex. 1004, Abstract, [0008]-[0017], [0039]-[0040]; Ex. 1005, [0065], [0083-0085], [0099]. The integration of Nguyen's calibration techniques and circuitry into Meaney's system would have, been obvious, straightforward and cost-effective. The required modifications would have been minimal, and the results of the combination predictable. Ex. 1005, 7:10-25.

Claim 1	
1 [pre]: A method of operation in a system that includes a receive component having circuitry to receive a digital signal, the method comprising:	<u>Discussion</u> As I discussed in 1[pre] of Ground 1, Meaney discloses a method of operation in a system that includes a receive component having a receiver and a receiver calibration logic for capturing a digital signal. Ex. 1004, [0003]-[0004], [0023]. Nguyen discloses a receive component having circuitry to receive a digital signal. For example, Nguyen discloses a receiving unit 115 that receives a signal 110 from the transmitting unit 110. Ex. 1005, [0030] Fig. 1 (annotated below).  The diagram illustrates a system 100 consisting of a transmitting unit 105 and a receiving unit 115. The transmitting unit 105 is represented by a white rounded rectangle, and the receiving unit 115 is represented by a red rounded rectangle. A solid arrow labeled 110 points from the transmitting unit 105 to the receiving unit 115. A dashed arrow labeled 120 points from the receiving unit 115 back to the transmitting unit 105. The receiving unit 115 is highlighted in red. Below the units, the text 'Fig. 1' is handwritten. A red box labeled 'receive component' has an arrow pointing to the receiving unit 115.

Nguyen discloses, for example, that the receiving unit may be part of an integrated circuit or printed circuit board, and discloses that the transmitted signal 110 may be a digital signal. Ex. 1005, [0031], [0032], [0047]; Nguyen also gives examples of specific circuitry that could be implemented in the receiving unit 115, including the circuitry of Fig. 4A, annotated below. Ex. 1005, Fig. 4A (annotated). As shown below, the receiving unit 115 can include a calibration component 415 and signal receivers 405A through 405N. *Id.* at [0047]-[0050].

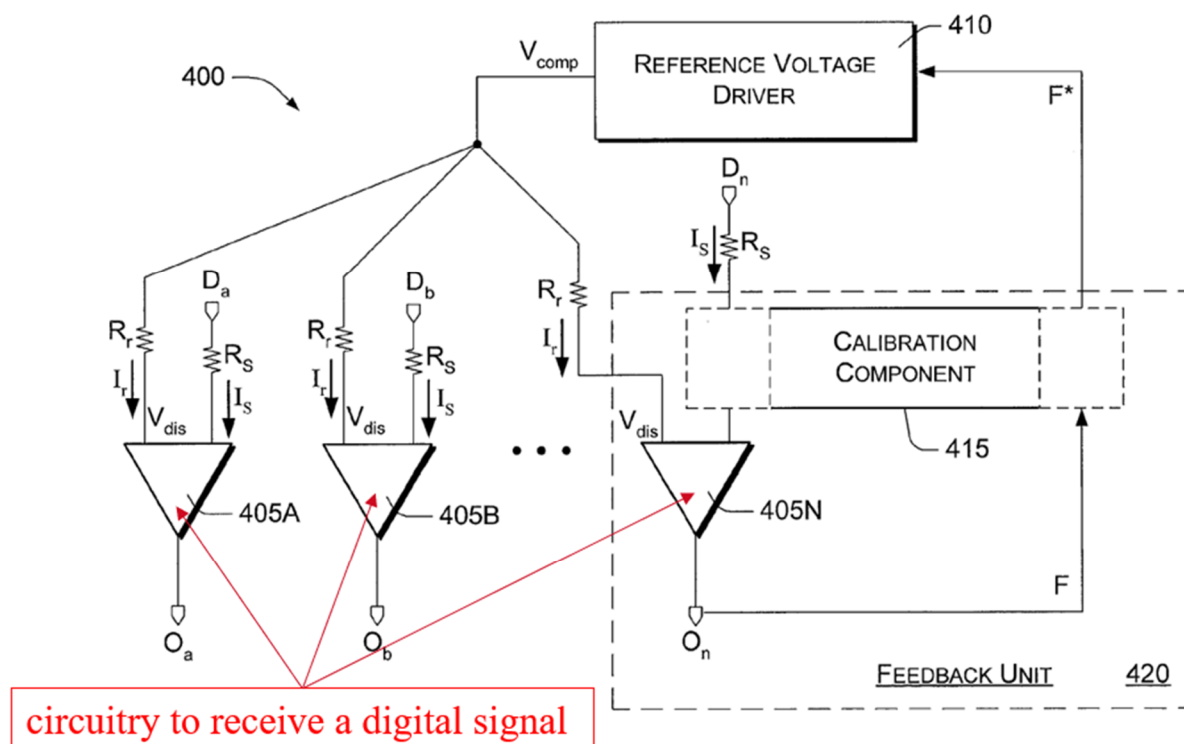


Fig. 4A

As I discussed above, it would have been obvious to a POSITA to enhance the computer system interface of Meaney with the receiver circuitry and digital signals of Nguyen. Because Meaney does not specify the particular receiver circuitry that is employed in receiving the transmitted data, it would have been obvious to a POSITA to modify Meaney to include the digital signal receiving unit of Nguyen to receive the digital signal of Meaney.

Other Relevant Citations

Meaney at Abstract: “An SMP computer system has an apparatus and method for recalibrating a self-timed, source-synchronous, pipelined interface while the computer system is running. The apparatus allows for quiescing the interface (ie. idling the processors to allow for no data transfers), raising fences (blocking interfaces), allowing for a quick clock centering recalibration step, and then unfencing and unquiescing to allow for the use of the interface again. The recalibration allows for compensating for drift over time on the interface to compensate for temperature, voltage, cycle time, and end-of-life degradation without bringing down and restarting the system.”

Id. at [0001]: “This invention relates to symmetrical computer systems, and particularly to an apparatus and method for recalibrating an [sic] source-synchronous pipelined interface with minimal impact to a running system, allowing for the system to remain operational, even in the midst of environmental drift or degradation.”

Id. at [0003]-[0004]: “In an SMP (symmetric multiprocessing) computer system with high-level packaging, it is often necessary to allow latencies on cards, wires, or boards that exceed the cycle times of the transferred data. In order to allow these transfers to take place, source-synchronous pipelined interfaces have been used in the s/390 machines of IBM. This allows for the proper capturing of the data within a small window, or ‘eye’. In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.”

Id. at [0007]: “Also, in a test environment, we often apply environmental stress to a machine as well as frequency changes. We often want to run a hardware/architecture exerciser (like SAK or PCX) under several environments (about 16 voltage corners as well as cycling up or down the frequency), the interface often fails without a re-calibration. If the system is not recalibrated under the new environment, the system must be restarted for every test. Therefore, our test time is jeopardized.”

Id. at [0015]-[0016]: “By keeping the interface from being used, the driving circuits can exclusively send the calibration pattern and the calibration logic can re-center the clock under any new environmental conditions or circuit changes. The invention can be triggered periodically. This way, the changes to circuit or environmental characteristics over time do not adversely affect the operation of the interface.”

Id. at [0020]-[0021]: “FIG. 2 illustrates the prior art state diagram for performing the initial calibration of the hardware as performed in conjunction with the prior art; FIG. 3 illustrates the preferred embodiment state diagrams for supporting the initial calibration as well as the recalibration of the hardware as performed in conjunction with the invention.”

Id. at [0023]: “Turning to FIG. 1, notice that there is driver logic, 11, used to drive interface bus, 12, and clock, 13, which feeds receiver calibration logic, 14. Said receiver logic, 14, consisting of state machine logic, 15, clock calibrating hardware, 16, and data de-skew hardware 17. The output of receiver calibration logic, 14, transfers data onto receiver output bus, 18, which connects to system logic, 19.”

Id. at Fig. 1:

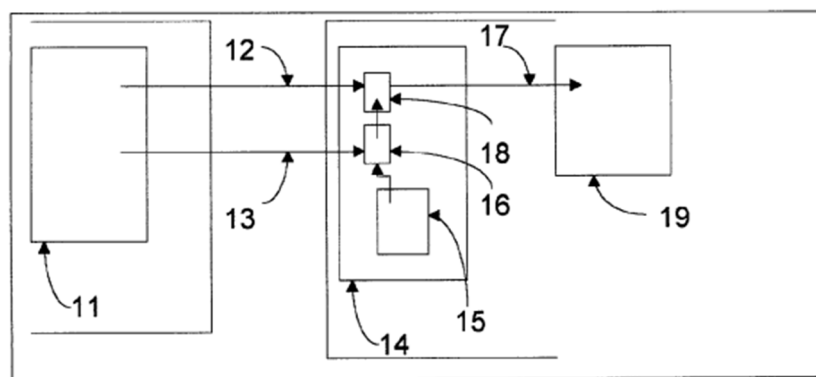


FIGURE 1.

Id. at Fig. 2:

E.I. CLOCK RECEIVER STATE MACHINE DIAGRAM

7/89

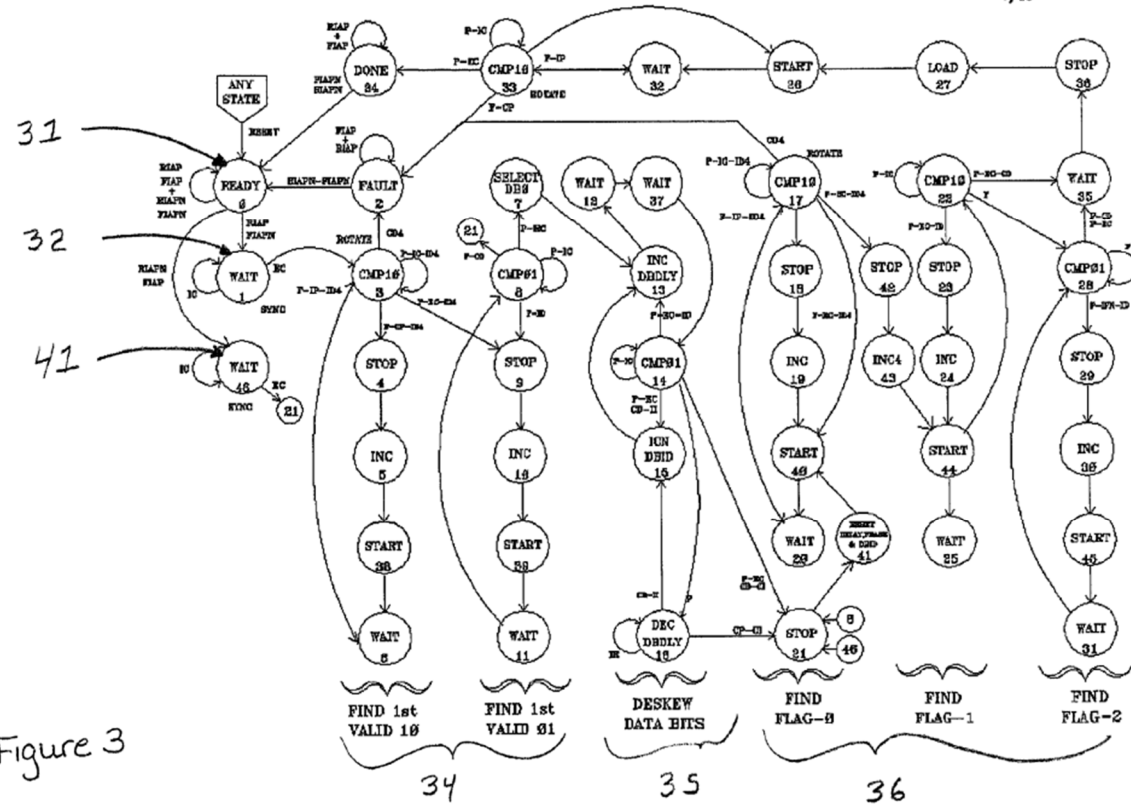


Figure 3

Nguyen discloses that the receiving unit may be part of an integrated circuit or printed circuit board, and teaches that the transmitted signal 110 may be a digital signal. Nguyen at [0031], [0032], [0047].

1[a]: subjecting the receive component to a first calibration during initialization to identify an initial value	<i>See</i> Ground 1, 1[a].
1[b]: for a parameter affecting proper reception by the circuitry of the receive component of data communicated across a channel as part of the digital signal;	<i>See</i> Ground 1, 1[b].
1[c]: periodically subjecting the receive component to a second calibration to update an existing value of the parameter for drift attributable to change in at least one of operating voltage or temperature; and	<i>See</i> Ground 1, 1[c].
1[d]: wherein the existing value is dependent on the initial value	<u>Discussion</u> As I discussed in Ground 1 at 1[d], Meaney discloses that an existing value for clock delay is dependent on an initial clock delay determined during initialization, either directly or indirectly. Ex. 1004, [0008], [0016], [0040]. Nguyen discloses adjusting a parameter of a digital signal receiver, compensated voltage, by applying an offset to a previously calibrated value. Ex. 1005, [0099] (“This calibrated value may be read from the register 710 and an offset value may be added. The resultant summed value may then be programmed back into the register 710 and used during normal operation of the device”). Specifically, Nguyen discloses determining a calibrated value, then adding or subtracting an offset value. Ex. 1005, [0099]. The offset value may be determined at design time or determined dynamically as part of an initialization or calibration procedure. Ex. 1005, [0099].

As I discussed above, it would have been obvious to a POSITA to enhance the calibration techniques of Meaney with the offset values of Nguyen. Meaney discloses recalibration techniques that include applying a delay shift to the existing clock delay. Ex. 1004, [0040], [0051]. A POSITA would have understood this as a similar concept to the offsets of Nguyen. In light of these similar disclosures, it would have been obvious to a POSITA to apply the offset values of Nguyen to the clock delay calibration of Meaney. For example, a POSITA could implement a predetermined offset to the clock delay of Meaney to compensate for an environmental difference (e.g., temperature) between the calibration environment and operational environment of the computer system interface.

Other Relevant Citations

Meany at [0004]: “In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.”

Id. at [0008]: “The invention allows for the re-calibration of the interface at periodic intervals with minor disruption. This invention is applicable to the SMP computer systems sold by IBM and others.”

Id. at [0016]: “The invention can be triggered periodically. This way, the changes to circuit or environmental characteristics over time do not adversely affect the operation of the interface.”

Id. at [0033]: “The limitation with this sequence is that, once the interfaces are used, there is no means to get the interface recalibrated without resetting the state machine and starting the entire sequence over.”

Id. at [0038]: “The SAP next signals the receiver calibration logic to perform the fast initialization procedure. Since the data deskew has already been done, it is not necessary to perform this lengthy step during recalibration. The fast initialization procedure consists of starting the state machine at the final clock calibration step.”

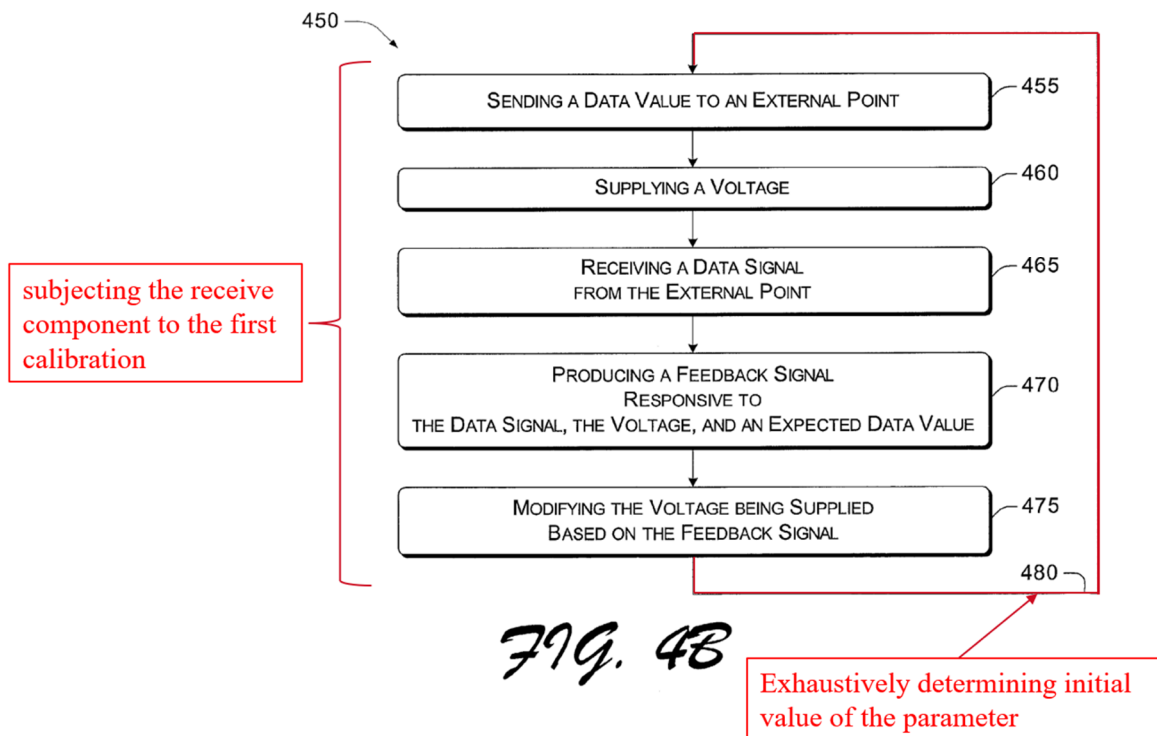
Id. at [0039]: “This recalibration can be understood by looking to FIG. 3. Notice that the normal calibration sequences of the initial clock calibration sequence, 34, data deskew sequence, 35, and final clock calibration sequence, 36, still exist. In addition, at the time of recalibration, a new wait state, 41, is used, which allows the calibration logic to proceed directly to the final clock calibration sequence, 36.”

	<i>Id.</i> at [0040]: “The clock delays are reset as part of the final clock calibration sequence. By adding delay to the clock path, the clock calibration logic is able to find the optimum data capture time for the interface. Since the calibration is done periodically, this window gets reoptimized every time recalibration occurs. Also, if the cycle time, voltage, or other changes are made to the environment for testing, the interface can recalibrate to the new conditions. However, the data deskew sequence does not need to be run since the deskew settings are not likely to change since the data nets will tend to track over time.” <i>Id.</i> at [0051]: “Although the preferred embodiment recalculates the flag0, flag1, and flag2 values, there are other methods that can be used to recalibrate the clock. For instance, if it is known that only the frequency changes, the change in frequency can be calculated by additional hardware and half the difference can be applied as a delay shift. Although this would be faster, we excluded this form of recalibration due to the complexity and the fact that the preferred embodiment was fast enough to avoid the system hangs that the full calibration would cause. Also, the preferred embodiment uses the same sequences used in the original.”
1[e]: and wherein the second calibration is constrained to occur during a time period that is shorter than a time period of the first calibration.	<u>Discussion</u> Meaney discloses that the recalibration is faster and has fewer steps than the initial calibration performed during the original initialization process. Ex. 1004, [0008], [0013], [0038]-[0039]. Specifically, Meaney discloses that recalibration is performed using a “fast initialization” process that can occur with minor disruption to the computer system by skipping certain steps (e.g., a data deskew and a clock centering step) that are performed during the original initialization process. Ex. 1004, [0008], [0013], [0038], Fig. 3. Moreover, the initial calibration of Meaney is performed from a stopped position, whereas the recalibration merely requires putting the system in a “wait” state for the duration of the recalibration. Ex. 1004, [0024], [0035], [0039], [0041]. For these reasons, a POSITA would have understood that the recalibration of Meaney would occur during a time period that is shorter than the time period of Meaney’s initial calibration. Thus, a POSITA would have understood that Meaney discloses “the second calibration is constrained to occur during a time period that is shorter than a time period of the first calibration.” <u>Other Relevant Citations</u>

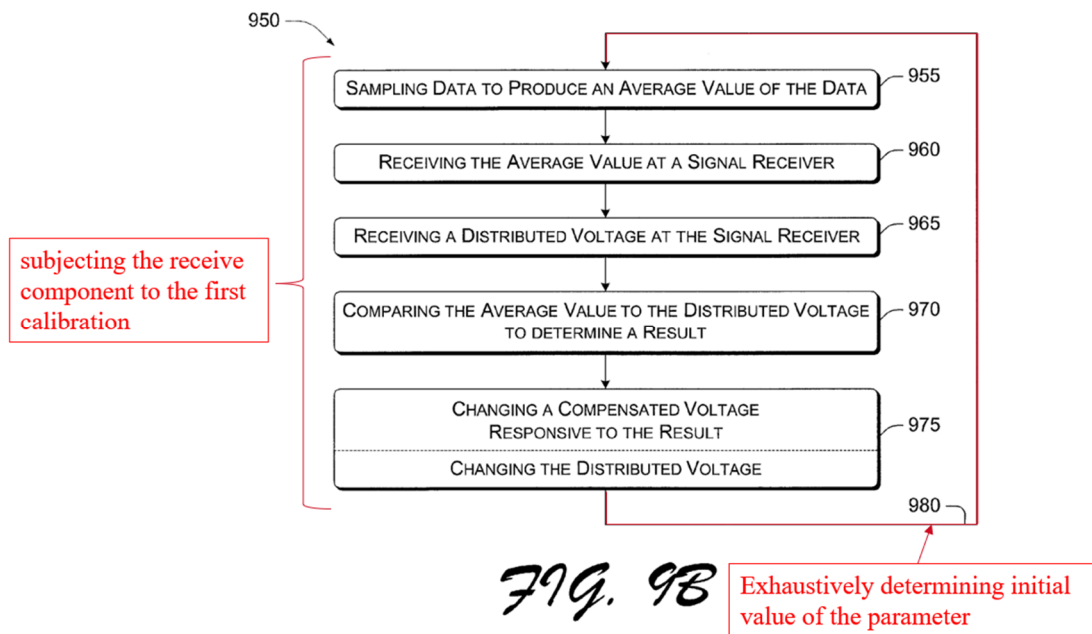
	Meany at [0008]: “The invention allows for the re-calibration of the interface at periodic intervals with minor disruption. This invention is applicable to the SMP computer systems sold by IBM and others.” <i>Id.</i> at [0013]: “The fast initialization process in the preferred embodiment is similar to the prior art, except that there is only one clock centering step, instead of two. Also, the data deskew step is skipped since that step has already been performed as part of the original initialization process.” <i>Id.</i> at [0024] : “In the prior art, the system was always recalibrated from a stopped position. The system would be scan-initialized to a reset state and the calibration would begin when the clocks were started.” <i>Id.</i> at [0035]: “The preferred method of recalibration is similar to the prior art calibration sequence. However, it includes the additional steps of putting the system into a wait state, running a modified recalibration, and restoring the system from a wait state to a running state.” <i>Id.</i> at [0038]: “The SAP next signals the receiver calibration logic to perform the fast initialization procedure. Since the data deskew has already been done, it is not necessary to perform this lengthy step during recalibration. The fast initialization procedure consists of starting the state machine at the final clock calibration step.” <i>Id.</i> at [0039]: “This recalibration can be understood by looking to FIG. 3. Notice that the normal calibration sequences of the initial clock calibration sequence, 34, data deskew sequence, 35, and final clock calibration sequence, 36, still exist. In addition, at the time of recalibration, a new wait state, 41, is used, which allows the calibration logic to proceed directly to the final clock calibration sequence, 36.” <i>Id.</i> at [0041]: “When the calibration is complete, the SAP turns off the driver calibration pattern, resets the interface fences and takes the processors out of the wait state.”
--	---

Claim 2	
2: The method of claim 1, wherein periodically subjecting includes calculating an interval and repeatedly (a) performing the second calibration at each expiration of the interval and (b) resetting the interval.	<i>See</i> Ground 1, claim 2
Claim 6	
6: The method of claim 1, wherein the second calibration uses less resources of the channel than the first calibration.	<i>See</i> Ground 1, claim 6
Claim 7	
7[a]: The method of claim 1, wherein subjecting the receive component to the first calibration comprises receiving a first set of operations at the circuitry to exhaustively determine the initial value of the parameter, and	<u>Discussion</u> As I discussed in Ground 1 at 7[a], Meaney discloses subjecting the circuitry of the receive component to a first calibration by calibrating clock delay during initialization of the computer system, which is triggered by sending a known data pattern across a data transfer interface, to exhaustively determine the initial value of clock delay. To the extent Patent Owner argues that Meaney does not disclose this element, Nguyen discloses the first calibration comprises receiving a first set of operations at the circuitry to exhaustively determine the initial value of the parameter. Ex. 1005, [0071]-[0074], [0110], Fig. 4B.

As shown in Fig. 4B, reproduced below, Nguyen discloses steps taken during an initialization or calibration phase, including repeating many of the steps of the initialization or calibration process until the voltage parameter reaches a satisfactory level. Ex. 1005, [0071]-[0074], Fig. 4B (annotated). Specifically, Nguyen explains that “arrow 480 leads from the action 475 back around to the action 455 to indicate that all or some of the actions 455-475 may be repeated during an initialization period, until the supplied voltage reaches a satisfactory level.” *Id.*, [0074].



Similarly, as shown in Fig. 9B, reproduced below, Nguyen discloses steps taken during an initialization or calibration phase, including repeating many of the steps of the initialization or calibration process 980 until the compensated voltage reaches a satisfactory level. Ex. 1005, [0107-0110], Fig. 9B (annotated):



A POSITA would have understood this disclosure of Nguyen to disclose “exhaustively determining the initial value” of the compensated voltage.

As I discussed above, it would have been obvious to a POSITA to enhance the calibration techniques of Meaney with the exhaustive techniques of Nguyen. Meaney discloses initialization calibration techniques that involve repeating the clock calibration step twice. Ex. 1004, [0004], [0024]-[0033], Fig. 3. In combination with Nguyen, a POSITA would have been motivated to repeat the clock calibration step of Meaney, as well as other initialization steps, until the clock delay recaches a satisfactory value. For example, performing these steps periodically not just during initialization but periodically to account for shifts in temperature or power supply voltage.

Other Relevant Citations

Meany at [0004]: “In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.”

	<i>Id.</i> at [0006]: “The prior s/390 IBM machines only handled the first calibration. There was no way to re-calibrate an interface once it was up and running. Therefore, the cumulative degradations could eventually cause a system failure.” <i>Id.</i> at [0021]: “FIG. 3 illustrates the preferred embodiment state diagrams for supporting the initial calibration as well as the recalibration of the hardware as performed in conjunction with the invention.” <i>Id.</i> at [0024]-[0032]: “In the prior art, the system was always recalibrated from a stopped position. The system would be scan-initialized to a reset state and the calibration would begin when the clocks were started. The high-level sequence for the prior art calibration (described in referenced Docket AT998212, Dynamic Wave-pipelined Interface Apparatus and Methods) included the steps of: 1. Setting interface fences (block interface from being used by system operations) 2. Resetting interface state machine 3. Turning on the driver calibration pattern 4. Calibrating clock 5. Deskewing the data, one bit at a time 6. Recalibrating clock 7. Turning off driver calibration pattern 8. Resetting interface fences” <i>Id.</i> at [0033]: “The limitation with this sequence is that, once the interfaces are used, there is no means to get the interface recalibrated without resetting the state machine and starting the entire sequence over.” <i>Id.</i> at [0038]: “The SAP next signals the receiver calibration logic to perform the fast initialization procedure. Since the data deskew has already been done, it is not necessary to perform this lengthy step during recalibration. The fast initialization procedure consists of starting the state machine at the final clock calibration step.”
--	---

Id. at [0040]: “The clock delays are reset as part of the final clock calibration sequence. By adding delay to the clock path, the clock calibration logic is able to find the optimum data capture time for the interface. Since the calibration is done periodically, this window gets reoptimized every time recalibration occurs. Also, if the cycle time, voltage, or other changes are made to the environment for testing, the interface can recalibrate to the new conditions. However, the data deskew sequence does not need to be run since the deskew settings are not likely to change since the data nets will tend to track over time.”

Id. at [0051]: “Although the preferred embodiment recalculates the flag0, flag1, and flag2 values, there are other methods that can be used to recalibrate the clock. For instance, if it is known that only the frequency changes, the change in frequency can be calculated by additional hardware and half the difference can be applied as a delay shift. Although this would be faster, we excluded this form of recalibration due to the complexity and the fact that the preferred embodiment was fast enough to avoid the system hangs that the full calibration would cause. Also, the preferred embodiment uses the same sequences used in the original.”

Id. at claim 12: “The method according to claim 10 wherein the recalibration stem includes re-calculating the clock frequency of the interface against the current hardware and re-applying the clock frequency calculation to the clock delay to re-center the clock when the machine is being cycled down to failure and the major change needing re-calibration is cycle time.”

Id. at Fig. 3:

E.I. CLOCK RECEIVER STATE MACHINE DIAGRAM

7/89

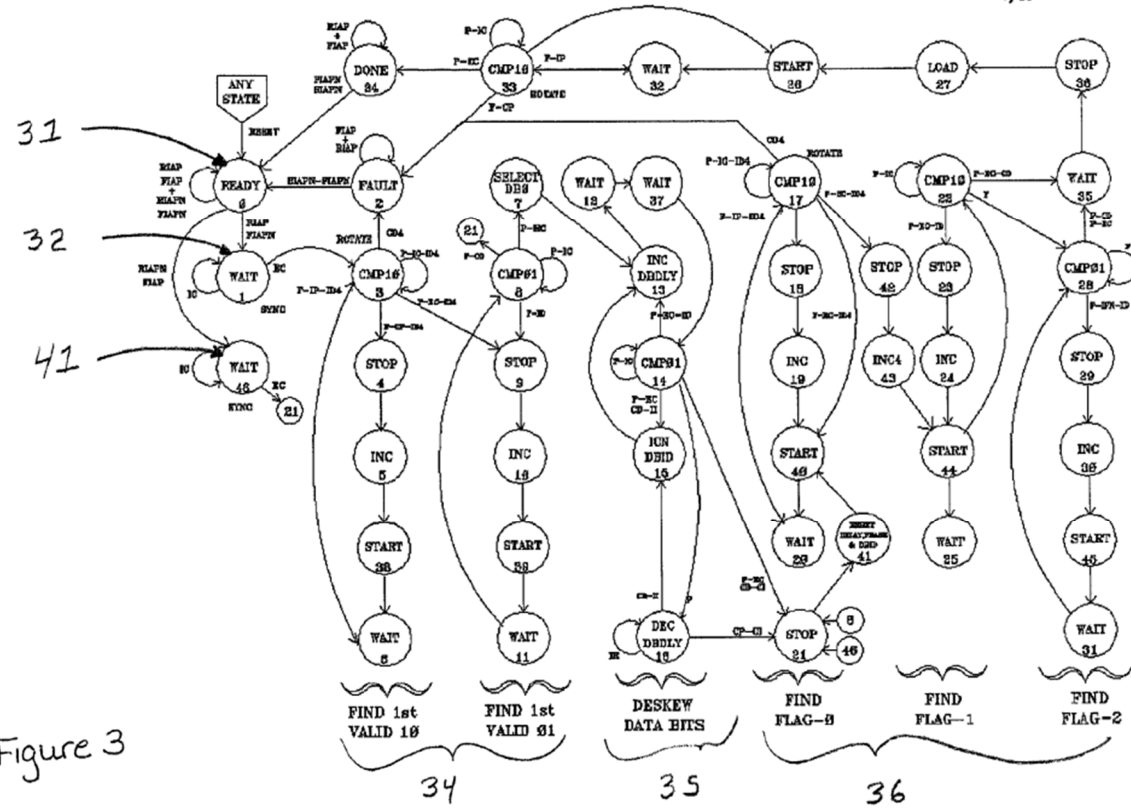


Figure 3

7[b]: wherein periodically subjecting the receive component to the second calibration comprises receiving a second set of operations at the circuitry to update the existing value if a desired value of the parameter has drifted from the existing value by more than a delta value.

Discussion

As I discussed in Ground 1 at 7[b], Meaney discloses receiving a second set of operations at the circuitry to update the existing value if a desired value of the parameter has drifted from the existing value by more than a delta value. To the extent Patent Owner argues that Meaney does not disclose this limitation, Nguyen also discloses that calibration may be repeated to correct for drifts in the calibrated parameter, or in response to “a predetermined number of detected (e.g., bit or byte) errors.” Ex. 1005, [0094], [0147]. A POSITA would have understood a predetermined number of detected errors to represent a delta value that represents a parameter drift. For these reasons, a POSITA would have understood that Nguyen discloses recalibration “if a desired value of the parameter has drifted from the existing value by more than a delta value.”

As I discussed above, it would have been obvious to a POSITA to enhance the recalibration triggers of Meaney with the predetermined number of errors of Nguyen. Meaney also discloses triggering recalibration when correctable errors are found in the data. Ex. 1004, [0017]. For these reasons, it would have been obvious for a POSITA to modify Meaney’s error-triggered recalibrations to include the “predetermined number of detected errors” of Nguyen to allow the system of Meaney to react to a desired level of error tolerance to, for example, account for errors due to changes in voltage and temperature.

Other Relevant Citations

Meany at Abstract: “An SMP computer system has an apparatus and method for recalibrating a self-timed, source-synchronous, pipelined interface while the computer system is running. The apparatus allows for quiescing the interface (ie. idling the processors to allow for no data transfers), raising fences (blocking interfaces), allowing for a quick clock centering recalibration step, and then unfencing and unquiescing to allow for the use of the interface again. The recalibration allows for compensating for drift over time on the interface to compensate for temperature, voltage, cycle time, and end-of-life degradation without bringing down and restarting the system.”

Id. at [0001]: “This invention relates to symmetrical computer systems, and particularly to an apparatus and method for recalibrating an source-synchronous pipelined interface with minimal impact to a running system, allowing for the system to remain operational, even in the midst of environmental drift or degradation.”

	<i>Id.</i> at [0004]: “In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.” <i>Id.</i> at [0005]: “However, over time, these large-latency interfaces can drift due to environmental changes like voltage, temperature, end-of-life degradation, etc.” <i>Id.</i> at [0017]: “Also, the preferred embodiment allows for the proactive running of the recalibration sequence based on a trigger event. For instance, when ECC is used on an interface and a correctable error (CE) is detected, the data is corrected. The CE can trigger an event to recalibrate the interface. This should protect against future CEs, if the original CE was based on a degraded interface.”
--	---

Claim 8	
8: The method of claim 1, wherein periodically subjecting the receive component to the second calibration comprises suspending transmit and receive operations of the circuitry of the receive component.	<i>See</i> Ground 1, claim 8

Claim 9	
9[a]: The method of claim 8, wherein subjecting the receive component to the first calibration comprises receiving at the circuitry a first pattern transmitted by a transmit component and comparing, in the receive component, the first pattern with a first expected pattern, and	<i>See</i> Ground 1, claim 9[a]
9[b]: wherein periodically subjecting the receive component to the second calibration comprises receiving at the circuitry a second pattern transmitted by the transmit component and comparing in the receive component the second pattern with a second expected pattern.	<i>See</i> Ground 1, claim 9[b]
Claim 10	
10[pre]: A system, comprising:	<i>See</i> Ground 1, claim 10[pre].

10[a]: a receive component, the receive component having circuitry to receive data communicated across a channel by a transmit component;	As I discussed above in Ground 1 at 10[a], Meaney discloses a receive component having circuitry to receive data communicated across a channel by a transmit component. As I discussed above in this Ground at 1[pre], the combination of Meaney and Nguyen, in view of the knowledge of a POSITA, renders obvious “the receive component having circuitry to receive data.”
10[b]: the receive component to perform at system initialization a first calibration, the first calibration to identify an initial value for a parameter affecting proper reception by the circuitry of the data communicated across the channel, and	<i>See</i> Ground 1, claim 10[b].
10[c]: perform on a periodic basis a second calibration, the second calibration to update an existing value of the parameter for drift attributable to change in at least one of voltage or temperature,	<i>See</i> Ground 1, claim 10[c].

10[d]: wherein a time duration of the second calibration is constrained to be shorter than a time duration of the first calibration; and	See Ground 1, claim 10[d].
10[e]: circuitry to store the existing value of the parameter,	<u>Discussion</u> As I discussed in Ground 1, Meaney discloses that the computer system includes clock calibrating hardware including an SAP that uses a hardware interface protocol to read and write registers in the logic of the interface. Ex. 1004, [0015]-[0016], [0020]-[0021], [0023]. Nguyen also discloses “circuitry to store the existing value of the parameter.” For example, Nguyen discloses the use of a “parameters table” and a parameters table bus during initialization/calibration to store calibrated parameter values. Ex. 1005, [0130-0133]. As I discussed in the motivation to combine above, it would have been obvious to a POSITA to enhance the computer system interface of Meaney with the calibration techniques and receiver circuitry of Nguyen. Moreover, it would have been obvious for a POSITA to modify Meaney’s system to include the parameter table and parameter table bus of Nguyen to allow the system of Meaney to more easily and reliably store and retrieve clock delay values from calibrations and recalibrations. <u>Other Relevant Citations</u> Meany at [0015]: “By keeping the interface from being used, the driving circuits can exclusively send the calibration pattern and the calibration logic can re-center the clock under any new environmental conditions or circuit changes.” <i>Id.</i> at [0016]: “The invention can be triggered periodically. This way, the changes to circuit or environmental characteristics over time do not adversely affect the operation of the interface.” <i>Id.</i> at [0020]: “FIG. 2 illustrates the prior art state diagram for performing the initial calibration of the hardware as performed in conjunction with the prior art; while” <i>Id.</i> at [0021]: “FIG. 3 illustrates the preferred embodiment state diagrams for supporting the initial calibration as well as the recalibration of the hardware as performed in conjunction with the invention.”

	<i>Id.</i> at [0023]: “Turning to FIG. 1, notice that there is driver logic, 11, used to drive interface bus, 12, and clock, 13, which feeds receiver calibration logic, 14. Said receiver logic, 14, consisting of state machine logic, 15, clock calibrating hardware, 16, and data de-skew hardware 17. The output of receiver calibration logic, 14, transfers data onto receiver output bus, 18, which connects to system logic, 19.” <i>Id.</i> at [0033]: “The limitation with this sequence is that, once the interfaces are used, there is no means to get the interface recalibrated without resetting the state machine and starting the entire sequence over.” <i>Id.</i> at [0037]: “Then, the SAP uses a hardware interface protocol to read and write registers in the interface logic directly. The SAP writes the Fence registers to make sure the interfaces are truly unused. Then, it loads the driver side calibration flag. This flag signals hardware to drive a repeating pattern across the interface. The pattern for the preferred embodiment is 100010001000” <i>Id.</i> at [0040]: “The clock delays are reset as part of the final clock calibration sequence. By adding delay to the clock path, the clock calibration logic is able to find the optimum data capture time for the interface. Since the calibration is done periodically, this window gets reoptimized every time recalibration occurs. Also, if the cycle time, voltage, or other changes are made to the environment for testing, the interface can recalibrate to the new conditions. However, the data deskew sequence does not need to be run since the deskew settings are not likely to change since the data nets will tend to track over time.” <i>Id.</i> at [0051]: “Although the preferred embodiment recalculates the flag0, flag1, and flag2 values, there are other methods that can be used to recalibrate the clock. For instance, if it is known that only the frequency changes, the change in frequency can be calculated by additional hardware and half the difference can be applied as a delay shift. Although this would be faster, we excluded this form of recalibration due to the complexity and the fact that the preferred embodiment was fast enough to avoid the system hangs that the full calibration would cause. Also, the preferred embodiment uses the same sequences used in the original.” Nguyen at [0130-0133].
--	---

10[f]: the existing value of the parameter dependent on the initial value and any updates from the second calibration.	<u>Discussion</u> As I discussed in Ground 1 at 10[f], Meaney discloses “the existing value of the parameter dependent on . . . any updates from the second calibration” by updating clock delay based on a recalibration. As I discussed above in this Ground 2, Meaney and Nguyen, in view of the knowledge of a POSITA, renders obvious “the existing value of the parameter dependent on the initial value.”
---	---

Claim 11	
11: The system of claim 10, wherein the second calibration is to be performed at expiration of an interval of time, at which time the circuitry is to suspend receive operations.	<i>See</i> Ground 1, claim 11.

Claim 12	
12: The system of claim 11, wherein passage of the interval of time is to be tracked by the transmit component and the second calibration is to be initiated by the transmit component.	<i>See</i> Ground 1, claim 12.

Claim 15	
15: The system of claim 10, wherein the receive component is to perform, responsive to the second calibration, one of an increment operation or a decrement operation upon the existing value if a desired value of the parameter has drifted from the existing value by more than a delta value.	<u>Discussion</u> As I discussed in Ground 1 at claim 15, Meaney teaches recalibration may be performed in response to a trigger event that suggests that the clock delay has drifted and the data capture time window is no longer accurate. As I discussed in this Ground regarding 1[d] and 7[b], the combination of Meaney and Nguyen teaches applying an increment or decrement operation by adjusting the clock delay through by a predetermined offset in response to drift by more than a delta value (e.g., a predetermined number of detected errors).

Claim 16	
16[a]: The system of claim 10, further comprising circuitry to store a first expected pattern and a second expected pattern, and 16[b]: circuitry to compare the first expected pattern with a pattern received from the transmit component in association with the first calibration, and to compare the second expected pattern with a pattern received from the transmit component in association with the second calibration.	<u>Discussion</u> As I discussed in Ground 1 at claim 16, Meaney teaches circuitry to store a first expected pattern and a second expected pattern, as well as comparing first expected patterns and second expected patterns to data patterns received from a transmit component in association with first and second calibrations. To the extent Patent Owner argues that Meaney does not teach these limitations, Nguyen discloses first and second expected patterns and circuitry for receiving, storing, and providing the expected data, and circuitry for comparing same. Ex. 1005, [0112], [0114], Fig. 10A. Specifically, as shown in annotated Fig. 10A below, Nguyen discloses a “digital calibration component 510” that has interfaces for “receiving and/or providing calibration data” including a “calibration data register 1005” which stores the calibration data. Ex. 1005, [0112], [0114]. Nguyen explains that the calibration data includes “expected value[s] to be received from a transmitting unit” which is compared against received data to determine if they are equal in the comparison unit 1010. <i>Id.</i>, [0114]. Accordingly, the calibration data register 1005 is circuitry to store a first expected pattern and a second expected pattern and the comparison unit 1010 is circuitry to compare the first expected pattern with a pattern received from the transmit component in association with the first calibration, and to compare the second expected pattern with a pattern received from the transmit component in association with the second calibration.

GROUND 3: Meaney and Greeff in view of the knowledge of a POSITA.

A. Motivation to Combine

A POSITA would have been motivated to combine Meaney and Greeff to arrive at the alleged inventions described in claims 4, 5, 13, 14, and 17-19. For example, a POSITA would have been motivated to incorporate the termination circuitry disclosed in Greeff into the system of Meaney to achieve enhanced system performance and reliability. Greeff's termination circuitry ensures proper signal integrity and reduces reflections in high-speed data transmission systems. Ex. 1006, 1:28-31, 4:10-12, 8:3-5. Meaney's calibration techniques, when applied to the termination resistance and transistor drive strength of Greeff, would ensure that these parameters remain optimal throughout system operation.

Moreover, both Meaney and Greeff disclose methods for calibrating variables initially and during operation for recalibration. Ex. 1004, Ex. 1004, [0006], [0008], [0013], [0021], [0024], [0038]; Ex. 1006, 4:11-20, 7:52-8:56. A POSITA would have found it obvious to apply Meaney's calibration process to Greeff's termination circuitry in order to optimize the termination resistance and drive strength of the termination circuitry.

A POSITA would have had a reasonable expectation of success in combining Meaney and Greeff. The integration of Greeff's termination circuitry into Meaney's system would have been straightforward and cost-effective, and the required modifications of such a combination minimal—and the results of the combination predictable—given the complementary nature of the technologies. For example, Meaney and Greeff both pertain to electronic systems and methods for calibration and adjustment of system parameters. Meaney discloses a system for calibrating a parameter of an electronic system, while Greeff discloses termination circuitry that are integral to electronic system performance. Ex. 1004, Abstract, [0008]-[0017], [0039]-[0040]; Ex. 1006, 3:65-4:20, 4:11-20, 5:59-6:6, 7:52-8:56. Both references emphasize the importance of calibrating variables both initially, prior to operation, and during operation. Ex. 1004, Ex. 1004, [0006], [0008], [0013], [0021], [0024], [0038]; Ex. 1006, 4:11-20, 7:52-8:56.

Claim 4

4: The method of claim 1, wherein the parameter is a link termination resistance.

Discussion

As shown below in annotated Fig. 1, Greeff discloses a multidrop bus including circuitry capable of receiving a digital signal that includes termination circuitry 120 including a switch 122 connected between a “trimmable” termination resistor 124, having a trimmable termination resistance R_{TERM} , and a reference voltage V_{TERM} . Ex. 1006, 3:65-4:20, 5:59-6:6, 7:52-8:56, Fig. 1. A POSITA would have understood Greeff’s termination resistance to be the claimed “link termination resistance.” Greeff explains that the termination resistance is modified during initial calibration and recalibration processes to provide a substantially optimal termination of the bus 102. Ex. 1006, 4:11-20, 7:52-8:56.

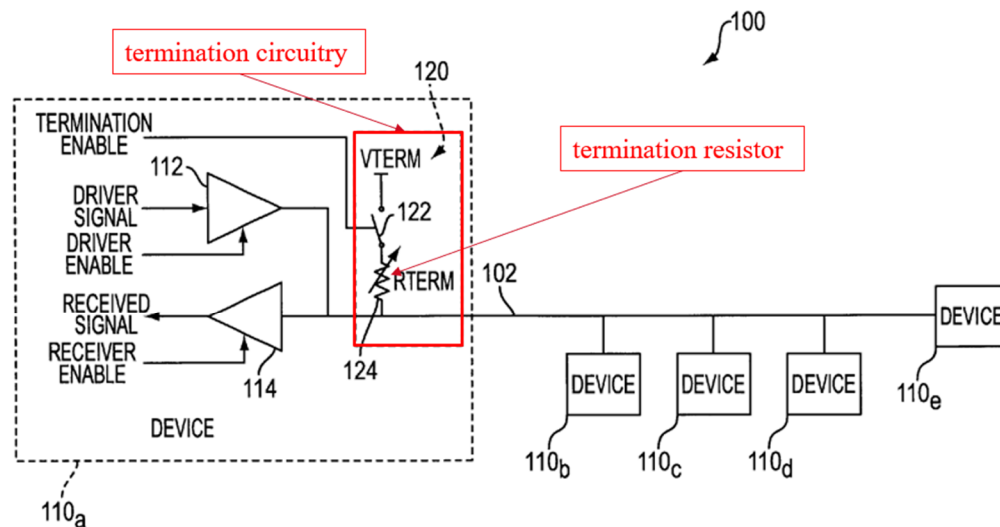
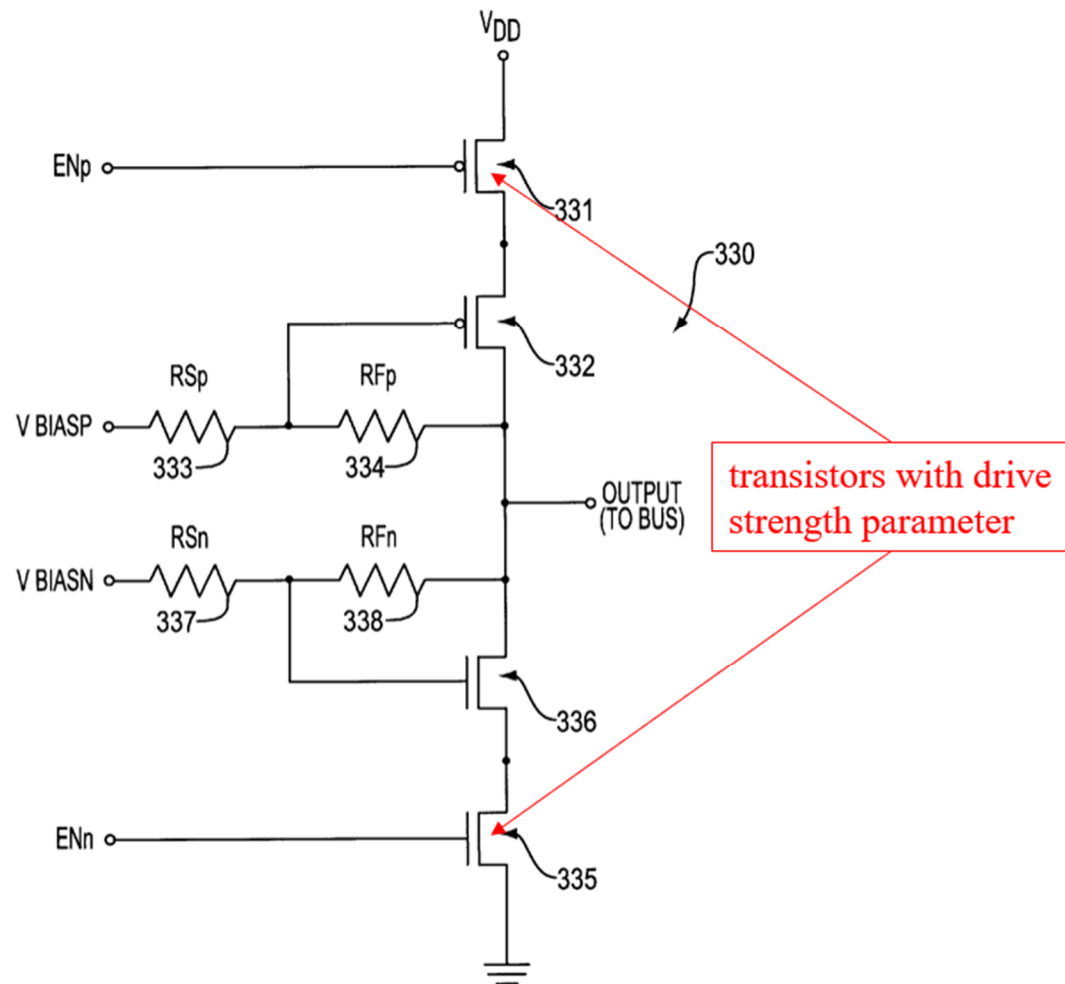


FIG. 1

Claim 5	
5: The method of claim 1, wherein the parameter is a driver strength.	<u>Discussion:</u> As shown below in annotated Fig. 3, Greeff discloses an active termination circuit 330 including circuitry capable of receiving a digital signal that uses transistors 331 and 335 as switches that enable or disable the active termination. Ex. 1006, 6:40-55, Fig. 3. Greeff explains that the active termination circuit 330 can be used, for example, in any/all of the devices of Greeff, including Fig. 1, as discussed above in claim 4. Greeff explains that the drive strength of the transistors can be calibrated. Ex. 1006, 8:3-5, claim 42. Ex. 1006, 6:39-44. A POSITA would have understood Greeff’s drive strength of the transistors 331 and 335 to be the claimed “driver strength.”



Claim 13	
13: The system of claim 10, wherein the parameter is a link termination resistance.	<i>See</i> Ground 3, claim 4.
Claim 14	
14: The system of claim 10, wherein the parameter is a driver strength for a driver of the receive component.	<i>See</i> Ground 3, claim 5.
Claim 17	
17[pre]: A system, comprising:	Meaney discloses this limitation, as I discussed above in Ground 1 at 10[pre].
17[a]: circuitry to store an existing value representative of at least one of a termination resistance and a driver strength to be applied in association with data communicated across a channel;	<u>Discussion</u> As I discussed in Ground 1 at 10[e], Meaney discloses circuitry to store an existing value of value of a parameter. Greeff discloses that the existing value is representative of at least one of a termination resistance and a driver strength to be applied in association with data communicated across a channel. As I discussed above regarding claims 4 and 5 in this Ground 3, Greeff discloses calibration of both a termination resistance and a drive strength of transistors for active termination. <u>Other Relevant Citations</u> Meany at [0015]: “By keeping the interface from being used, the driving circuits can exclusively send the calibration pattern and the calibration logic can re-center the clock under any new environmental conditions or circuit changes.” <i>Id.</i> at [0016]: “The invention can be triggered periodically. This way, the changes to circuit or environmental characteristics over time do not adversely affect the operation of the interface.”

	<i>Id.</i> at [0020]: “FIG. 2 illustrates the prior art state diagram for performing the initial calibration of the hardware as performed in conjunction with the prior art; while” <i>Id.</i> at [0021]: “FIG. 3 illustrates the preferred embodiment state diagrams for supporting the initial calibration as well as the recalibration of the hardware as performed in conjunction with the invention.” <i>Id.</i> at [0023]: “Turning to FIG. 1, notice that there is driver logic, 11, used to drive interface bus, 12, and clock, 13, which feeds receiver calibration logic, 14. Said receiver logic, 14, consisting of state machine logic, 15, clock calibrating hardware, 16, and data de-skew hardware 17. The output of receiver calibration logic, 14, transfers data onto receiver output bus, 18, which connects to system logic, 19.” <i>Id.</i> at [0033]: “The limitation with this sequence is that, once the interfaces are used, there is no means to get the interface recalibrated without resetting the state machine and starting the entire sequence over.” <i>Id.</i> at [0037]: “Then, the SAP uses a hardware interface protocol to read and write registers in the interface logic directly. The SAP writes the Fence registers to make sure the interfaces are truly unused. Then, it loads the driver side calibration flag. This flag signals hardware to drive a repeating pattern across the interface. The pattern for the preferred embodiment is 100010001000” <i>Id.</i> at [0040]: “The clock delays are reset as part of the final clock calibration sequence. By adding delay to the clock path, the clock calibration logic is able to find the optimum data capture time for the interface. Since the calibration is done periodically, this window gets reoptimized every time recalibration occurs. Also, if the cycle time, voltage, or other changes are made to the environment for testing, the interface can recalibrate to the new conditions. However, the data deskew sequence does not need to be run since the deskew settings are not likely to change since the data nets will tend to track over time.” <i>Id.</i> at [0051]: “Although the preferred embodiment recalculates the flag0, flag1, and flag2 values, there are other methods that can be used to recalibrate the clock. For instance, if it is known that only the frequency changes, the change in frequency can be calculated by additional hardware and half the difference can be applied as a delay shift. Although this would be faster, we excluded this form of recalibration due to the complexity and the fact that the preferred embodiment was fast enough to avoid the system hangs that the full calibration would cause. Also, the preferred embodiment uses the same sequences used in the original.”
--	--

As shown below in Fig. 3, Greeff discloses an active termination circuit 330 including circuitry capable of receiving a digital signal that uses transistors 331 and 335 as switches that enable or disable the active termination. Greeff at 6:40-55, Fig. 3. Greeff explains that the drive strength of the transistors can be calibrated. *Id.* at 8:3-5, claim 42. A POSITA would have understood Greeff's drive strength to be the claimed "driver strength."

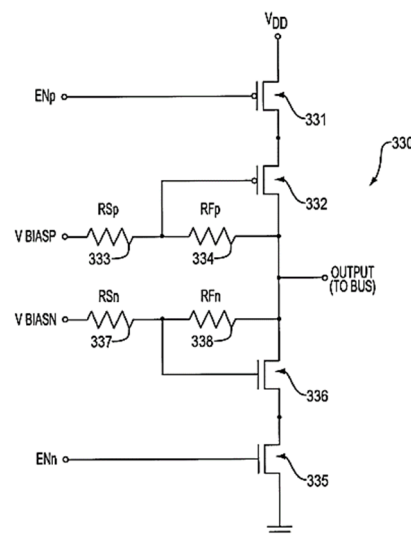


FIG. 3

Additionally, Greeff discloses a multidrop bus including circuitry capable of receiving a digital signal that includes termination circuitry 120 including a switch 122 connected between a "trimmable" termination resistor 124, having a trimmable termination resistance R_{TERM} , and a reference voltage V_{TERM} . Greeff at 3:65-4:20, 5:59-6:6, 7:52-8:56, Fig. 1. A POSITA would have understood Greeff's termination resistance to be the claimed "link termination resistance." Greeff explains that the termination resistance is modified during initial calibration and recalibration processes to provide a substantially optimal termination of the bus 102. *Id.* at 4:11-20, 7:52-8:56.

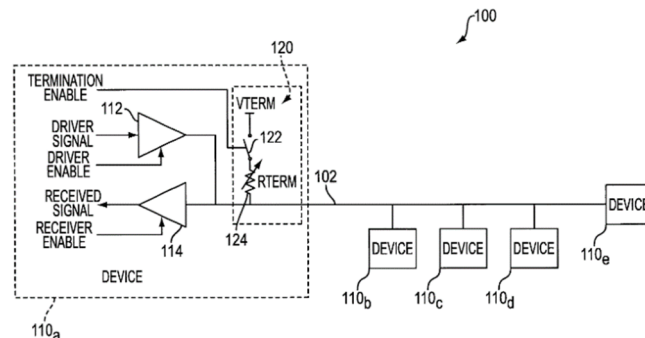


FIG. 1

17[b]: circuitry to perform a first calibration at system initialization, to identify a first value to be initially applied as the existing value,

Discussion

As I discussed in Ground 1 at 10[pre]-10[b], Meaney discloses circuitry to perform a first calibration at system initialization, to identify a first value to be initially applied as the existing value.

Greeff discloses calibrating parameters such as termination resistance prior to the first use of the system, which is referred to as “static calibration.” Ex. 1006, 4:10-13, 7:52-61. As shown in Fig. 2 (reproduced below), Greeff discloses active termination circuitry that would be used to calibrate the termination resistance (“RTERM”). Ex. 1006, 4:10-13, 7:52-61, Fig. 2.

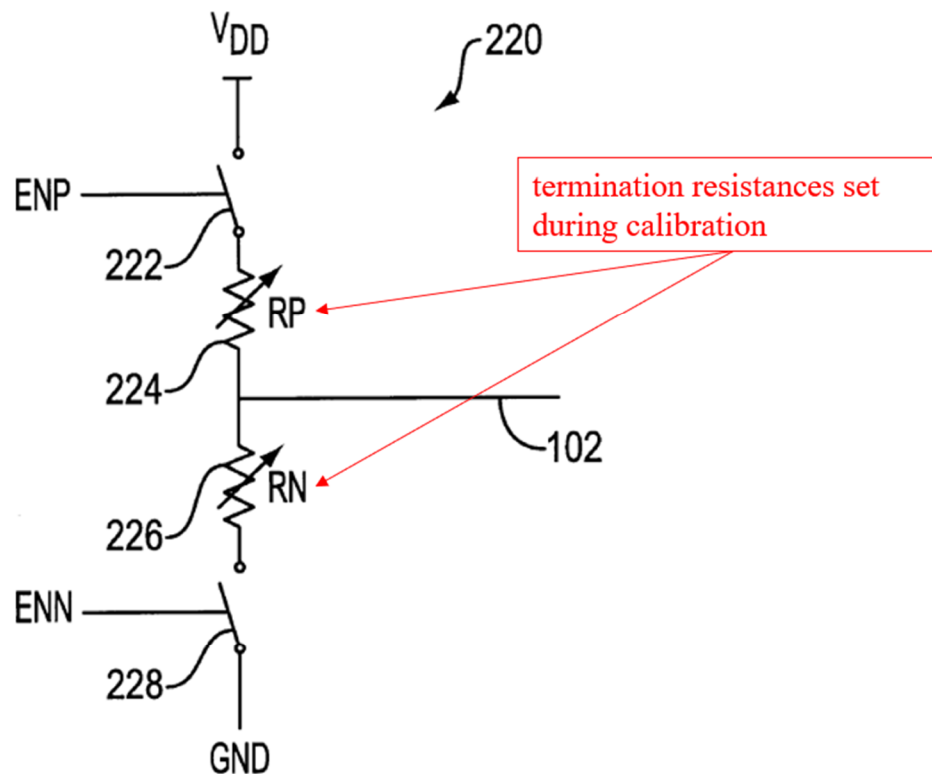


FIG. 2

For example, Greeff explains that the “two termination resistances RP, RN would be set during the calibration process to provide substantially optimal termination,” and that the termination enable signals ENP and ENN would be generated during calibration as well. Ex. 1006, 6:3-5.

Other Relevant Citations

	Meaney at Abstract: “An SMP computer system has an apparatus and method for recalibrating a self-timed, source-synchronous, pipelined interface while the computer system is running. The apparatus allows for quiescing the interface (ie. idling the processors to allow for no data transfers), raising fences (blocking interfaces), allowing for a quick clock centering recalibration step, and then unfencing and unquiescing to allow for the use of the interface again. The recalibration allows for compensating for drift over time on the interface to compensate for temperature, voltage, cycle time, and end-of-life degradation without bringing down and restarting the system.” <i>Id.</i> at [0001]: “This invention relates to symmetrical computer systems, and particularly to an apparatus and method for recalibrating an [sic] source-synchronous pipelined interface with minimal impact to a running system, allowing for the system to remain operational, even in the midst of environmental drift or degradation.” <i>Id.</i> at [0003]-[0004]: “In an SMP (symmetric multiprocessing) computer system with high-level packaging, it is often necessary to allow latencies on cards, wires, or boards that exceed the cycle times of the transferred data. In order to allow these transfers to take place, source-synchronous pipelined interfaces have been used in the s/390 machines of IBM. This allows for the proper capturing of the data within a small window, or ‘eye’. In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.” <i>Id.</i> at [0006]: “The prior s/390 IBM machines only handled the first calibration. There was no way to re-calibrate an interface once it was up and running. Therefore, the cumulative degradations could eventually cause a system failure.” <i>Id.</i> at [0007]: “Also, in a test environment, we often apply environmental stress to a machine as well as frequency changes. We often want to run a hardware/architecture exerciser (like SAK or PCX) under several environments (about 16 voltage corners as well as cycling up or down the frequency), the interface often fails without a re-calibration. If the system is not recalibrated under the new environment, the system must be restarted for every test. Therefore, our test time is jeopardized.” <i>Id.</i> at [0015]-[0016]: “By keeping the interface from being used, the driving circuits can exclusively send the calibration pattern and the calibration logic can re-center the clock under any new environmental conditions or circuit changes. The invention can be triggered periodically. This way, the changes to circuit or environmental characteristics over time do not adversely affect the operation of the interface.”
--	---

	<i>Id.</i> at [0020]-[0021]: “FIG. 2 illustrates the prior art state diagram for performing the initial calibration of the hardware as performed in conjunction with the prior art; FIG. 3 illustrates the preferred embodiment state diagrams for supporting the initial calibration as well as the recalibration of the hardware as performed in conjunction with the invention.” <i>Id.</i> at [0023]: “Turning to FIG. 1, notice that there is driver logic, 11, used to drive interface bus, 12, and clock, 13, which feeds receiver calibration logic, 14. Said receiver logic, 14, consisting of state machine logic, 15, clock calibrating hardware, 16, and data de-skew hardware 17. The output of receiver calibration logic, 14, transfers data onto receiver output bus, 18, which connects to system logic, 19.” <i>Id.</i> at [0024] : “In the prior art, the system was always recalibrated from a stopped position. The system would be scan-initialized to a reset state and the calibration would begin when the clocks were started.”³: “ <i>Id.</i> at [0038]: “The SAP next signals the receiver calibration logic to perform the fast initialization procedure. Since the data deskew has already been done, it is not necessary to perform this lengthy step during recalibration. The fast initialization procedure consists of starting the state machine at the final clock calibration step.” <i>Id.</i> at [0040]: “The clock delays are reset as part of the final clock calibration sequence. By adding delay to the clock path, the clock calibration logic is able to find the optimum data capture time for the interface. Since the calibration is done periodically, this window gets reoptimized every time recalibration occurs. Also, if the cycle time, voltage, or other changes are made to the environment for testing, the interface can recalibrate to the new conditions. However, the data deskew sequence does not need to be run since the deskew settings are not likely to change since the data nets will tend to track over time.”
--	---

³ Although *Meaney* refers to this discussion as prior art, *Meaney* is clear that the preferred method is “similar to the prior art calibration sequence” but includes “additional steps.” Ex. 1004 at [0035]. Thus, *Meaney* is clear that the initial calibration of the clock would be part of the preferred method as well.

Id. at [0051]: “Although the preferred embodiment recalculates the flag0, flag1, and flag2 values, there are other methods that can be used to recalibrate the clock. For instance, if it is known that only the frequency changes, the change in frequency can be calculated by additional hardware and half the difference can be applied as a delay shift. Although this would be faster, we excluded this form of recalibration due to the complexity and the fact that the preferred embodiment was fast enough to avoid the system hangs that the full calibration would cause. Also, the preferred embodiment uses the same sequences used in the original.”

Id. at claim 12: “The method according to claim 10 wherein the recalibration stem includes re-calculating the clock frequency of the interface against the current hardware and re-applying the clock frequency calculation to the clock delay to re-center the clock when the machine is being cycled down to failure and the major change needing re-calibration is cycle time.”

Id. at Fig. 1:

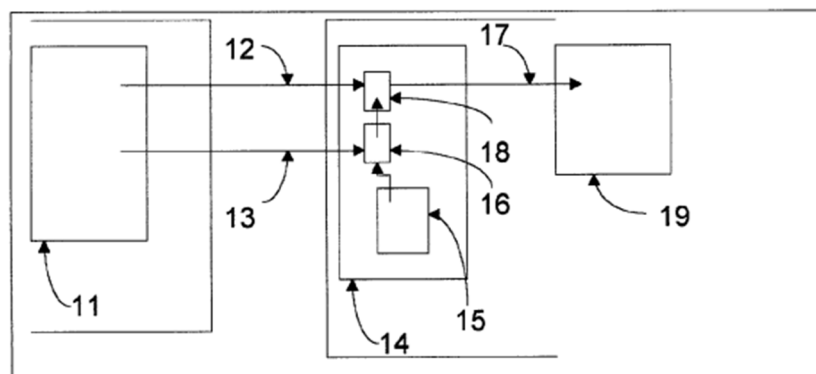


FIGURE 1.

Id. at Fig. 2:

E.I. CLOCK RECEIVER STATE MACHINE DIAGRAM

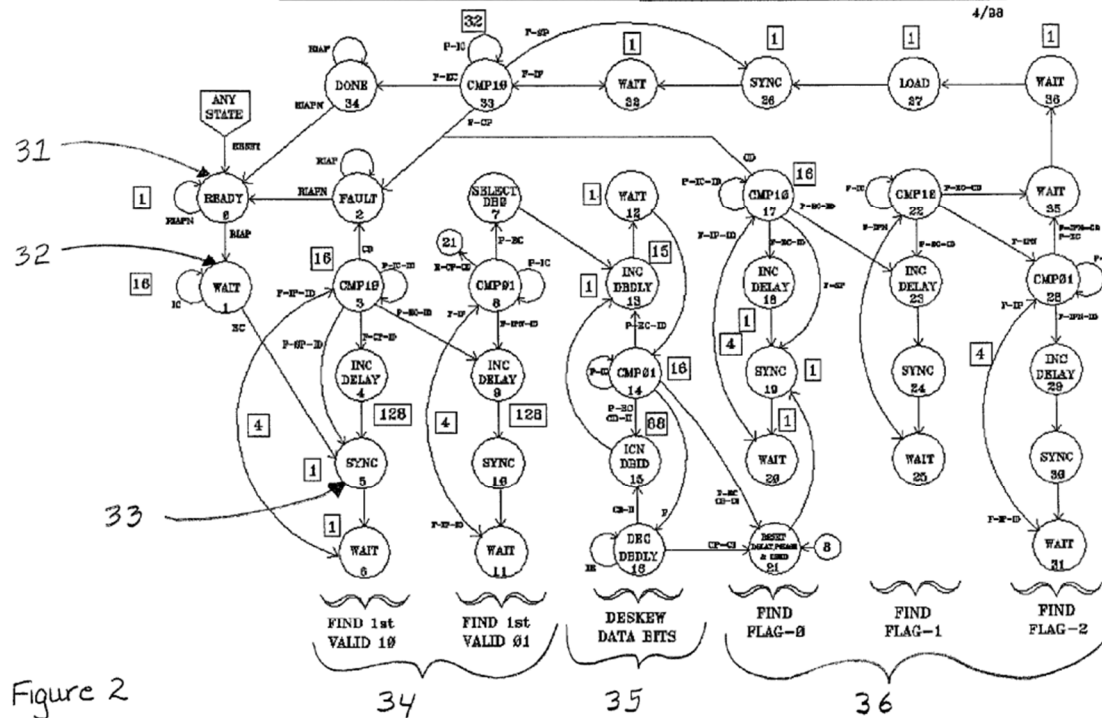


Figure 2

Id. at Fig. 3:

7/89

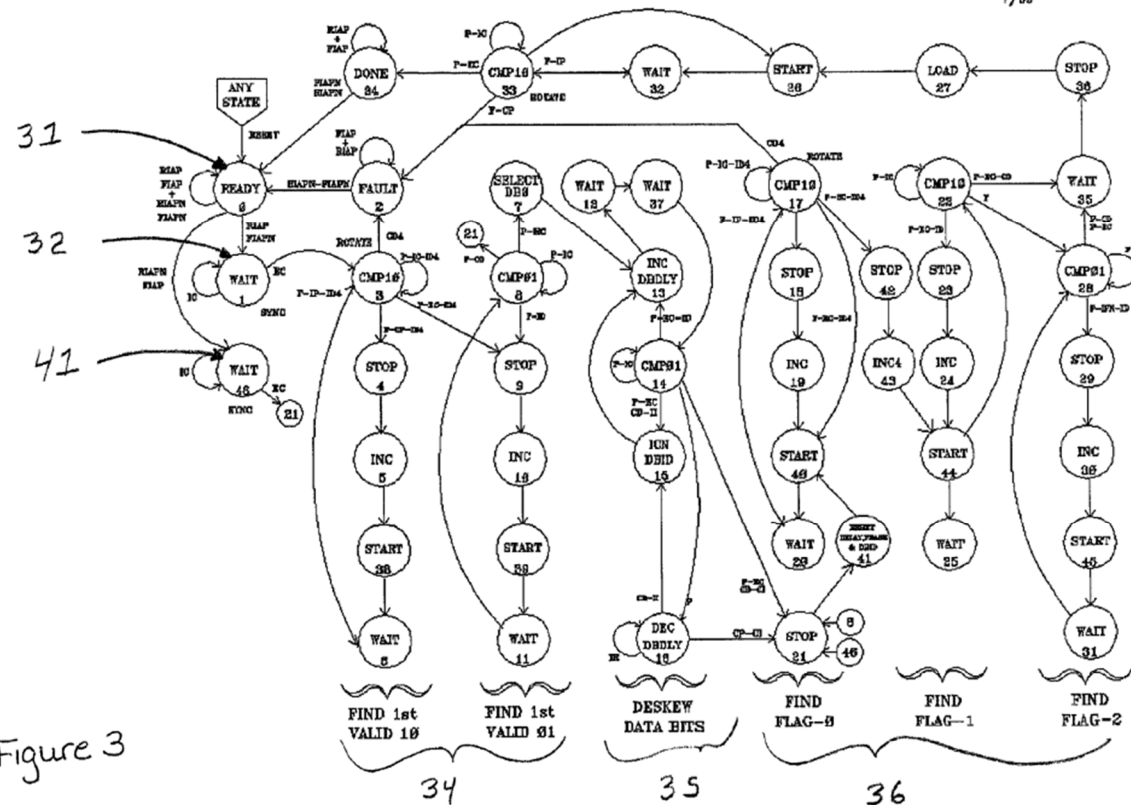


Figure 3

Greeff explains that the “two termination resistances RP, RN would be set during the calibration process to provide substantially optimal termination,” and that the termination enable signals ENP and ENN would be generated during calibration as well. *Id.* at 6:3-5.

17[c]: the first calibration to be performed during a first calibration interval prior to normal system operation; and	<u>Discussion</u> As I discussed in Ground 1 at 10[a], Meaney discloses the first calibration to be performed during initialization, prior to normal system operation. Greeff discloses calibrating parameters such as termination resistance prior to the first use of the system, which is referred to as “static calibration.” Ex. 1006, 7:52-61. <u>Other Relevant Citations</u> Meaney at Abstract: “An SMP computer system has an apparatus and method for recalibrating a self-timed, source-synchronous, pipelined interface while the computer system is running. The apparatus allows for quiescing the interface (ie. idling the processors to allow for no data transfers), raising fences (blocking interfaces), allowing for a quick clock centering recalibration step, and then unfencing and unquiescing to allow for the use of the interface again. The recalibration allows for compensating for drift over time on the interface to compensate for temperature, voltage, cycle time, and end-of-life degradation without bringing down and restarting the system.” <i>Id.</i> at [0001]: “This invention relates to symmetrical computer systems, and particularly to an apparatus and method for recalibrating an [sic] source-synchronous pipelined interface with minimal impact to a running system, allowing for the system to remain operational, even in the midst of environmental drift or degradation.” <i>Id.</i> at [0003]-[0004]: “In an SMP (symmetric multiprocessing) computer system with high-level packaging, it is often necessary to allow latencies on cards, wires, or boards that exceed the cycle times of the transferred data. In order to allow these transfers to take place, source-synchronous pipelined interfaces have been used in the s/390 machines of IBM. This allows for the proper capturing of the data within a small window, or ‘eye’. In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.” <i>Id.</i> at [0007]: “Also, in a test environment, we often apply environmental stress to a machine as well as frequency changes. We often want to run a hardware/architecture exerciser (like SAK or PCX) under several environments (about 16 voltage corners as well as cycling up or down the frequency), the interface often fails without a re-calibration. If the system is not recalibrated under the new environment, the system must be restarted for every test. Therefore, our test time is jeopardized.”
--	---

Id. at [0015]-[0016]: “By keeping the interface from being used, the driving circuits can exclusively send the calibration pattern and the calibration logic can re-center the clock under any new environmental conditions or circuit changes. The invention can be triggered periodically. This way, the changes to circuit or environmental characteristics over time do not adversely affect the operation of the interface.”

Id. at [0020]-[0021]: “FIG. 2 illustrates the prior art state diagram for performing the initial calibration of the hardware as performed in conjunction with the prior art; FIG. 3 illustrates the preferred embodiment state diagrams for supporting the initial calibration as well as the recalibration of the hardware as performed in conjunction with the invention.”

Id. at [0023]: “Turning to FIG. 1, notice that there is driver logic, 11, used to drive interface bus, 12, and clock, 13, which feeds receiver calibration logic, 14. Said receiver logic, 14, consisting of state machine logic, 15, clock calibrating hardware, 16, and data de-skew hardware 17. The output of receiver calibration logic, 14, transfers data onto receiver output bus, 18, which connects to system logic, 19.”

Id. at Fig. 1:

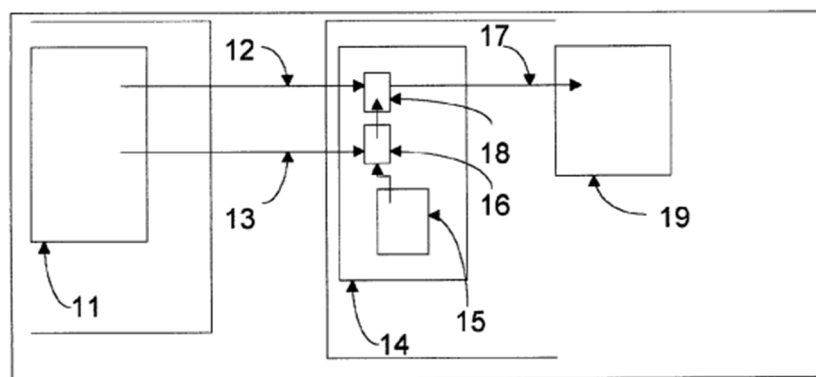


FIGURE 1.

Id. at Fig. 2:

E.I. CLOCK RECEIVER STATE MACHINE DIAGRAM

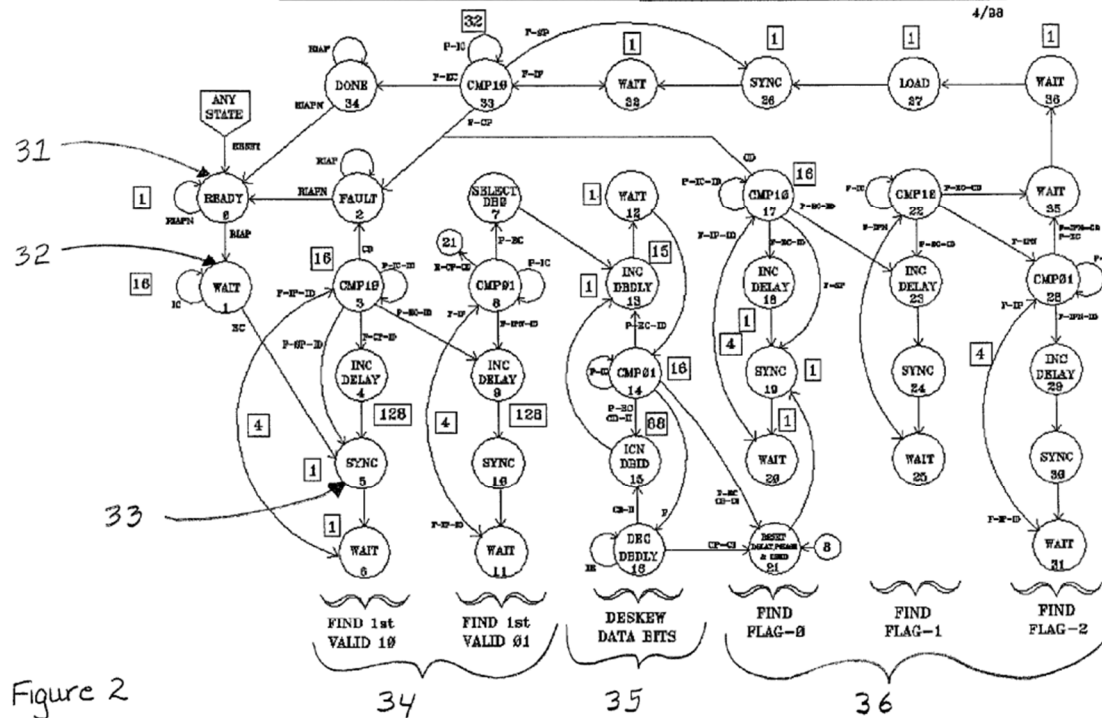


Figure 2

Id. at Fig. 3:

E.I. CLOCK RECEIVER STATE MACHINE DIAGRAM

7/89

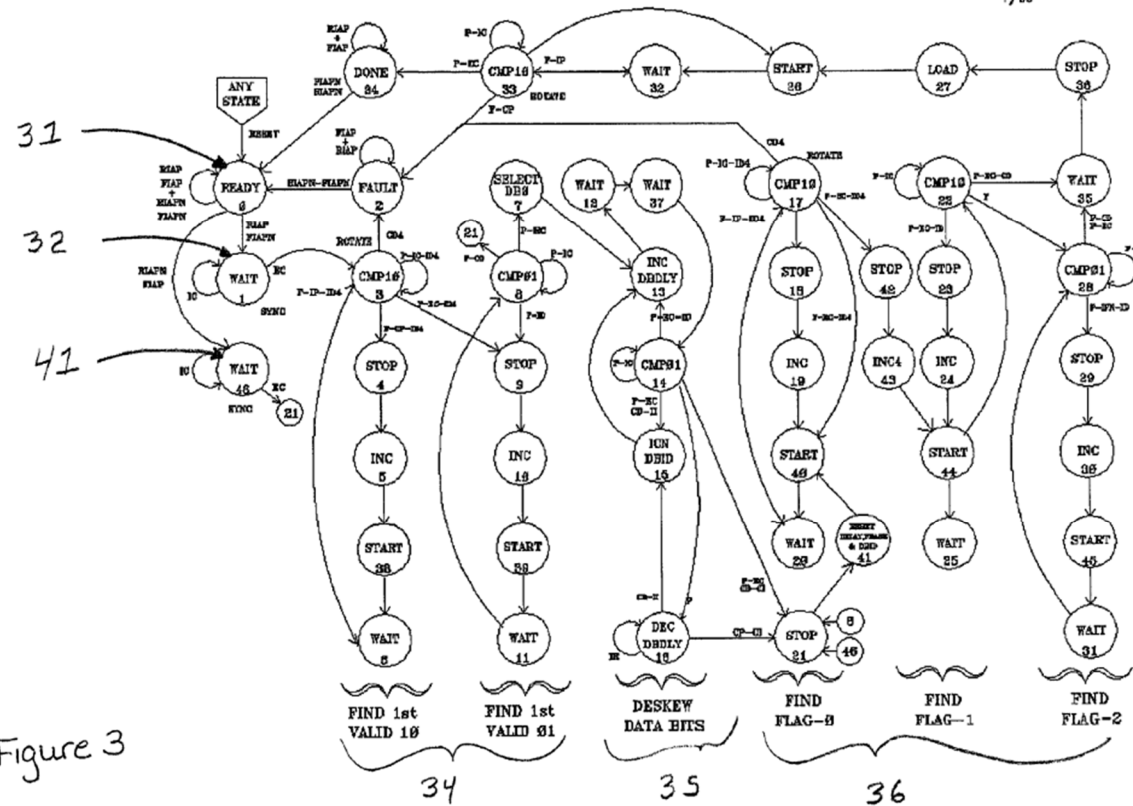


Figure 3

Additionally, for example, Greeff teaches calibrating parameters such as termination resistance prior to the first use of the system, which is referred to as “static calibration.” Greeff at 7:52-61.

17[d]: circuitry to periodically perform a second calibration, to update the existing value;	<u>Discussion</u> As I discussed in Ground 1 at 10[a] and 10[c], Meaney discloses periodically subjecting receiver circuitry to a second calibration in the form of a “recalibration” that updates an existing parameter value. As I discussed above in 17[b] of this Ground 3, Greeff discloses circuitry for performing calibration of termination resistance. Greeff further discloses recalibrating termination resistance during operation of the system, for example when the value changes due to temperature or supply voltage changes, which is referred to as “dynamic calibration.” Ex. 1006, 7:52-65. Thus, the combination of Meaney and Greeff discloses circuitry to periodically recalibrate an existing value of termination resistance. <u>Other Relevant Citations</u> Meaney at Abstract: “An SMP computer system has an apparatus and method for recalibrating a self-timed, source-synchronous, pipelined interface while the computer system is running. The apparatus allows for quiescing the interface (ie. idling the processors to allow for no data transfers), raising fences (blocking interfaces), allowing for a quick clock centering recalibration step, and then unfencing and unquiescing to allow for the use of the interface again. The recalibration allows for compensating for drift over time on the interface to compensate for temperature, voltage, cycle time, and end-of-life degradation without bringing down and restarting the system.” <i>Id.</i> at [0001]: “This invention relates to symmetrical computer systems, and particularly to an apparatus and method for recalibrating an [sic] source-synchronous pipelined interface with minimal impact to a running system, allowing for the system to remain operational, even in the midst of environmental drift or degradation.” <i>Id.</i> at [0003]-[0004]: “In an SMP (symmetric multiprocessing) computer system with high-level packaging, it is often necessary to allow latencies on cards, wires, or boards that exceed the cycle times of the transferred data. In order to allow these transfers to take place, source-synchronous pipelined interfaces have been used in the s/390 machines of IBM. This allows for the proper capturing of the data within a small window, or ‘eye’. In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.”
--	--

	<i>Id.</i> at [0007]: “Also, in a test environment, we often apply environmental stress to a machine as well as frequency changes. We often want to run a hardware/architecture exerciser (like SAK or PCX) under several environments (about 16 voltage corners as well as cycling up or down the frequency), the interface often fails without a re-calibration. If the system is not recalibrated under the new environment, the system must be restarted for every test. Therefore, our test time is jeopardized.” <i>Id.</i> at [0009]-[0016]: “The steps for re-calibration involve: 1. Putting the system of the interface into a wait state, 2. Performing a fast initialization process for calibration, and 3. Taking the system of the interface out of a wait state[.] The fast initialization process in the preferred embodiment is similar to the prior art, except that there is only one clock centering step, instead of two. Also, the data deskew step is skipped since that step has already been performed as part of the original initialization process. The step of putting the system into the wait state can be done through several methods as described in the referenced prior art. One way is to idle all the processors while having one processor oversee the recalibration sequence. This same processor may take the system out of the wait state as well. By keeping the interface from being used, the driving circuits can exclusively send the calibration pattern and the calibration logic can re-center the clock under any new environmental conditions or circuit changes. The invention can be triggered periodically. This way, the changes to circuit or environmental characteristics over time do not adversely affect the operation of the interface.” <i>Id.</i> at [0020]-[0021]: “FIG. 2 illustrates the prior art state diagram for performing the initial calibration of the hardware as performed in conjunction with the prior art; FIG. 3 illustrates the preferred embodiment state diagrams for supporting the initial calibration as well as the recalibration of the hardware as performed in conjunction with the invention.” <i>Id.</i> at [0023]: “Turning to FIG. 1, notice that there is driver logic, 11, used to drive interface bus, 12, and clock, 13, which feeds receiver calibration logic, 14. Said receiver logic, 14, consisting of state machine logic, 15, clock calibrating hardware, 16, and data de-skew hardware 17. The output of receiver calibration logic, 14, transfers data onto receiver output bus, 18, which connects to system logic, 19.”
--	---

Id. at [0040]: “The clock delays are reset as part of the final clock calibration sequence. By adding delay to the clock path, the clock calibration logic is able to find the optimum data capture time for the interface. Since the calibration is done periodically, this window gets reoptimized every time recalibration occurs. Also, if the cycle time, voltage, or other changes are made to the environment for testing, the interface can recalibrate to the new conditions. However, the data deskew sequence does not need to be run since the deskew settings are not likely to change since the data nets will tend to track over time.”

Id. at [0051]: “Although the preferred embodiment recalculates the flag0, flag1, and flag2 values, there are other methods that can be used to recalibrate the clock. For instance, if it is known that only the frequency changes, the change in frequency can be calculated by additional hardware and half the difference can be applied as a delay shift. Although this would be faster, we excluded this form of recalibration due to the complexity and the fact that the preferred embodiment was fast enough to avoid the system hangs that the full calibration would cause. Also, the preferred embodiment uses the same sequences used in the original.”

Id. at claim 12: “The method according to claim 10 wherein the recalibration stem includes re-calculating the clock frequency of the interface against the current hardware and re-applying the clock frequency calculation to the clock delay to re-center the clock when the machine is being cycled down to failure and the major change needing re-calibration is cycle time.”

Id. at Fig. 1:

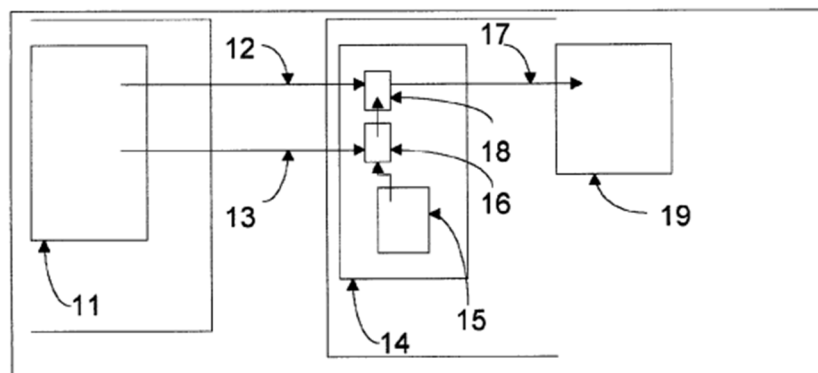


FIGURE 1.

Id. at Fig. 2:

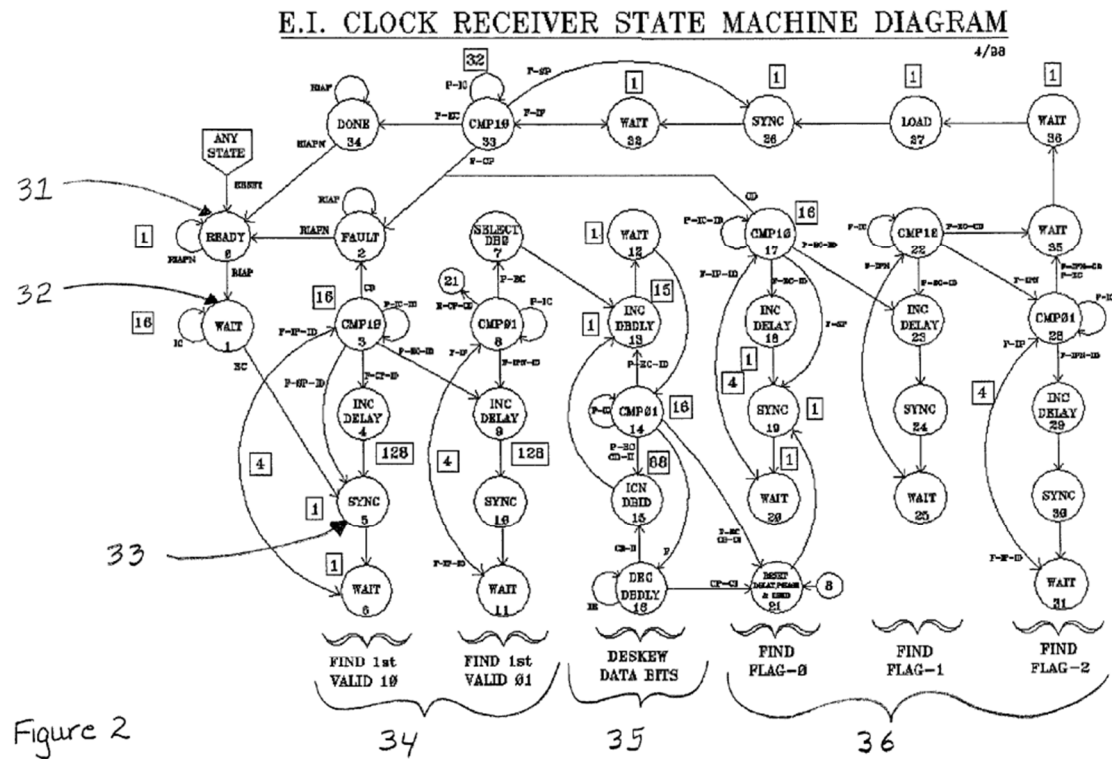
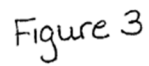


Figure 2

Id. at Fig. 3:

7/99



Greeff teaches calibrating parameters such as termination resistance prior to the first use of the system, which is referred to as “static calibration.” Greeff at 4:10-13, 7:52-61. As shown in Fig. 2 (reproduced below), Greeff teaches active termination circuitry that would be used to calibrate the termination resistance (“RTERM”). *Id.* at 4:10-13, 7:52-61, Fig. 2. Greeff further teaches recalibrating termination resistance during operation of the system, for example when the value changes due to temperature or supply voltage changes, which is referred to as “dynamic calibration.” Greeff at 7:52-65.

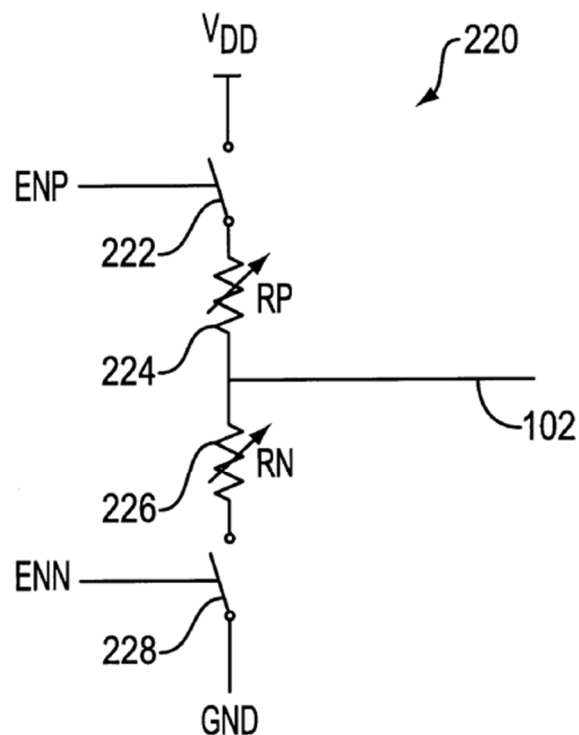


FIG. 2

17[e]: wherein the existing value is initially set and periodically updated responsive to the first calibration and the second calibration, respectively, and	<u>Discussion</u> As I discussed in Ground 1 at 1[a] and 1[b], Meaney discloses determining an initial value for a parameter through an initial calibration and periodically performing a recalibration to update the parameter value. Greeff discloses calibration techniques including initially setting a termination resistance at a known value and then performing the calibration process to update the value. Ex. 1006, 8:27-33 (“Another calibration process may [] first set[] the resistance of the pull-up (or pull-down) resistor . . . against a known passive resistor, and then balance[e] the pull-down (or pull-up) resistor.”). Greeff also discloses an initial calibration to set a value (static calibration) and a second calibration (dynamic calibration) to adapt to changes in operating conditions. Ex. 1006, 4:10-13, 7:52-617:52-65. Thus, Meaney and Greeff, in view of the knowledge of a POSITA, render obvious “wherein the existing value is initially set and periodically updated responsive to the first calibration and the second calibration, respectively.” <u>Other Relevant Citations</u> Meaney at Abstract: “An SMP computer system has an apparatus and method for recalibrating a self-timed, source-synchronous, pipelined interface while the computer system is running. The apparatus allows for quiescing the interface (ie. idling the processors to allow for no data transfers), raising fences (blocking interfaces), allowing for a quick clock centering recalibration step, and then unfencing and unquiescing to allow for the use of the interface again. The recalibration allows for compensating for drift over time on the interface to compensate for temperature, voltage, cycle time, and end-of-life degradation without bringing down and restarting the system.” <i>Id.</i> at [0001]: “This invention relates to symmetrical computer systems, and particularly to an apparatus and method for recalibrating an [sic] source-synchronous pipelined interface with minimal impact to a running system, allowing for the system to remain operational, even in the midst of environmental drift or degradation.”
---	---

	<i>Id.</i> at [0003]-[0004]: “In an SMP (symmetric multiprocessing) computer system with high-level packaging, it is often necessary to allow latencies on cards, wires, or boards that exceed the cycle times of the transferred data. In order to allow these transfers to take place, source-synchronous pipelined interfaces have been used in the s/390 machines of IBM. This allows for the proper capturing of the data within a small window, or ‘eye’. In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.” <i>Id.</i> at [0006]: “The prior s/390 IBM machines only handled the first calibration. There was no way to re-calibrate an interface once it was up and running. Therefore, the cumulative degradations could eventually cause a system failure.” <i>Id.</i> at [0007]: “Also, in a test environment, we often apply environmental stress to a machine as well as frequency changes. We often want to run a hardware/architecture exerciser (like SAK or PCX) under several environments (about 16 voltage corners as well as cycling up or down the frequency), the interface often fails without a re-calibration. If the system is not recalibrated under the new environment, the system must be restarted for every test. Therefore, our test time is jeopardized.” <i>Id.</i> at [0015]-[0016]: “By keeping the interface from being used, the driving circuits can exclusively send the calibration pattern and the calibration logic can re-center the clock under any new environmental conditions or circuit changes. The invention can be triggered periodically. This way, the changes to circuit or environmental characteristics over time do not adversely affect the operation of the interface.” <i>Id.</i> at [0020]-[0021]: “FIG. 2 illustrates the prior art state diagram for performing the initial calibration of the hardware as performed in conjunction with the prior art; FIG. 3 illustrates the preferred embodiment state diagrams for supporting the initial calibration as well as the recalibration of the hardware as performed in conjunction with the invention.” <i>Id.</i> at [0023]: “Turning to FIG. 1, notice that there is driver logic, 11, used to drive interface bus, 12, and clock, 13, which feeds receiver calibration logic, 14. Said receiver logic, 14, consisting of state machine logic, 15, clock calibrating hardware, 16, and data de-skew hardware 17. The output of receiver calibration logic, 14, transfers data onto receiver output bus, 18, which connects to system logic, 19.”
--	--

	<i>Id.</i> at [0024] : “In the prior art, the system was always recalibrated from a stopped position. The system would be scan-initialized to a reset state and the calibration would begin when the clocks were started.”⁴; “ <i>Id.</i> at [0038]: “The SAP next signals the receiver calibration logic to perform the fast initialization procedure. Since the data deskew has already been done, it is not necessary to perform this lengthy step during recalibration. The fast initialization procedure consists of starting the state machine at the final clock calibration step.” <i>Id.</i> at [0040]: “The clock delays are reset as part of the final clock calibration sequence. By adding delay to the clock path, the clock calibration logic is able to find the optimum data capture time for the interface. Since the calibration is done periodically, this window gets reoptimized every time recalibration occurs. Also, if the cycle time, voltage, or other changes are made to the environment for testing, the interface can recalibrate to the new conditions. However, the data deskew sequence does not need to be run since the deskew settings are not likely to change since the data nets will tend to track over time.” <i>Id.</i> at [0051]: “Although the preferred embodiment recalculates the flag0, flag1, and flag2 values, there are other methods that can be used to recalibrate the clock. For instance, if it is known that only the frequency changes, the change in frequency can be calculated by additional hardware and half the difference can be applied as a delay shift. Although this would be faster, we excluded this form of recalibration due to the complexity and the fact that the preferred embodiment was fast enough to avoid the system hangs that the full calibration would cause. Also, the preferred embodiment uses the same sequences used in the original.” <i>Id.</i> at claim 12: “The method according to claim 10 wherein the recalibration stem includes re-calculating the clock frequency of the interface against the current hardware and re-applying the clock frequency calculation to the clock delay to re-center the clock when the machine is being cycled down to failure and the major change needing re-calibration is cycle time.” <i>Id.</i> at Fig. 1:
--	--

⁴ Although *Meaney* refers to this discussion as prior art, *Meaney* is clear that the preferred method is “similar to the prior art calibration sequence” but includes “additional steps.” Ex. 1004 at [0035]. Thus, *Meaney* is clear that the initial calibration of the clock would be part of the preferred method as well.

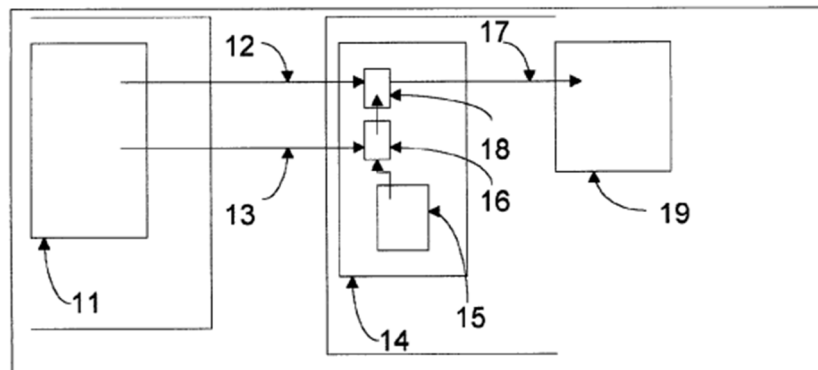
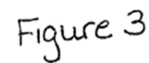


FIGURE 1.

Id. at Fig. 2:

7/89



111

17[f]: wherein the second calibration is constrained to be performed during a time period that is shorter than a time period of the first calibration; and	<u>Discussion</u> As I discussed in Ground 1 at 1[e], Meaney discloses “wherein the second calibration is constrained to be performed during a time period that is shorter than a time period of the first calibration.” <u>Other Relevant Citations</u> Meany at [0015]: “By keeping the interface from being used, the driving circuits can exclusively send the calibration pattern and the calibration logic can re-center the clock under any new environmental conditions or circuit changes.” <i>Id.</i> at [0016]: “The invention can be triggered periodically. This way, the changes to circuit or environmental characteristics over time do not adversely affect the operation of the interface.” <i>Id.</i> at [0020]: “FIG. 2 illustrates the prior art state diagram for performing the initial calibration of the hardware as performed in conjunction with the prior art; while” <i>Id.</i> at [0021]: “FIG. 3 illustrates the preferred embodiment state diagrams for supporting the initial calibration as well as the recalibration of the hardware as performed in conjunction with the invention.” <i>Id.</i> at [0023]: “Turning to FIG. 1, notice that there is driver logic, 11, used to drive interface bus, 12, and clock, 13, which feeds receiver calibration logic, 14. Said receiver logic, 14, consisting of state machine logic, 15, clock calibrating hardware, 16, and data de-skew hardware 17. The output of receiver calibration logic, 14, transfers data onto receiver output bus, 18, which connects to system logic, 19.” <i>Id.</i> at [0033]: “The limitation with this sequence is that, once the interfaces are used, there is no means to get the interface recalibrated without resetting the state machine and starting the entire sequence over.” <i>Id.</i> at [0037]: “Then, the SAP uses a hardware interface protocol to read and write registers in the interface logic directly. The SAP writes the Fence registers to make sure the interfaces are truly unused. Then, it loads the driver side calibration flag. This flag signals hardware to drive a repeating pattern across the interface. The pattern for the preferred embodiment is 100010001000”
--	--

	<i>Id.</i> at [0040]: “The clock delays are reset as part of the final clock calibration sequence. By adding delay to the clock path, the clock calibration logic is able to find the optimum data capture time for the interface. Since the calibration is done periodically, this window gets reoptimized every time recalibration occurs. Also, if the cycle time, voltage, or other changes are made to the environment for testing, the interface can recalibrate to the new conditions. However, the data deskew sequence does not need to be run since the deskew settings are not likely to change since the data nets will tend to track over time.” <i>Id.</i> at [0051]: “Although the preferred embodiment recalculates the flag0, flag1, and flag2 values, there are other methods that can be used to recalibrate the clock. For instance, if it is known that only the frequency changes, the change in frequency can be calculated by additional hardware and half the difference can be applied as a delay shift. Although this would be faster, we excluded this form of recalibration due to the complexity and the fact that the preferred embodiment was fast enough to avoid the system hangs that the full calibration would cause. Also, the preferred embodiment uses the same sequences used in the original.”
17[g]: wherein the system further comprises a receive component, the at least one to be applied to permit proper reception of a digital signal to be communicated across a communications channel.	<u>Discussion</u> As I discussed in Ground 1 at 1[pre] and 1[b], Meaney discloses “a system further comprises a receive component” and at least one parameter “to be applied to permit proper reception of a digital signal to be communicated across a communications channel.” Greeff further discloses that the termination resistance and a driver strength are applied to permit proper reception of a digital signal to be communicated across a communications channel. Greeff explains that termination resistance is calibrated to provide substantially optimal termination of the bus, and that drive strength is calibrated to improve achieve the desired active termination. Ex. 1006, 1:28-31, 4:10-12. Greeff discloses that both of these calibrations are performed to improve the termination of the system, which improves digital signal integrity by minimizing transmission line reflections. Ex. 1006, 1:28-31, 4:10-12, 8:3-5. <u>Other Relevant Citations</u>

	Meaney at Abstract: “An SMP computer system has an apparatus and method for recalibrating a self-timed, source-synchronous, pipelined interface while the computer system is running. The apparatus allows for quiescing the interface (ie. idling the processors to allow for no data transfers), raising fences (blocking interfaces), allowing for a quick clock centering recalibration step, and then unfencing and unquiescing to allow for the use of the interface again. The recalibration allows for compensating for drift over time on the interface to compensate for temperature, voltage, cycle time, and end-of-life degradation without bringing down and restarting the system.” <i>Id.</i> at [0001]: “This invention relates to symmetrical computer systems, and particularly to an apparatus and method for recalibrating an [sic] source-synchronous pipelined interface with minimal impact to a running system, allowing for the system to remain operational, even in the midst of environmental drift or degradation.” <i>Id.</i> at [0003]-[0004]: “In an SMP (symmetric multiprocessing) computer system with high-level packaging, it is often necessary to allow latencies on cards, wires, or boards that exceed the cycle times of the transferred data. In order to allow these transfers to take place, source-synchronous pipelined interfaces have been used in the s/390 machines of IBM. This allows for the proper capturing of the data within a small window, or ‘eye’. In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.” <i>Id.</i> at [0015]-[0016]: “By keeping the interface from being used, the driving circuits can exclusively send the calibration pattern and the calibration logic can re-center the clock under any new environmental conditions or circuit changes. The invention can be triggered periodically. This way, the changes to circuit or environmental characteristics over time do not adversely affect the operation of the interface.” <i>Id.</i> at [0020]-[0021]: “FIG. 2 illustrates the prior art state diagram for performing the initial calibration of the hardware as performed in conjunction with the prior art; FIG. 3 illustrates the preferred embodiment state diagrams for supporting the initial calibration as well as the recalibration of the hardware as performed in conjunction with the invention.” <i>Id.</i> at [0023]: “Turning to FIG. 1, notice that there is driver logic, 11, used to drive interface bus, 12, and clock, 13, which feeds receiver calibration logic, 14. Said receiver logic, 14, consisting of state machine logic, 15, clock calibrating hardware, 16, and data de-skew hardware 17. The output of receiver calibration logic, 14, transfers data onto receiver output bus, 18, which connects to system logic, 19.”
--	--

Id. at [0040]: “The clock delays are reset as part of the final clock calibration sequence. By adding delay to the clock path, the clock calibration logic is able to find the optimum data capture time for the interface. Since the calibration is done periodically, this window gets reoptimized every time recalibration occurs. Also, if the cycle time, voltage, or other changes are made to the environment for testing, the interface can recalibrate to the new conditions. However, the data deskew sequence does not need to be run since the deskew settings are not likely to change since the data nets will tend to track over time.”

Id. at claim 12: “The method according to claim 10 wherein the recalibration stem includes re-calculating the clock frequency of the interface against the current hardware and re-applying the clock frequency calculation to the clock delay to re-center the clock when the machine is being cycled down to failure and the major change needing re-calibration is cycle time.”

Id. at Fig. 1:

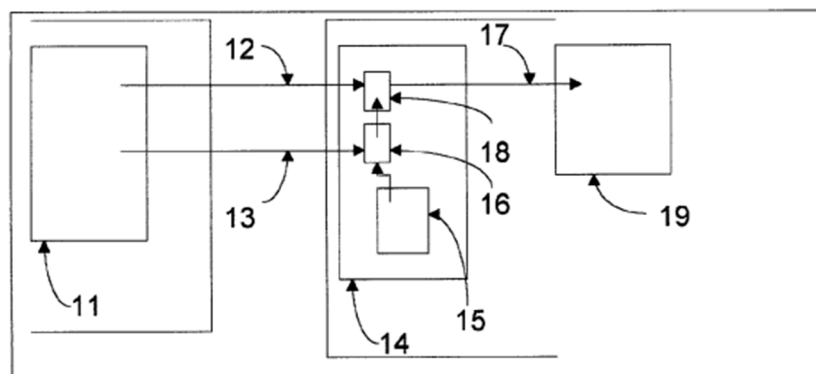


FIGURE 1.

Id. at Fig. 2:

E.I. CLOCK RECEIVER STATE MACHINE DIAGRAM

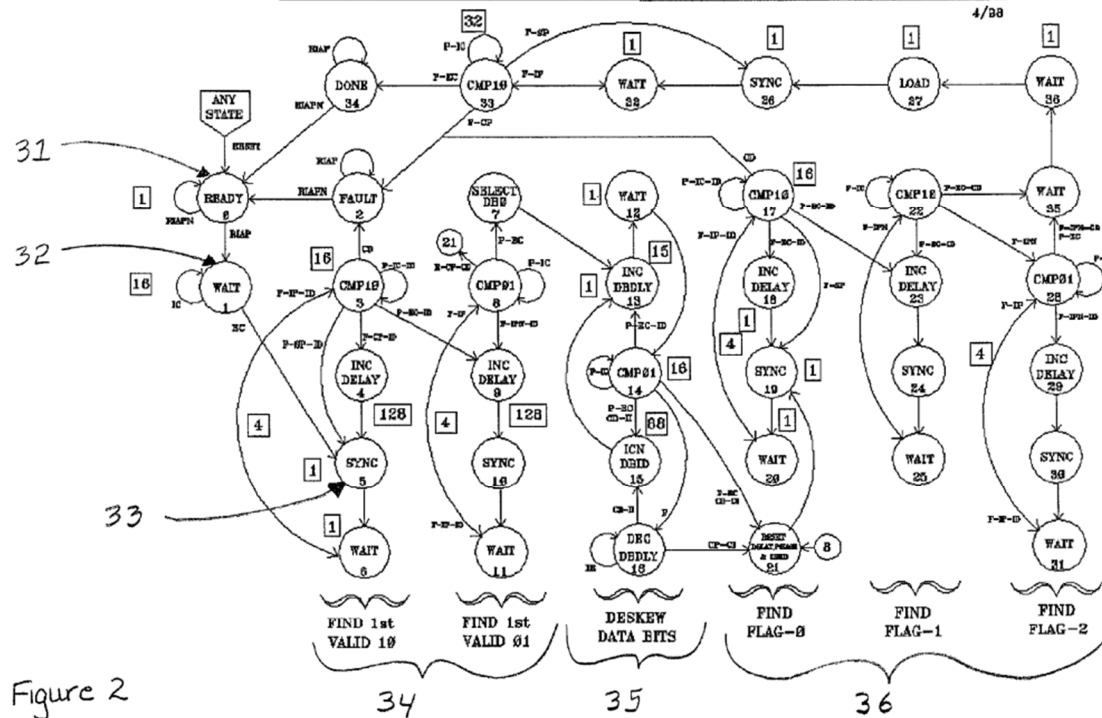


Figure 2

Id. at Fig. 3:

7/99

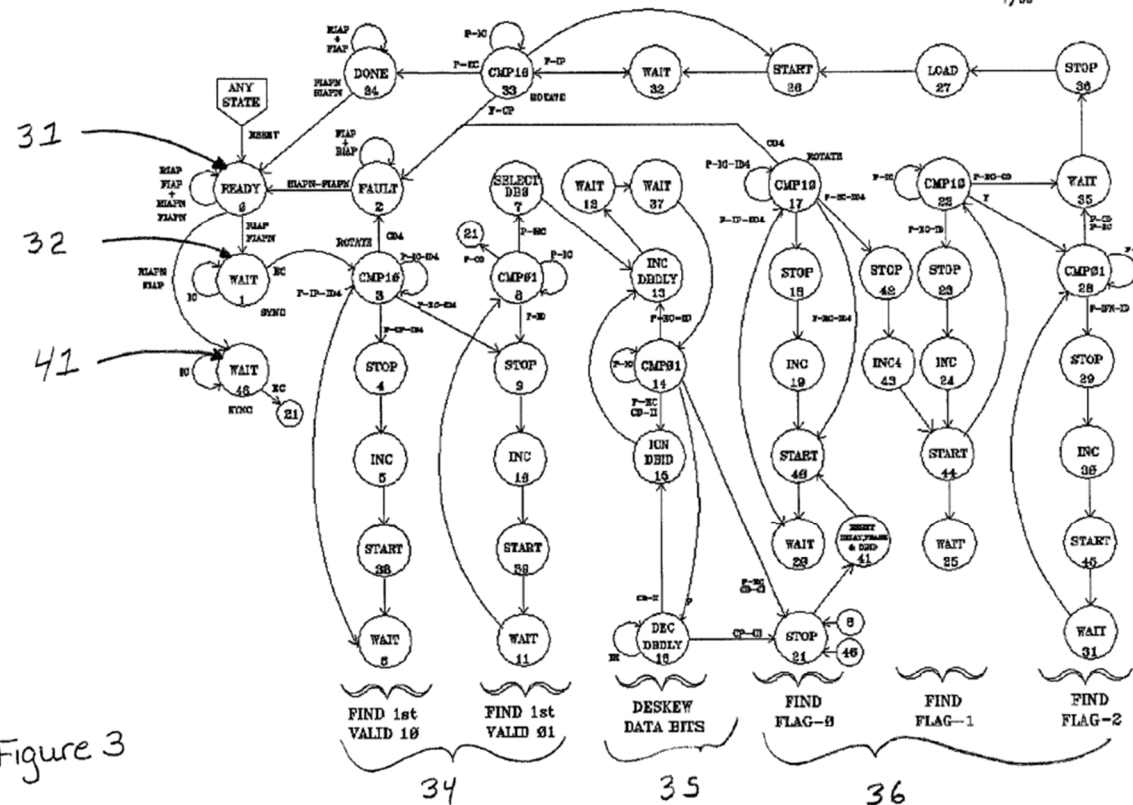


Figure 3

For example, Greeff discloses that the termination resistance and a driver strength are applied to permit proper reception of a digital signal to be communicated across a communications channel. Greeff explains that termination resistance is calibrated to provide substantially optimal termination of the bus, and that drive strength is calibrated to improve achieve the desired active termination. Greeff at 1:28-31, 4:10-12. Both of these calibrations are performed to improve the termination of the system, which improves digital signal integrity by minimizing transmission line reflections. *Id.* at 1:28-31, 4:10-12, 8:3-5.

Claim 18	
18: The system of claim 17, wherein the circuitry to perform the second calibration is, responsive to the second calibration, to perform one of an increment operation or a decrement operation upon the existing value if the desired value has drifted from the existing value by more than a delta value relative to an existing value.	<u>Discussion</u> As I discussed in Ground 1 at claim 15, Meaney discloses this limitation either alone or in combination with the knowledge of a POSITA. Specifically, I explained that Meaney discloses that the receive component performs, responsive to the second calibration, one of an increment operation or a decrement operation upon the existing value if a desired value of the parameter has drifted from the existing value by more than a delta value. I further explained that Meaney discloses that recalibration may be performed in response to a trigger event that suggests that the parameter has drifted and the data capture time window has changed and is no longer accurate, which may be indicated, for example, by the detection of correctable errors. Moreover, I explained in Ground 1 at claim 15, that, it would have been obvious to a POSITA, in response to a recalibration, to perform an increment operation or a decrement operation on the existing value, to shift the value of the based on the results of the recalibration. Greeff similarly discloses increasing and decreasing parameters used in the active termination circuit to account for variations in process, voltage, or temperature. Ex. 1006, 7:66-8:5. It would have been obvious to a POSITA to, in response to a recalibration, implement the increment or decrement operations of Meaney (and that were known in the art) when calibrating the termination resistance of Greeff. This is because a POSITA would have been informed by Meaney on how to implement a specific way to do calibration in Greeff. <u>Other Relevant Citations</u> Meany at [0004]: “In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.” <i>Id.</i> at [0005]: “However, over time, these large-latency interfaces can drift due to environmental changes like voltage, temperature, end-of-life degradation, etc.” <i>Id.</i> at [0008]: “The invention allows for the re-calibration of the interface at periodic intervals with minor disruption. This invention is applicable to the SMP computer systems sold by IBM and others.” <i>Id.</i> at [0016]: “The invention can be triggered periodically. This way, the changes to circuit or environmental characteristics over time do not adversely affect the operation of the interface.”

	<i>Id.</i> at [0017]: “Also, the preferred embodiment allows for the proactive running of the recalibration sequence based on a trigger event. For instance, when ECC is used on an interface and a correctable error (CE) is detected, the data is corrected. The CE can trigger an event to recalibrate the interface. This should protect against future CEs, if the original CE was based on a degraded interface.” <i>Id.</i> at [0033]: “The limitation with this sequence is that, once the interfaces are used, there is no means to get the interface recalibrated without resetting the state machine and starting the entire sequence over.” <i>Id.</i> at [0037]: “Then, the SAP uses a hardware interface protocol to read and write registers in the interface logic directly. The SAP writes the Fence registers to make sure the interfaces are truly unused. Then, it loads the driver side calibration flag. This flag signals hardware to drive a repeating pattern across the interface. The pattern for the preferred embodiment is 100010001000” <i>Id.</i> at [0038]: “The SAP next signals the receiver calibration logic to perform the fast initialization procedure. Since the data deskew has already been done, it is not necessary to perform this lengthy step during recalibration. The fast initialization procedure consists of starting the state machine at the final clock calibration step.” <i>Id.</i> at [0039]: “This recalibration can be understood by looking to FIG. 3. Notice that the normal calibration sequences of the initial clock calibration sequence, 34, data deskew sequence, 35, and final clock calibration sequence, 36, still exist. In addition, at the time of recalibration, a new wait state, 41, is used, which allows the calibration logic to proceed directly to the final clock calibration sequence, 36.” <i>Id.</i> at [0040]: “The clock delays are reset as part of the final clock calibration sequence. By adding delay to the clock path, the clock calibration logic is able to find the optimum data capture time for the interface. Since the calibration is done periodically, this window gets reoptimized every time recalibration occurs. Also, if the cycle time, voltage, or other changes are made to the environment for testing, the interface can recalibrate to the new conditions. However, the data deskew sequence does not need to be run since the deskew settings are not likely to change since the data nets will tend to track over time.”
--	--

	<i>Id.</i> at [0042]-[0048]: “To summarize, the preferred method includes the steps of: 1. Putting the interface or system into a wait state, 2. Setting interface fences (block interface from being used by system operations) 3. Resetting interface state machine 4. Turning on the driver calibration pattern 5. Calibrating clock 6. Turning off driver calibration pattern . . .” <i>Id.</i> at [0051]: “Although the preferred embodiment recalculates the flag0, flag1, and flag2 values, there are other methods that can be used to recalibrate the clock. For instance, if it is known that only the frequency changes, the change in frequency can be calculated by additional hardware and half the difference can be applied as a delay shift. Although this would be faster, we excluded this form of recalibration due to the complexity and the fact that the preferred embodiment was fast enough to avoid the system hangs that the full calibration would cause. Also, the preferred embodiment uses the same sequences used in the original.” Greeff similarly discloses increasing and decreasing parameters used in the active termination circuit to account for variations in process, voltage, or temperature. Greeff, at 7:66-8:5.
--	---

Claim 19	
19: The system of claim 18, further comprising circuitry to store a first expected pattern and a second expected pattern, and circuitry to compare the first expected pattern with a pattern received from an external component in association with the first calibration, and to compare the second expected pattern with a pattern received from the external component in association with the second calibration.	<u>Discussion</u> As I discussed in Ground 1 at claim 9 and claim 16, Meaney discloses circuitry to store a first expected pattern and a second expected pattern, and circuitry to compare the first expected pattern with a pattern received from an external component in association with the first calibration, and to compare the second expected pattern with a pattern received from the external component in association with the second calibration. <u>Other Relevant Citations</u> Meany at [0008]: “The invention allows for the re-calibration of the interface at periodic intervals with minor disruption. This invention is applicable to the SMP computer systems sold by IBM and others.” <i>Id.</i> at [0009]-[0016]: “The steps for re-calibration involve: 1. Putting the system of the interface into a wait state, 2. Performing a fast initialization process for calibration, and 3. Taking the system of the interface out of a wait state[.] The fast initialization process in the preferred embodiment is similar to the prior art, except that there is only one clock centering step, instead of two. Also, the data deskew step is skipped since that step has already been performed as part of the original initialization process. The step of putting the system into the wait state can be done through several methods as described in the referenced prior art. One way is to idle all the processors while having one processor oversee the recalibration sequence. This same processor may take the system out of the wait state as well. By keeping the interface from being used, the driving circuits can exclusively send the calibration pattern and the calibration logic can re-center the clock under any new environmental conditions or circuit changes. The invention can be triggered periodically. This way, the changes to circuit or environmental characteristics over time do not adversely affect the operation of the interface.” <i>Id.</i> at [0037]: “Then, the SAP uses a hardware interface protocol to read and write registers in the interface logic directly. The SAP writes the Fence registers to make sure the interfaces are truly unused. Then, it loads the driver side calibration flag. This flag signals hardware to drive a repeating pattern across the interface. The pattern for the preferred embodiment is 100010001000”

	<i>Id.</i> at [0038]: “The SAP next signals the receiver calibration logic to perform the fast initialization procedure. Since the data deskew has already been done, it is not necessary to perform this lengthy step during recalibration. The fast initialization procedure consists of starting the state machine at the final clock calibration step.” <i>Id.</i> at [0039]: “This recalibration can be understood by looking to FIG. 3. Notice that the normal calibration sequences of the initial clock calibration sequence, 34, data deskew sequence, 35, and final clock calibration sequence, 36, still exist. In addition, at the time of recalibration, a new wait state, 41, is used, which allows the calibration logic to proceed directly to the final clock calibration sequence, 36.” <i>Id.</i> at [0041]: “When the calibration is complete, the SAP turns off the driver calibration pattern, resets the interface fences and takes the processors out of the wait state.” <i>Id.</i> at [0042]-[0044]: “To summarize, the preferred method includes the steps of: 1. Putting the interface or system into a wait state, 2. Setting interface fences (block interface from being used by system operations) . . .” <i>Id.</i> at claim 5: “In an SMP computer system having an source-synchronous interface, the method for re-calibration of the interface at periodic intervals comprising the steps of: a. putting the system of the interface into a wait state, b. performing a fast initialization process for calibration, c. taking the system of the interface out of said wait state.” <i>Id.</i> at claim 7: “The method according to claim 6 wherein said wait state keeps the interface from being used for processing steps other than re-calibration and sending a calibration pattern and allowing claibration [sic] logic to re-center the clock applicable to the interface to compensate for new environmental conditions and circuit changes.” <i>Id.</i> at Fig. 3:
--	--

E.I. CLOCK RECEIVER STATE MACHINE DIAGRAM

7/89

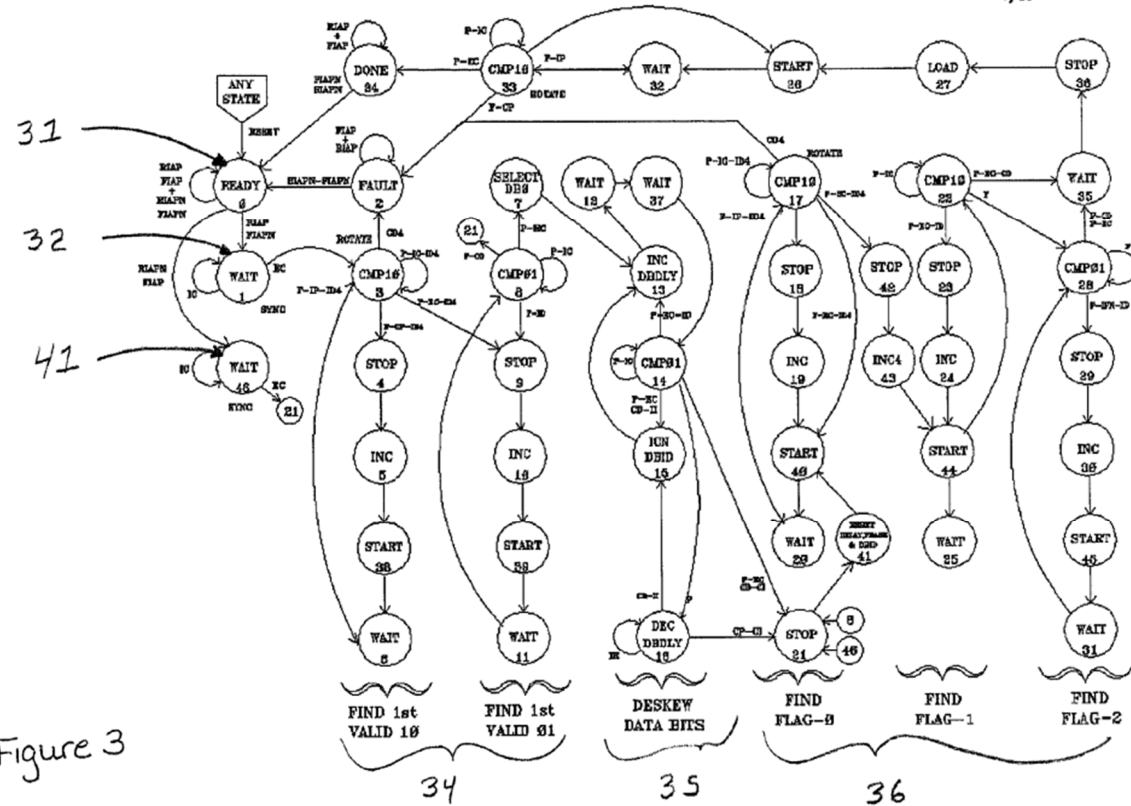


Figure 3

For example, U.S. Patent Publication No. 2002/0066001 (“Olarig”) discloses calibration timers that include time intervals for conducting a calibration. Olarig, Ex. 1010 at [0042]. When the timer of Olarig expires, the calibration is performed. *Id.*

GROUND 4: Meaney, Nguyen, and Greeff in view of the knowledge of a POSITA.

A. Motivation to Combine

In addition to being motivated to combine Meaney with Nguyen as described in the motivation to combine for Ground 2, a POSITA would have been motivated to further combine Meaney and Nguyen with Greeff with a reasonable expectation of success for the same reasons discussed above in the motivation to combine for Ground 3.

Claim 4	
4: The method of claim 1, wherein the parameter is a link termination resistance.	<i>See</i> Ground 3, claim 4.
Claim 5	
5: The method of claim 1, wherein the parameter is a driver strength.	<i>See</i> Ground 3, claim 5.
Claim 13	
13: The system of claim 10, wherein the parameter is a link termination resistance.	<i>See</i> Ground 3, claim 13.
Claim 14	
14: The system of claim 10, wherein the parameter is a driver strength for a driver of the receive component.	<i>See</i> Ground 3, claim 14.

Claim 17	
17[pre]: A system, comprising:	<i>See</i> Ground 3, claim 17[pre].
17[a]: circuitry to store an existing value representative of at least one of a termination resistance and a driver strength to be applied in association with data communicated across a channel;	As I discussed in Ground 2 at 10[b], Meaney and Nguyen disclose circuitry to store an existing value of a parameter. As I discussed in Ground 3 at 17[a], Meaney and Greeff disclose termination resistance to be applied in association with data communicated across a channel.
17[b]: circuitry to perform a first calibration at system initialization, to identify a first value to be initially applied as the existing value,	<u>Discussion</u> As I discussed in Ground 3 at 17[a], Meaney and Greeff disclose circuitry to perform a first calibration at system initialization, to identify a first value to be initially applied as the existing value. To the extent Patent Owner argues that Meaney and Greeff do not disclose circuitry to perform a first calibration, Nguyen discloses calibration circuitry, as discussed above in Ground 2 at 1[pre] and claim 16. <u>Other Relevant Citations</u> Meaney at Abstract: “An SMP computer system has an apparatus and method for recalibrating a self-timed, source-synchronous, pipelined interface while the computer system is running. The apparatus allows for quiescing the interface (ie. idling the processors to allow for no data transfers), raising fences (blocking interfaces), allowing for a quick clock centering recalibration step, and then unfencing and unquiescing to allow for the use of the interface again. The recalibration allows for compensating for drift over time on the interface to compensate for temperature, voltage, cycle time, and end-of-life degradation without bringing down and restarting the system.” <i>Id.</i> at [0001]: “This invention relates to symmetrical computer systems, and particularly to an apparatus and method for recalibrating an [sic] source-synchronous pipelined interface with minimal impact to a running system, allowing for the system to remain operational, even in the midst of environmental drift or degradation.”

	<i>Id.</i> at [0003]-[0004]: “In an SMP (symmetric multiprocessing) computer system with high-level packaging, it is often necessary to allow latencies on cards, wires, or boards that exceed the cycle times of the transferred data. In order to allow these transfers to take place, source-synchronous pipelined interfaces have been used in the s/390 machines of IBM. This allows for the proper capturing of the data within a small window, or ‘eye’. In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.” <i>Id.</i> at [0006]: “The prior s/390 IBM machines only handled the first calibration. There was no way to re-calibrate an interface once it was up and running. Therefore, the cumulative degradations could eventually cause a system failure.” <i>Id.</i> at [0007]: “Also, in a test environment, we often apply environmental stress to a machine as well as frequency changes. We often want to run a hardware/architecture exerciser (like SAK or PCX) under several environments (about 16 voltage corners as well as cycling up or down the frequency), the interface often fails without a re-calibration. If the system is not recalibrated under the new environment, the system must be restarted for every test. Therefore, our test time is jeopardized.” <i>Id.</i> at [0015]-[0016]: “By keeping the interface from being used, the driving circuits can exclusively send the calibration pattern and the calibration logic can re-center the clock under any new environmental conditions or circuit changes. The invention can be triggered periodically. This way, the changes to circuit or environmental characteristics over time do not adversely affect the operation of the interface.” <i>Id.</i> at [0020]-[0021]: “FIG. 2 illustrates the prior art state diagram for performing the initial calibration of the hardware as performed in conjunction with the prior art; FIG. 3 illustrates the preferred embodiment state diagrams for supporting the initial calibration as well as the recalibration of the hardware as performed in conjunction with the invention.” <i>Id.</i> at [0023]: “Turning to FIG. 1, notice that there is driver logic, 11, used to drive interface bus, 12, and clock, 13, which feeds receiver calibration logic, 14. Said receiver logic, 14, consisting of state machine logic, 15, clock calibrating hardware, 16, and data de-skew hardware 17. The output of receiver calibration logic, 14, transfers data onto receiver output bus, 18, which connects to system logic, 19.”
--	--

	<i>Id.</i> at [0024] : “In the prior art, the system was always recalibrated from a stopped position. The system would be scan-initialized to a reset state and the calibration would begin when the clocks were started.”⁵; “ <i>Id.</i> at [0038]: “The SAP next signals the receiver calibration logic to perform the fast initialization procedure. Since the data deskew has already been done, it is not necessary to perform this lengthy step during recalibration. The fast initialization procedure consists of starting the state machine at the final clock calibration step.” <i>Id.</i> at [0040]: “The clock delays are reset as part of the final clock calibration sequence. By adding delay to the clock path, the clock calibration logic is able to find the optimum data capture time for the interface. Since the calibration is done periodically, this window gets reoptimized every time recalibration occurs. Also, if the cycle time, voltage, or other changes are made to the environment for testing, the interface can recalibrate to the new conditions. However, the data deskew sequence does not need to be run since the deskew settings are not likely to change since the data nets will tend to track over time.” <i>Id.</i> at [0051]: “Although the preferred embodiment recalculates the flag0, flag1, and flag2 values, there are other methods that can be used to recalibrate the clock. For instance, if it is known that only the frequency changes, the change in frequency can be calculated by additional hardware and half the difference can be applied as a delay shift. Although this would be faster, we excluded this form of recalibration due to the complexity and the fact that the preferred embodiment was fast enough to avoid the system hangs that the full calibration would cause. Also, the preferred embodiment uses the same sequences used in the original.” <i>Id.</i> at claim 12: “The method according to claim 10 wherein the recalibration stem includes re-calculating the clock frequency of the interface against the current hardware and re-applying the clock frequency calculation to the clock delay to re-center the clock when the machine is being cycled down to failure and the major change needing re-calibration is cycle time.” <i>Id.</i> at Fig. 1:
--	--

⁵ Meaney refers to this discussion as prior art, but is clear that the preferred method is “similar to the prior art calibration sequence” but includes “additional steps.” Ex. 1004 at [0035]. Thus, Meaney is clear that the initial calibration of the clock would be part of the preferred method as well.

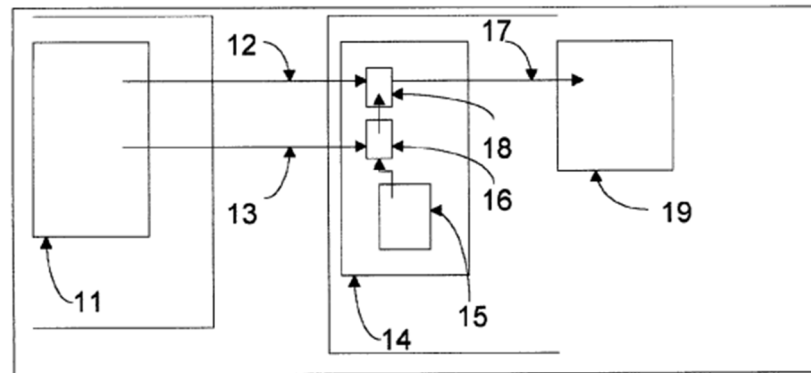


FIGURE 1.

Id. at Fig. 2:

E.I. CLOCK RECEIVER STATE MACHINE DIAGRAM

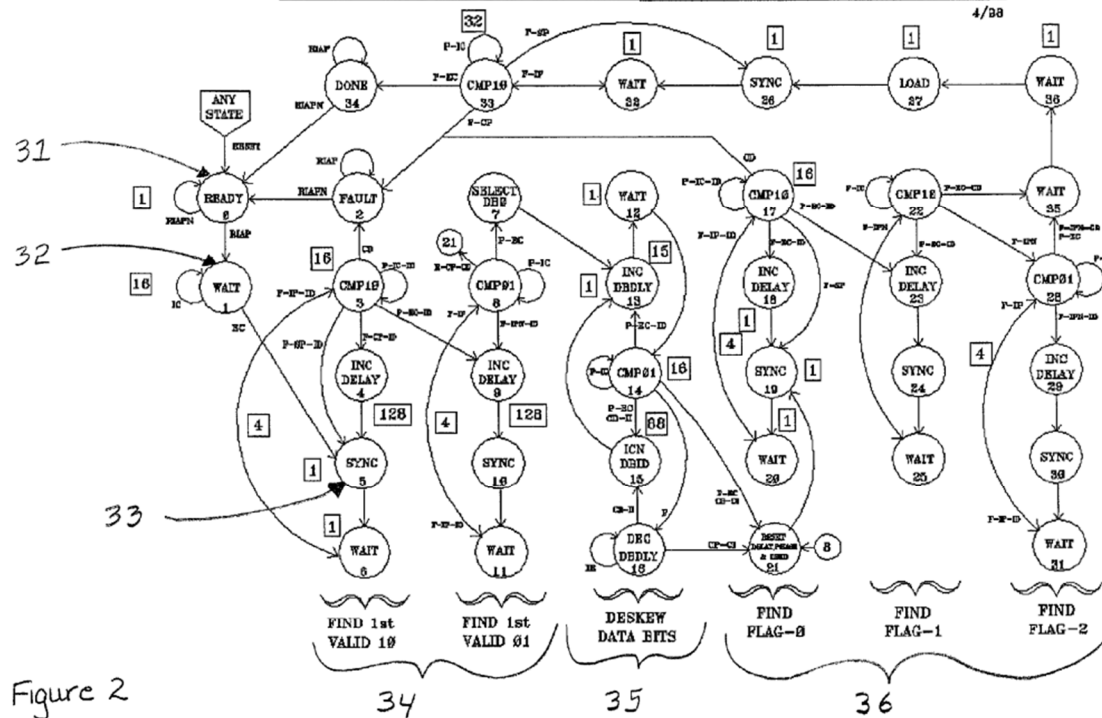


Figure 2

Id. at Fig. 3:

7/89

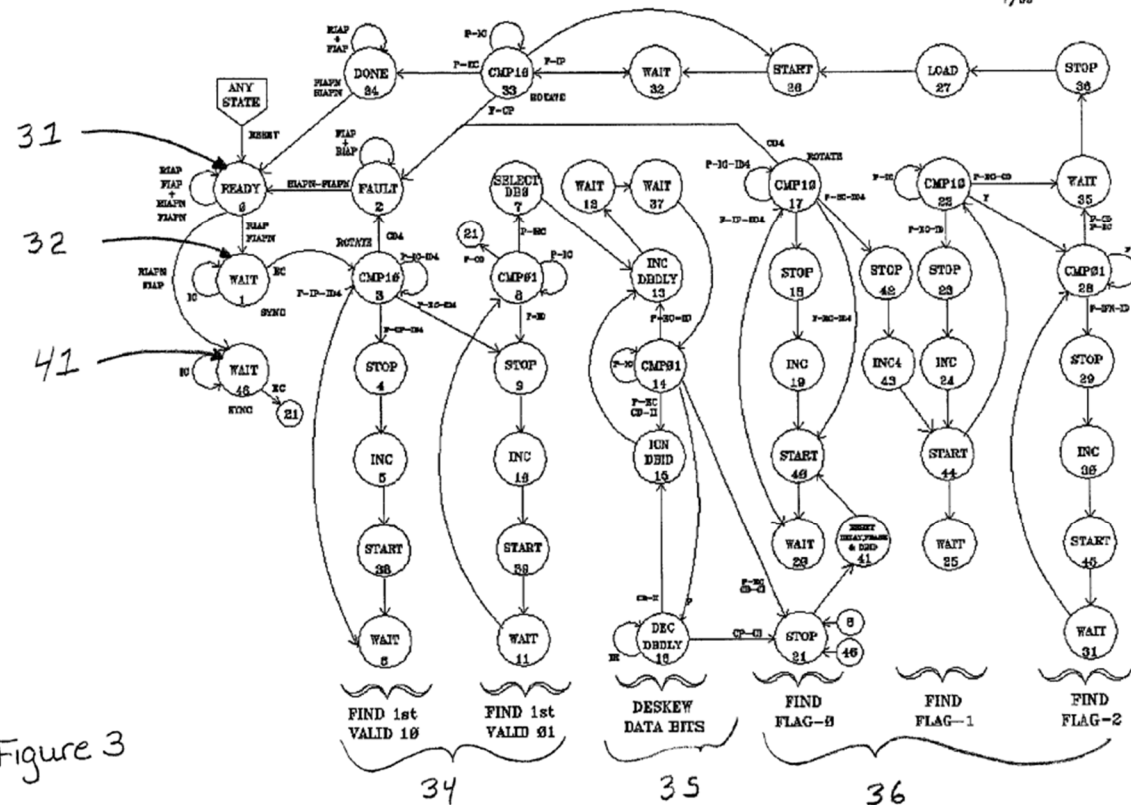


Figure 3

Nguyen teaches a receiving unit 115 that receives a signal 110 from the transmitting unit 110. Nguyen at [0030]; Fig. 1 (annotated below).

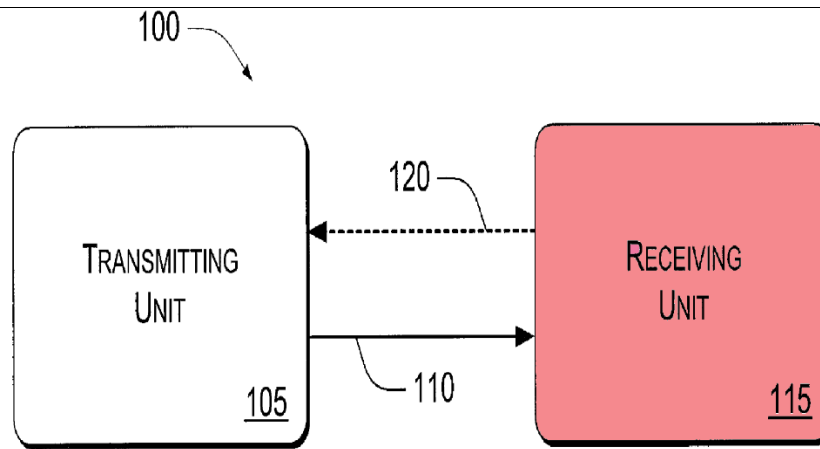


Fig. 1

Nguyen discloses, for example, that the receiving unit may be part of an integrated circuit or printed circuit board, and teaches that the transmitted signal 110 may be a digital signal. Nguyen at [0031], [0032], [0047]. Nguyen also gives examples of specific circuitry that could be implemented in the receiving unit. Nguyen, Fig. 4A:

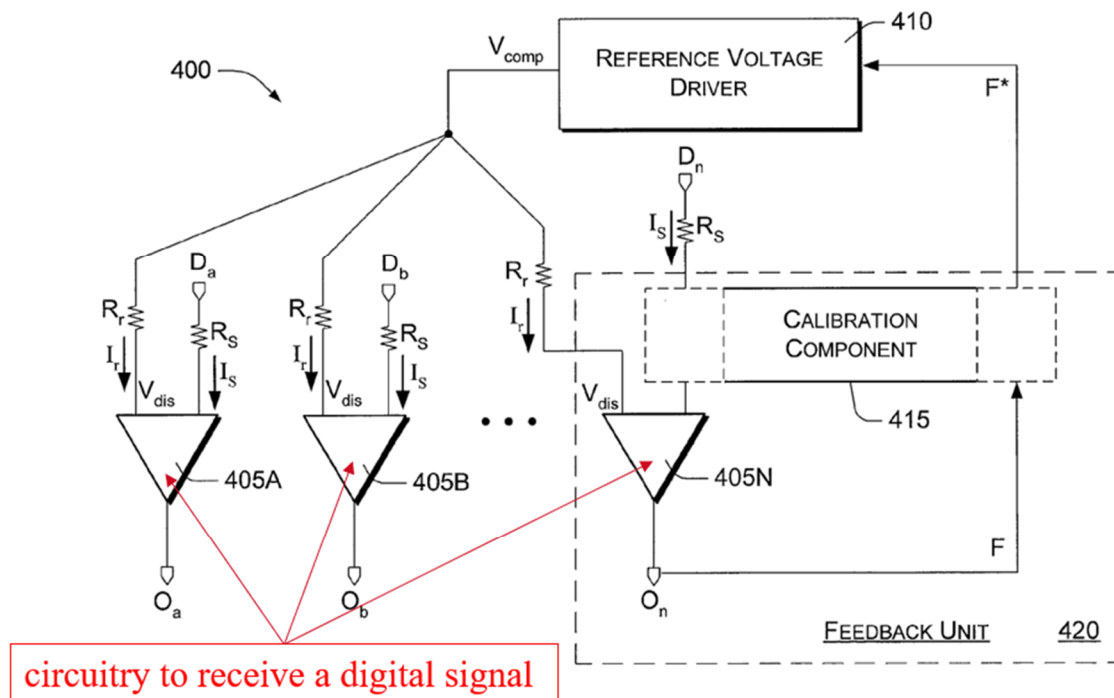


Fig. 4A

Greeff teaches calibrating parameters such as termination resistance prior to the first use of the system, which is referred to as “static calibration.” Greeff at 4:10-13, 7:52-61. As shown in Fig. 2 (reproduced below), Greeff teaches active termination circuitry that would be used to calibrate the termination resistance (“RTERM”). *Id.* at 4:10-13, 7:52-61, Fig. 2.

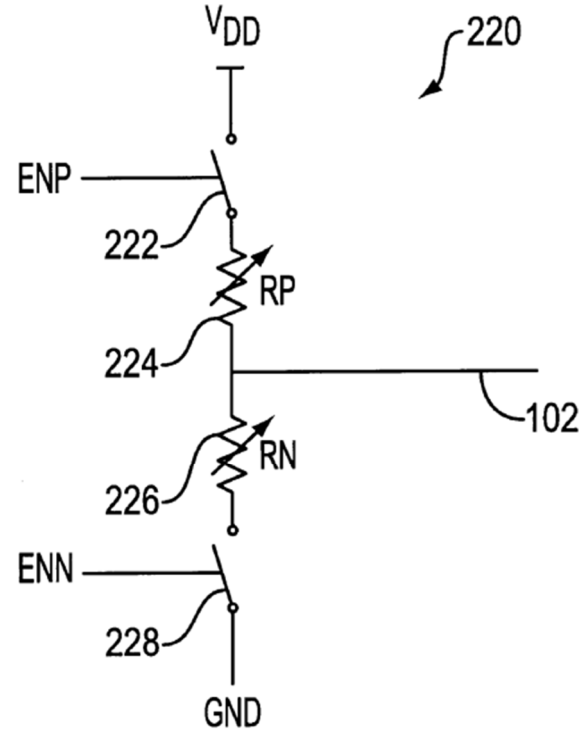


FIG. 2

For example, Greeff explains that the “two termination resistances RP, RN would be set during the calibration process to provide substantially optimal termination,” and that the termination enable signals ENP and ENN would be generated during calibration as well. *Id.* at 6:3-5.

17[c]: the first calibration to be performed during a first calibration interval prior to normal system operation; and	See Ground 3 at 17[c]
17[d]: circuitry to periodically perform a second calibration, to update the existing value;	<u>Discussion</u> As I discussed in Ground 3 at 17[a], Meaney and Greeff disclose circuitry to perform a first calibration at system initialization, to identify a first value to be initially applied as the existing value. To the extent Patent Owner argues that Meaney and Greeff do not disclose circuitry to perform a first calibration, Nguyen discloses calibration circuitry, as discussed above in Ground 2 at 1[pre] and claim 16. <u>Other Relevant Citations</u> Meaney at Abstract: “An SMP computer system has an apparatus and method for recalibrating a self-timed, source-synchronous, pipelined interface while the computer system is running. The apparatus allows for quiescing the interface (ie. idling the processors to allow for no data transfers), raising fences (blocking interfaces), allowing for a quick clock centering recalibration step, and then unfencing and unquiescing to allow for the use of the interface again. The recalibration allows for compensating for drift over time on the interface to compensate for temperature, voltage, cycle time, and end-of-life degradation without bringing down and restarting the system.” <i>Id.</i> at [0001]: “This invention relates to symmetrical computer systems, and particularly to an apparatus and method for recalibrating an [sic] source-synchronous pipelined interface with minimal impact to a running system, allowing for the system to remain operational, even in the midst of environmental drift or degradation.” <i>Id.</i> at [0003]-[0004]: “In an SMP (symmetric multiprocessing) computer system with high-level packaging, it is often necessary to allow latencies on cards, wires, or boards that exceed the cycle times of the transferred data. In order to allow these transfers to take place, source-synchronous pipelined interfaces have been used in the s/390 machines of IBM. This allows for the proper capturing of the data within a small window, or ‘eye’. In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.”

	<i>Id.</i> at [0007]: “Also, in a test environment, we often apply environmental stress to a machine as well as frequency changes. We often want to run a hardware/architecture exerciser (like SAK or PCX) under several environments (about 16 voltage corners as well as cycling up or down the frequency), the interface often fails without a re-calibration. If the system is not recalibrated under the new environment, the system must be restarted for every test. Therefore, our test time is jeopardized.” <i>Id.</i> at [0009]-[0016]: “The steps for re-calibration involve: 1. Putting the system of the interface into a wait state, 2. Performing a fast initialization process for calibration, and 3. Taking the system of the interface out of a wait state[.] The fast initialization process in the preferred embodiment is similar to the prior art, except that there is only one clock centering step, instead of two. Also, the data deskew step is skipped since that step has already been performed as part of the original initialization process. The step of putting the system into the wait state can be done through several methods as described in the referenced prior art. One way is to idle all the processors while having one processor oversee the recalibration sequence. This same processor may take the system out of the wait state as well. By keeping the interface from being used, the driving circuits can exclusively send the calibration pattern and the calibration logic can re-center the clock under any new environmental conditions or circuit changes. The invention can be triggered periodically. This way, the changes to circuit or environmental characteristics over time do not adversely affect the operation of the interface.” <i>Id.</i> at [0020]-[0021]: “FIG. 2 illustrates the prior art state diagram for performing the initial calibration of the hardware as performed in conjunction with the prior art; FIG. 3 illustrates the preferred embodiment state diagrams for supporting the initial calibration as well as the recalibration of the hardware as performed in conjunction with the invention.” <i>Id.</i> at [0023]: “Turning to FIG. 1, notice that there is driver logic, 11, used to drive interface bus, 12, and clock, 13, which feeds receiver calibration logic, 14. Said receiver logic, 14, consisting of state machine logic, 15, clock calibrating hardware, 16, and data de-skew hardware 17. The output of receiver calibration logic, 14, transfers data onto receiver output bus, 18, which connects to system logic, 19.”
--	---

Id. at [0040]: “The clock delays are reset as part of the final clock calibration sequence. By adding delay to the clock path, the clock calibration logic is able to find the optimum data capture time for the interface. Since the calibration is done periodically, this window gets reoptimized every time recalibration occurs. Also, if the cycle time, voltage, or other changes are made to the environment for testing, the interface can recalibrate to the new conditions. However, the data deskew sequence does not need to be run since the deskew settings are not likely to change since the data nets will tend to track over time.”

Id. at [0051]: “Although the preferred embodiment recalculates the flag0, flag1, and flag2 values, there are other methods that can be used to recalibrate the clock. For instance, if it is known that only the frequency changes, the change in frequency can be calculated by additional hardware and half the difference can be applied as a delay shift. Although this would be faster, we excluded this form of recalibration due to the complexity and the fact that the preferred embodiment was fast enough to avoid the system hangs that the full calibration would cause. Also, the preferred embodiment uses the same sequences used in the original.”

Id. at claim 12: “The method according to claim 10 wherein the recalibration stem includes re-calculating the clock frequency of the interface against the current hardware and re-applying the clock frequency calculation to the clock delay to re-center the clock when the machine is being cycled down to failure and the major change needing re-calibration is cycle time.”

Id. at Fig. 1:

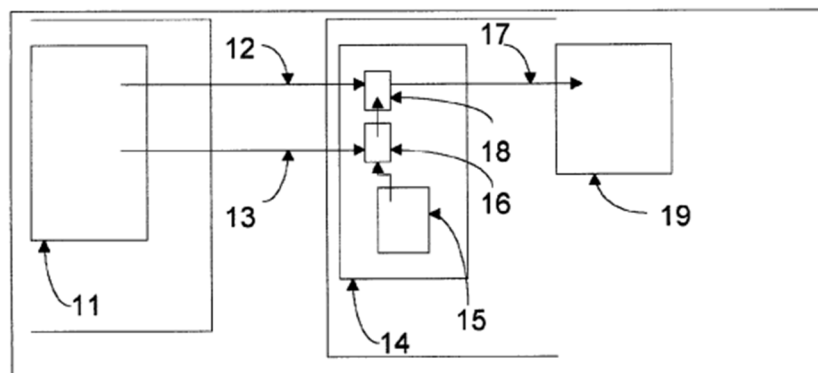


FIGURE 1.

Id. at Fig. 2:

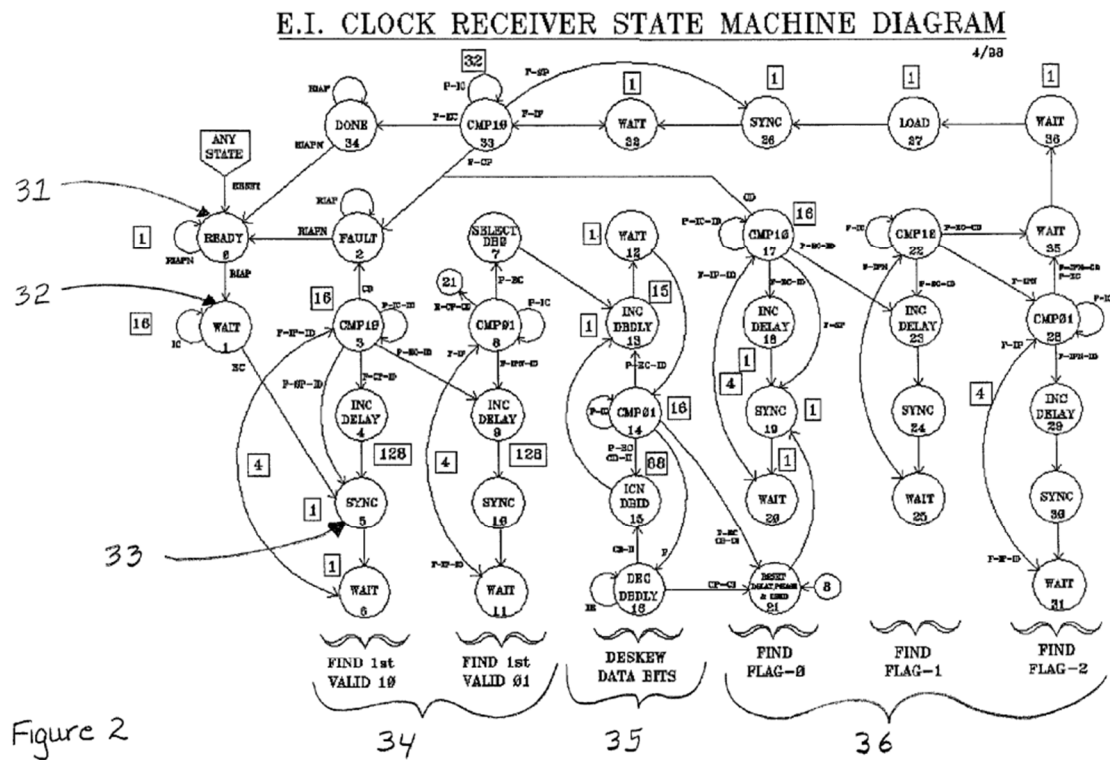
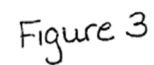


Figure 2

Id. at Fig. 3:

7/89



138

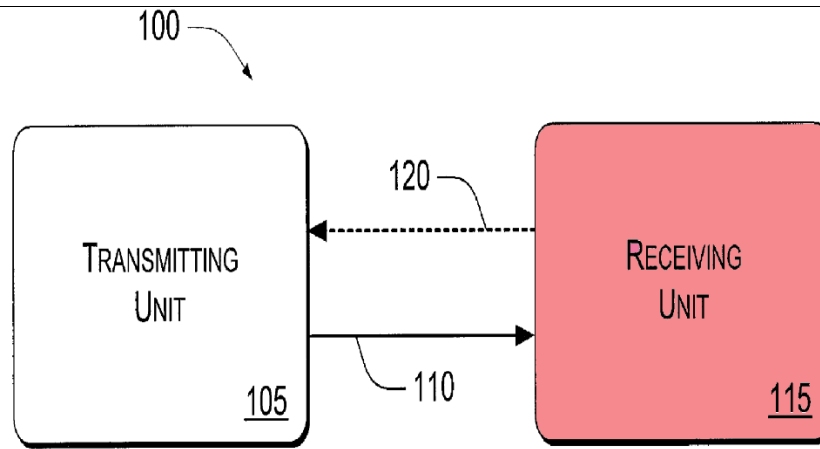


Fig. 1

Nguyen discloses, for example, that the receiving unit may be part of an integrated circuit or printed circuit board, and teaches that the transmitted signal 110 may be a digital signal. Nugyen at [0031], [0032], [0047]. Nguyen also gives examples of specific circuitry that could be implemented in the receiving unit. Nguyen, Fig. 4A:

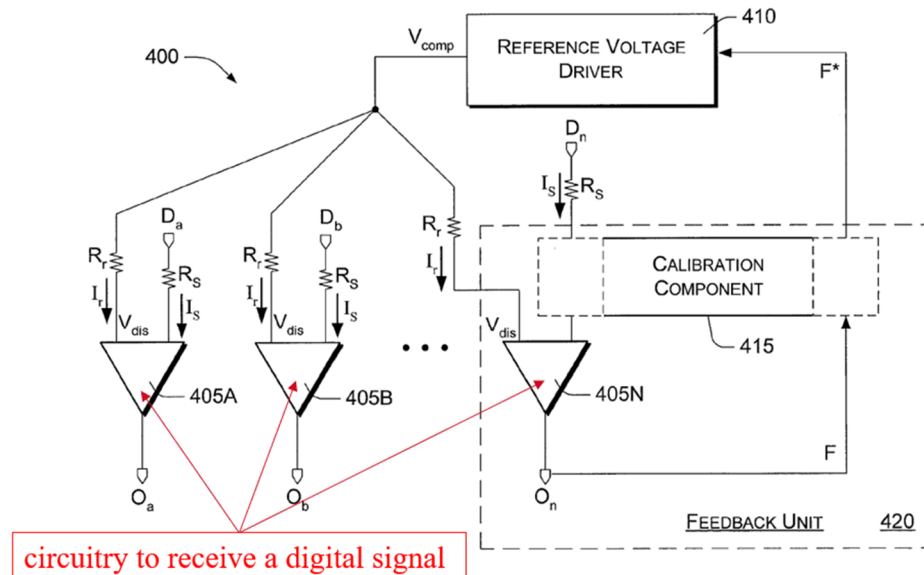


Fig. 4A

Greeff teaches calibrating parameters such as termination resistance prior to the first use of the system, which is referred to as “static calibration.” Greeff at 4:10-13, 7:52-61. As shown in Fig. 2 (reproduced below), Greeff teaches active termination circuitry that would be used to calibrate the termination resistance (“RTERM”). *Id.* at 4:10-13, 7:52-61, Fig. 2. Greeff further teaches recalibrating termination resistance during operation of the system, for example when the value changes due to temperature or supply voltage changes, which is referred to as “dynamic calibration.” Greeff at 7:52-65.

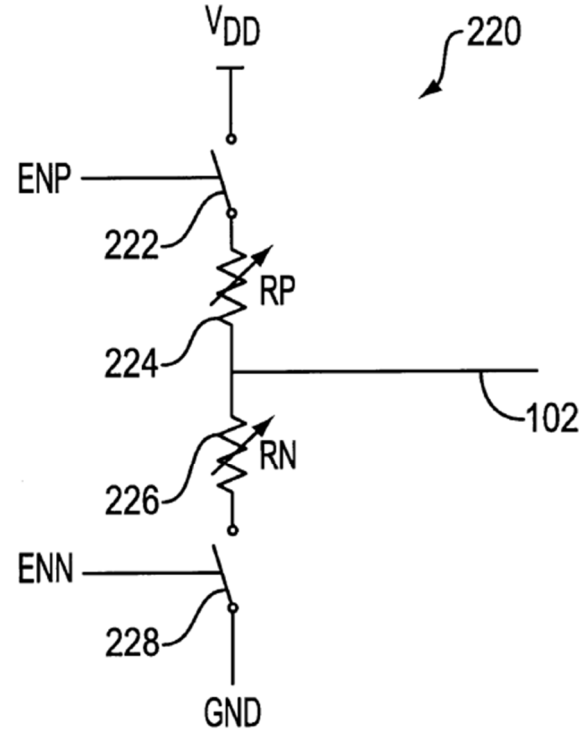


FIG. 2

17[e]: wherein the existing value is initially set and periodically updated responsive to the first calibration and the second calibration, respectively, and	<i>See</i> Ground 3 at 17[e].
17[f]: wherein the second calibration is constrained to be performed during a time period that is shorter than a time period of the first calibration; and	<i>See</i> Ground 3 at 17[f].
17[g]: wherein the system further comprises a receive component, the at least one to be applied to permit proper reception of a digital signal to be communicated across a communications channel.	<i>See</i> Ground 3 at 17[g].

Claim 18	
18: The system of claim 17, wherein the circuitry to perform the second calibration is, responsive to the second calibration, to perform one of an increment operation or a decrement operation upon the existing value if the desired value has drifted from the existing value by more than a delta value relative to an existing value.	<u>Discussion</u> As I discussed above in Ground 3 at claims 18 and 19, claim 18 is obvious over the combination of Meaney and Greeff, and claim 19 is obvious over Meaney. <u>Other Relevant Citations</u> Meany at [0004]: “In order to capture this data, there is usually a calibration or learning period. A known data pattern is sent across the interface and the receiver compensates for the various package tolerances using calibration techniques.” <i>Id.</i> at [0005]: “However, over time, these large-latency interfaces can drift due to environmental changes like voltage, temperature, end-of-life degradation, etc.” <i>Id.</i> at [0008]: “The invention allows for the re-calibration of the interface at periodic intervals with minor disruption. This invention is applicable to the SMP computer systems sold by IBM and others.” <i>Id.</i> at [0016]: “The invention can be triggered periodically. This way, the changes to circuit or environmental characteristics over time do not adversely affect the operation of the interface.” <i>Id.</i> at [0017]: “Also, the preferred embodiment allows for the proactive running of the recalibration sequence based on a trigger event. For instance, when ECC is used on an interface and a correctable error (CE) is detected, the data is corrected. The CE can trigger an event to recalibrate the interface. This should protect against future CEs, if the original CE was based on a degraded interface.” <i>Id.</i> at [0033]: “The limitation with this sequence is that, once the interfaces are used, there is no means to get the interface recalibrated without resetting the state machine and starting the entire sequence over.” <i>Id.</i> at [0037]: “Then, the SAP uses a hardware interface protocol to read and write registers in the interface logic directly. The SAP writes the Fence registers to make sure the interfaces are truly unused. Then, it loads the driver side calibration flag. This flag signals hardware to drive a repeating pattern across the interface. The pattern for the preferred embodiment is 100010001000”

	<i>Id.</i> at [0038]: “The SAP next signals the receiver calibration logic to perform the fast initialization procedure. Since the data deskew has already been done, it is not necessary to perform this lengthy step during recalibration. The fast initialization procedure consists of starting the state machine at the final clock calibration step.” <i>Id.</i> at [0039]: “This recalibration can be understood by looking to FIG. 3. Notice that the normal calibration sequences of the initial clock calibration sequence, 34, data deskew sequence, 35, and final clock calibration sequence, 36, still exist. In addition, at the time of recalibration, a new wait state, 41, is used, which allows the calibration logic to proceed directly to the final clock calibration sequence, 36.” <i>Id.</i> at [0040]: “The clock delays are reset as part of the final clock calibration sequence. By adding delay to the clock path, the clock calibration logic is able to find the optimum data capture time for the interface. Since the calibration is done periodically, this window gets reoptimized every time recalibration occurs. Also, if the cycle time, voltage, or other changes are made to the environment for testing, the interface can recalibrate to the new conditions. However, the data deskew sequence does not need to be run since the deskew settings are not likely to change since the data nets will tend to track over time.” <i>Id.</i> at [0042]-[0048]: “To summarize, the preferred method includes the steps of: 1. Putting the interface or system into a wait state, 2. Setting interface fences (block interface from being used by system operations) 3. Resetting interface state machine 4. Turning on the driver calibration pattern 5. Calibrating clock 6. Turning off driver calibration pattern . . .” <i>Id.</i> at [0051]: “Although the preferred embodiment recalculates the flag0, flag1, and flag2 values, there are other methods that can be used to recalibrate the clock. For instance, if it is known that only the frequency changes, the change in frequency can be calculated by additional hardware and half the difference can be applied as a delay shift. Although this would be faster, we excluded this form of recalibration due to the complexity and the fact that the preferred embodiment was fast enough to avoid the system hangs that the full calibration would cause. Also, the preferred embodiment uses the same sequences used in the original.”
--	--

	Nguyen discloses adjusting a parameter of a digital signal receiver, compensated voltage, by applying an offset to a previously calibrated value. Nguyen at [0099] (“This calibrated value may be read from the register 710 and an offset value may be added. The resultant summed value may then be programmed back into the register 710 and used during normal operation of the device”). Specifically, Nguyen discloses determining a calibrated value, then adding or subtracting an offset value. <i>Id.</i> at [0099]. The offset value may be determined at design time or determined dynamically as part of an initialization or calibration procedure. <i>Id.</i> Nguyen also teaches that calibration may be repeated to correct for drifts in the calibrated parameter, or in response to “a predetermined number of detected (e.g., bit or byte) errors.” Nguyen at [0094], [0147]. A POSITA would have understood a predetermined number of detected errors to represent a delta value that represents a parameter drift. For these reasons, a POSITA would have understood that Nguyen teaches recalibration “if a desired value of the parameter has drifted from the existing value by more than a delta value.” Greeff similarly discloses increasing and decreasing parameters used in the active termination circuit to account for variations in process, voltage, or temperature. Greeff, at 7:66-8:5.
--	--

Claim 19	
19: The system of claim 18, further comprising circuitry to store a first expected pattern and a second expected pattern, and circuitry to compare the first expected pattern with a pattern received from an external component in association with the first calibration, and to compare the second expected pattern with a pattern received from the external component in association with the second calibration.	<u>Discussion</u> As I discussed above in Ground 3 at claims 18 and 19, claim 18 is obvious over the combination of Meaney and Greeff, and claim 19 is obvious over Meaney. <u>Other Relevant Citations</u> Meany at [0008]: “The invention allows for the re-calibration of the interface at periodic intervals with minor disruption. This invention is applicable to the SMP computer systems sold by IBM and others.” <i>Id.</i> at [0009]-[0016]: “The steps for re-calibration involve: 1. Putting the system of the interface into a wait state, 2. Performing a fast initialization process for calibration, and 3. Taking the system of the interface out of a wait state[.] The fast initialization process in the preferred embodiment is similar to the prior art, except that there is only one clock centering step, instead of two. Also, the data deskew step is skipped since that step has already been performed as part of the original initialization process. The step of putting the system into the wait state can be done through several methods as described in the referenced prior art. One way is to idle all the processors while having one processor oversee the recalibration sequence. This same processor may take the system out of the wait state as well. By keeping the interface from being used, the driving circuits can exclusively send the calibration pattern and the calibration logic can re-center the clock under any new environmental conditions or circuit changes. The invention can be triggered periodically. This way, the changes to circuit or environmental characteristics over time do not adversely affect the operation of the interface.” <i>Id.</i> at [0037]: “Then, the SAP uses a hardware interface protocol to read and write registers in the interface logic directly. The SAP writes the Fence registers to make sure the interfaces are truly unused. Then, it loads the driver side calibration flag. This flag signals hardware to drive a repeating pattern across the interface. The pattern for the preferred embodiment is 100010001000” <i>Id.</i> at [0038]: “The SAP next signals the receiver calibration logic to perform the fast initialization procedure. Since the data deskew has already been done, it is not necessary to perform this lengthy step during recalibration. The fast initialization procedure consists of starting the state machine at the final clock calibration step.”

	<i>Id.</i> at [0039]: “This recalibration can be understood by looking to FIG. 3. Notice that the normal calibration sequences of the initial clock calibration sequence, 34, data deskew sequence, 35, and final clock calibration sequence, 36, still exist. In addition, at the time of recalibration, a new wait state, 41, is used, which allows the calibration logic to proceed directly to the final clock calibration sequence, 36.” <i>Id.</i> at [0041]: “When the calibration is complete, the SAP turns off the driver calibration pattern, resets the interface fences and takes the processors out of the wait state.” <i>Id.</i> at [0042]-[0044]: “To summarize, the preferred method includes the steps of: 1. Putting the interface or system into a wait state, 2. Setting interface fences (block interface from being used by system operations) . . .” <i>Id.</i> at claim 5: “In an SMP computer system having an source-synchronous interface, the method for re-calibration of the interface at periodic intervals comprising the steps of: a. putting the system of the interface into a wait state, b. performing a fast initialization process for calibration, c. taking the system of the interface out of said wait state.” <i>Id.</i> at claim 7: “The method according to claim 6 wherein said wait state keeps the interface from being used for processing steps other than re-calibration and sending a calibration pattern and allowing claibration [sic] logic to re-center the clock applicable to the interface to compensate for new environmental conditions and circuit changes.” <i>Id.</i> at Fig. 3:
--	---

E.I. CLOCK RECEIVER STATE MACHINE DIAGRAM

7/89

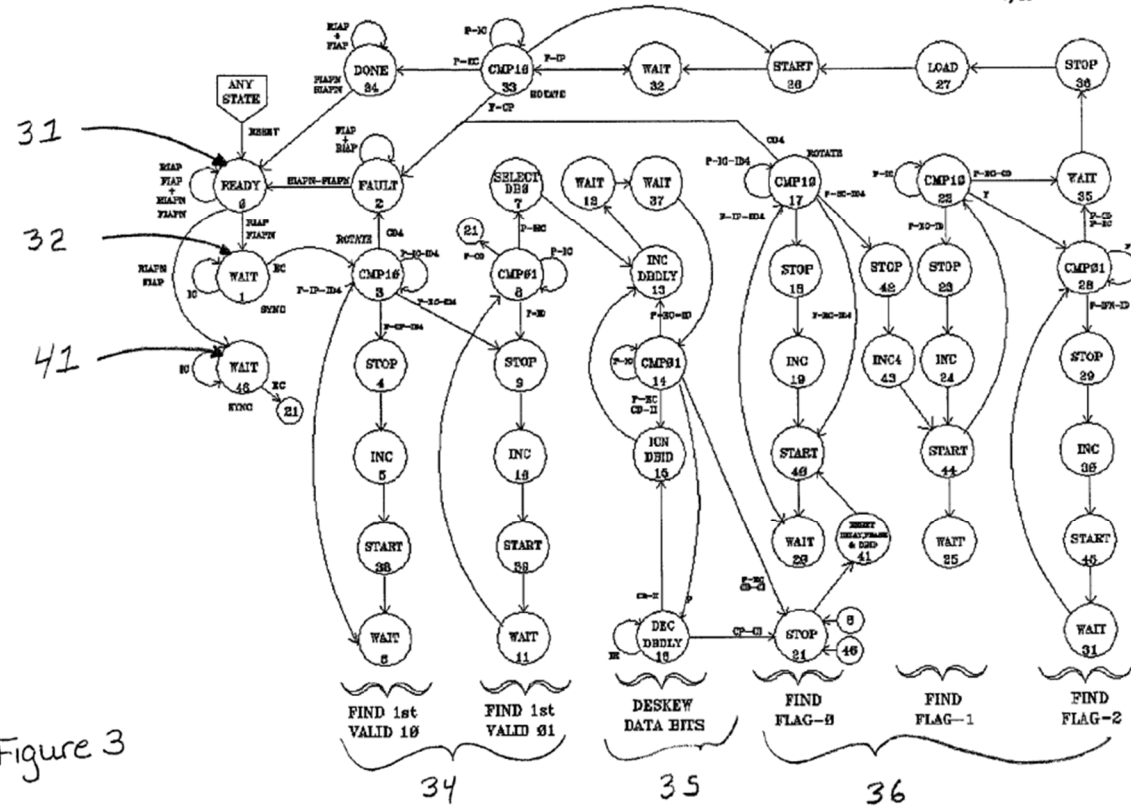


Figure 3

For example, U.S. Patent Publication No. 2002/0066001 (“Olarig”) discloses calibration timers that include time intervals for conducting a calibration. Olarig, Ex. 1010 at [0042]. When the timer of Olarig expires, the calibration is performed. *Id.*

GROUND 5A: Meaney and Allee in view of the knowledge of a POSITA.

A. Motivation to Combine

As described above in Ground 1 at claim 9, Meaney discloses calibration by comparing a known data pattern with a received data pattern. Allee provides details of a comparator that is self-calibrating and allows expansion of the functionality of digital integrated circuits. Ex. 1007, 2:35-36, 3:9-19.

A POSITA would have been motivated to combine Meaney and Allee with a reasonable expectation of success to use the comparator of Allee to compare the data patterns of Meaney to achieve the benefits of self-calibration and expanded functionality of digital circuitry. For example, Fig. 3 of Allee teaches a comparator that would have informed a POSITA of a specific way to implement the decisions disclosed in Meaney. For these reasons, a POSITA would have been motivated to combine Meaney and Allee with a reasonable expectation of success, because the teaching, suggestion, and motivation in Meaney and in Allee would have led a POSITA to modify the system of Meaney to arrive at the claimed invention.

Claim 3	
3: The method of claim 1, wherein the parameter is a voltage level for a comparator.	<u>Discussion:</u> Allee discloses computer systems including comparators that may be calibrated by a self-calibration circuit 116 to adjust the input voltage. Ex. 1007, 3:9-19, 4:1-9, 6:17-39, Fig. 1. As depicted in Fig. 1 below, Allee discloses comparators 102 that may be self-calibrated to compensate for an input offset by adjusting an input signal equal to a desired reference voltage. Ex. 1007, 3:9-19, Fig. 1 (annotated).

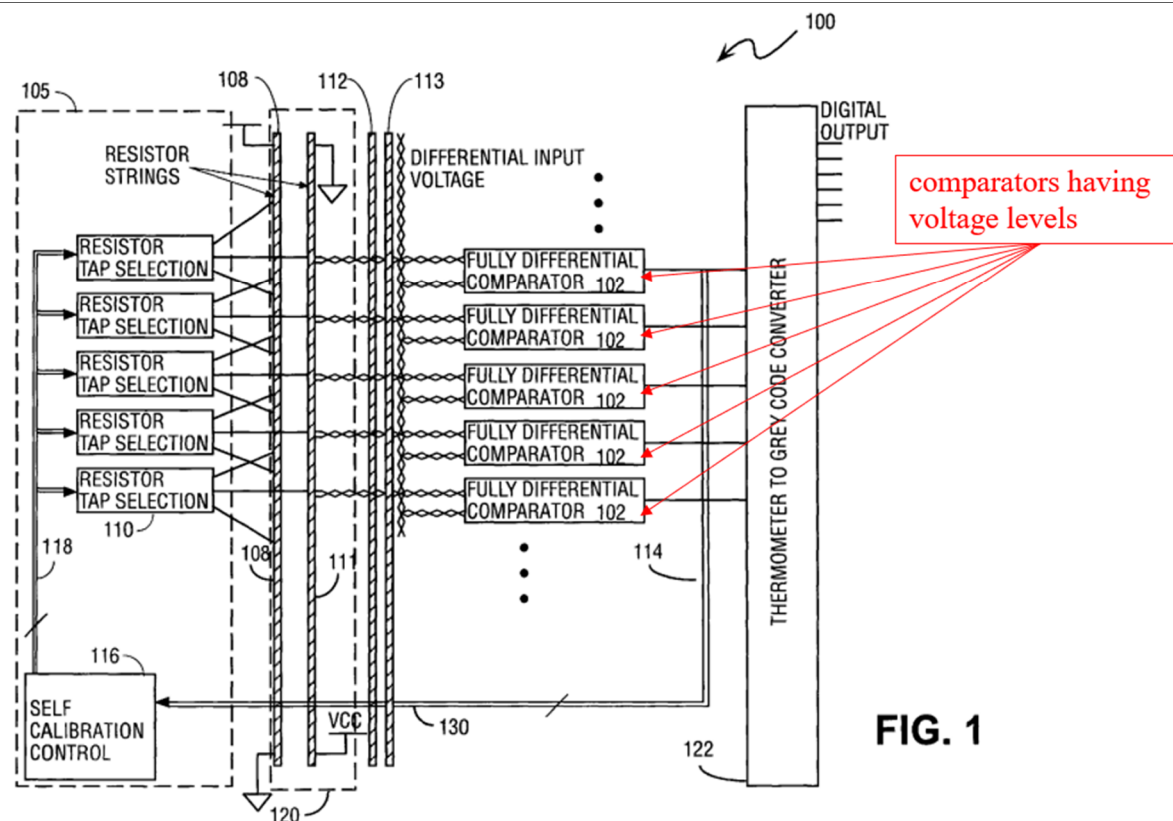


FIG. 1

Allee further discloses a self-calibration circuit 116 that calibrates the comparators 102 to adjust the voltage by adjusting the tap position of the reference voltage. Ex. 1007, 4:1-9. Allee discloses both an initial calibration process (before operation), and self-calibration of the comparator during operation. Ex. 1007, 6:17-39.

Other Relevant Citations

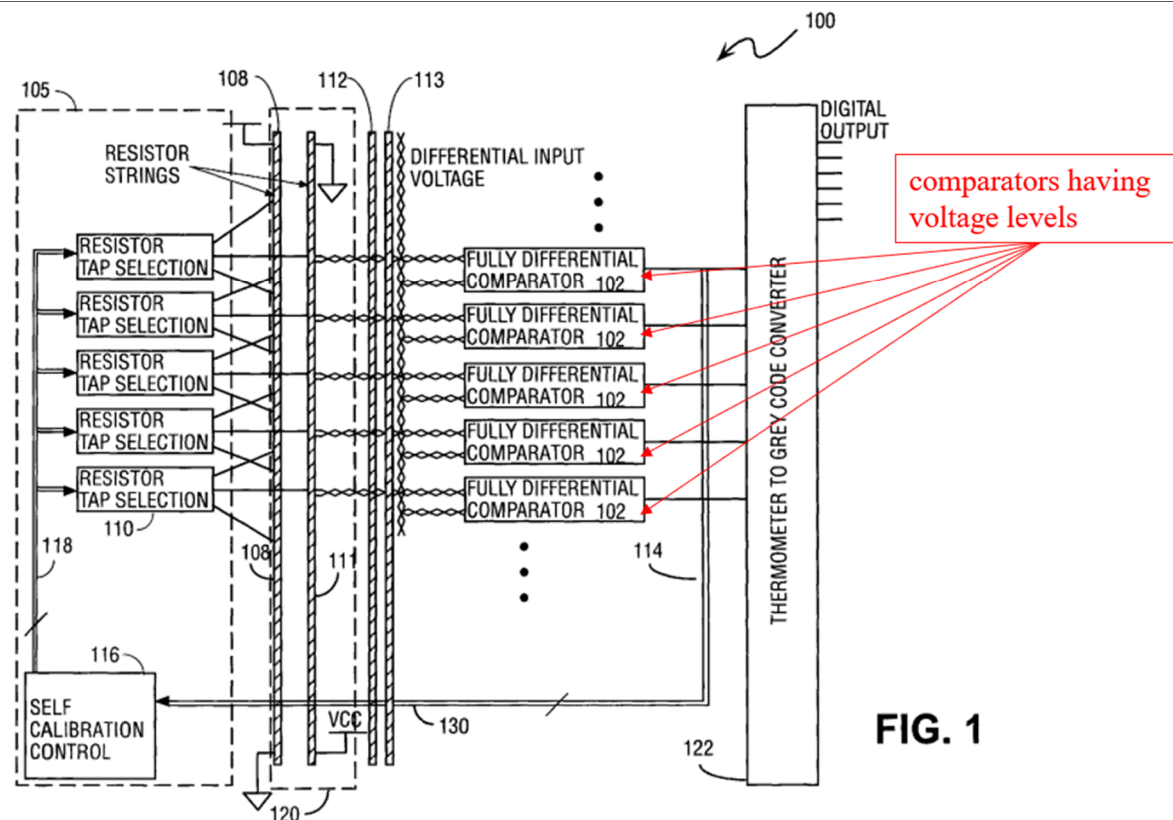
	Allee discloses details of a comparator that is self-calibrating and allows expansion of the functionality of digital integrated circuits. Allee at 2:35-36, 3:9-19. Allee teaches computer systems including comparators that may be calibrated by a self-calibration circuit 116 to adjust the input voltage. <i>Id.</i> at 3:9-19, 4:1-9, 6:17-39, Fig. 1. As depicted in Fig. 1 below, Allee teaches comparators 102 that may be self-calibrated to compensate for an input offset by adjusting an input signal equal to a desired reference voltage. <i>Id.</i> at 3:9-19, Fig. 1. Allee further teaches a self-calibration circuit 116 that calibrates the comparators 102 to adjust the voltage by adjusting the tap position of the reference voltage. <i>Id.</i> at 4:1-9. Allee discloses both an initial calibration process (before operation), and self-calibration of the comparator during operation. <i>Id.</i> at 6:17-39.
--	---

GROUND 5B: Meaney, Nguyen, and Allee in view of the knowledge of a POSITA.

A. Motivation to Combine

In addition to being motivated to combine Meaney with Nguyen as explained in the motivation to combine for Ground 2, a POSITA would have been motivated to further combine Meaney and Nguyen with Allee with a reasonable expectation of success for the same reasons stated above in the motivation to combine for Ground 5A.

Claim 3	
3: The method of claim 1, wherein the parameter is a voltage level for a comparator.	<u>Discussion:</u> As I discussed above in Ground 5A at claim 3, Allee discloses wherein the parameter is a voltage level for a comparator. <u>Other Relevant Citations</u> Allee discloses details of a comparator that is self-calibrating and allows expansion of the functionality of digital integrated circuits. Allee at 2:35-36, 3:9-19. Allee teaches computer systems including comparators that may be calibrated by a self-calibration circuit 116 to adjust the input voltage. <i>Id.</i> at 3:9-19,4:1-9, 6:17-39, Fig. 1. As depicted in Fig. 1 below, Allee teaches comparators 102 that may be self-calibrated to compensate for an input offset by adjusting an input signal equal to a desired reference voltage. <i>Id.</i> at 3:9-19, Fig. 1 (annotated below).



Allee further teaches a self-calibration circuit 116 that calibrates the comparators 102 to adjust the voltage by adjusting the tap position of the reference voltage. *Id.* at 4:1-9. Allee discloses both an initial calibration process (before operation), and self-calibration of the comparator during operation. *Id.* at 6:17-39.