

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

HP INC., DELL TECHNOLOGIES INC., DELL INC.,
LENOVO (UNITED STATES) INC.,
Petitioners,

v.

UNIVERSAL CONNECTIVITY TECHNOLOGIES, INC.,
Patent Owner.

IPR2024-01429
U.S. Patent No. 7,187,307

**DECLARATION OF ANDREW WOLFE, PH.D
UNDER 37 C.F.R. § 1.68 IN SUPPORT OF PETITION FOR
INTER PARTES REVIEW**

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	QUALIFICATIONS AND PROFESSIONAL EXPERIENCE	4
III.	LEVEL OF ORDINARY SKILL IN THE ART	9
IV.	RELEVANT LEGAL STANDARDS	11
V.	BACKGROUND	15
VI.	OVERVIEW OF THE '307 PATENT	16
VII.	CLAIM CONSTRUCTION	19
VIII.	IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE	20
A.	Grounds 1 and 2: Claims 1, 2, 6, 7, and 53 are Anticipated by Shin and Claims 1, 2, 6, 7, and 53 are Obvious over Shin.....	21
1.	Summary of Shin	21
2.	Claim 1	26
3.	Claim 2	53
4.	Claim 6	60
5.	Claim 7	61
6.	Claim 53	67
B.	Ground 3: Claims 1, 2, 6, 7, 23, 25, 31, 53, 68, 70, 71, 72, 74, and 75 are Obvious over Yusairi in view of Shin.....	68
1.	Summary of Yusairi	68
2.	Summary of Shin	70
3.	Claim 1	70
4.	Claim 2	88
5.	Claim 6	90
6.	Claim 7	91
7.	Claim 23	91
8.	Claim 25	95
9.	Claim 31	96
10.	Claim 53	100

11.	Claim 68	101
12.	Claim 70	114
13.	Claim 71	115
14.	Claim 72	116
15.	Claim 74	117
16.	Claim 75	117
C.	Secondary Considerations of Non-obviousness.....	119
IX.	CONCLUSION.....	119

I, Dr. Andrew Wolfe, do hereby declare as follows:

I. INTRODUCTION

1. I am making this declaration at the request of Petitioners HP Inc., Dell Technologies Inc., Dell Inc., and Lenovo (United States) Inc., (collectively, “Petitioners”) in the matter of this *Inter Partes* Review of U.S. Patent No. 7,187,307 (the “’307 Patent”).

2. I am being compensated for my work on this matter and for reasonable and customary expenses associated with my work and testimony in this proceeding. My compensation is not contingent on the outcome of this matter or the specifics of my testimony, and I have no other interest in this case or the parties thereto.

3. I have been asked to provide my opinions regarding whether claims 1, 2, 6, 7, 23, 25, 31, 53, 68, 70, 71, 72, 74, and 75 (the “Challenged Claims”) of the ’307 Patent are unpatentable insofar as they would have been anticipated and/or obvious to a person having ordinary skill in the art (“POSITA”) at the time of the alleged invention, in light of the prior art.

4. It is my opinion that every element of the Challenged Claims is found in the prior art references.

5. It is also my opinion that all of the Challenged Claims are unpatentable in that they would have been anticipated and/or obvious to a POSITA.

6. In the preparation of this declaration, I have reviewed the below as well as any other documents cited herein not listed below:

Ex. 1001	U.S. Patent No. 7,187,307 (“’307 Patent”)
Ex. 1002	Prosecution History of U.S. Patent No. 7,187,307 (U.S. Patent App. No. 10/459,989)
Ex. 1003	Declaration of Dr. Andrew Wolfe
Ex. 1004	U.S. Patent Pub. No. 2002/0159450 to Shin <i>et al.</i> (“Shin”)
Ex. 1005	“Hardware Design and Implementation of IP-over-1394 Protocol Stack and Its Evaluation,” Mohd Yusairi, Bin Abu Hassan, and Koki Abe, ITE Technical Report Vol.27, No.18, PP 51-85 (March 2003) (“Yusairi”).
Ex. 1006	Declaration of Nathaniel E. Frank-White
Ex. 1007	Declaration of Yukio Iwahana
Ex. 1008	“IEEE Standard for a High Performance Serial Bus,” in IEEE Std 1394-1995, pp.1-384 (1996) (“IEEE1394”)
Ex. 1009	“Serial ATA: High Speed Serialized AT Attachment,” Revision 1.0a, January 7, 2003. (“SATA 1.0a”)
Ex. 1010	“High-Definition Multimedia Interface Specification Version 1.0,” December 9, 2002 (“HDMI 1.0”)
Ex. 1011	“Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications,” IEEE Standard 802.3, 1998 Edition (“IEEE 802.3”)
Ex. 1012	Declaration of Dr. Knut Grimsrud

Ex. 1013	Declaration of Steve Venuti
Ex. 1014	U.S. Pat. No. 6,324,602 to Chen et al. (“Chen”)
Ex. 1015	CV of Andrew Wolfe, Ph.D.
Ex. 1016	English translation, certification, and Japanese original of Apostille from National Diet Library (Tokyo, Japan) regarding Yusairi
Ex. 1017	English translation, certification, and Japanese original of Yusairi from National Diet Library (Tokyo, Japan)
Ex. 1018	“Line Coding,” from EEWeb, https://www.eeweb.com/line-coding/#:~:text=Line%20coding%20is%20the%20process%20of
Ex. 1019	“Line code,” from Wikipedia, https://en.wikipedia.org/wiki/Line_code
Ex. 1020	“8b/10b encoding,” from Wikipedia, https://en.wikipedia.org/wiki/8b/10b_encoding
Ex. 1021	“Duplex (telecommunications),” from Wikipedia, https://en.wikipedia.org/wiki/Duplex_(telecommunications)
Ex. 1022	“Plesiochronous system,” from Wikipedia, https://en.wikipedia.org/wiki/Plesiochronous_system
Ex.1023	“Encapsulation (networking),” from Wikipedia, https://en.wikipedia.org/wiki/Encapsulation_(networking)
Ex. 1024	“Non-return-to-zero,” from Wikipedia, https://en.wikipedia.org/wiki/Non-return-to-zero#:~:text=In%20telecommunications%2C%20a%20non%2Dr return,other%20neutral%20or%20rest%20condition.

7. In forming the opinions expressed below, I have considered: the documents listed above, the relevant legal standards, including the standards for anticipation and obviousness; any additional authoritative documents as cited in the body of this declaration; and my own knowledge and experience based upon my work in in the computer and electronics fields.

8. I may rely upon these materials, my knowledge and experience, and/or additional materials to rebut arguments raised by the patent owner. Further, I may also consider additional documents and information in forming any necessary opinions, including documents that may not yet have been provided to me.

9. Unless otherwise noted, all emphasis in any quoted material has been added.

II. QUALIFICATIONS AND PROFESSIONAL EXPERIENCE

10. My complete qualifications and professional experience are described in my Curriculum Vitae, a current copy of which can be found in Exhibit 1015. The following is a brief summary of my relevant qualifications and professional experience.

11. In 1985, I earned a Bachelor's degree in Electrical Engineering and Computer Science from The Johns Hopkins University. In 1987, I received an Master's degree in Electrical and Computer Engineering from Carnegie Mellon University. In 1992, I received a Ph.D. in Computer Engineering from Carnegie

Mellon University. My doctoral dissertation proposed a new approach for the architecture of a computer processor.

12. I have more than 35 years of experience as a computer architect, computer system designer, personal computer graphics designer, educator, and executive in the electronics industry.

13. In 1983, I began designing touch sensors, microprocessor-based computer systems, and I/O (input/output) cards for personal computers as a senior design engineer for Touch Technology, Inc. During the course of my design projects with Touch Technology, I designed I/O cards for PC-compatible computer systems, including the IBM PC-AT, to interface with interactive touch-based computer terminals that I designed for use in public information systems.

14. I have extensive experience in the design, development, and analysis of portable computing devices including portable PCs. In the mid-late 1980s I worked as part of the development team for the Linus Write-Top, which is commonly acknowledged to be the first commercial tablet computer. I later worked on screen technology for many handheld computing devices including the Motorola Envoy and prototypes for the Apple Newton.

15. From 1986 through 1987, I designed and built a high-performance computer system as a student at Carnegie Mellon University. From 1986 through

early 1988, I also developed the curriculum and supervised the teaching laboratory for processor design courses.

16. At the end of 1989, I (along with some partners) reacquired the rights to the technology I had developed at Touch Technology and at AMP and founded The Graphics Technology Company. Over the next seven years, as an officer and a consultant for The Graphics Technology Company, I managed the company's engineering development activities and personally developed dozens of touch screen sensors, controllers, and interactive touch-based computer systems.

17. I have also served as a technology advisor to Motorola and to several venture capital funds in the U.S. and Europe. Currently, I am a director of Turtle Beach Corporation, providing guidance in its development of premium audio peripheral devices for a variety of commercial electronic products.

18. From 1991 through 1997, I served on the Faculty of Princeton University as an Assistant Professor of Electrical Engineering. At Princeton, I taught undergraduate and graduate-level courses in Computer Architecture, Advanced Computer Architecture, Display Technology, and Microprocessor Systems, and conducted sponsored research in the areas of embedded computing systems, multimedia, video signal processors, compiler optimization, and high performance computer architecture. I was also a principal investigator for DOD research in video

technology and a principal investigator for the New Jersey Center for Multimedia Research.

19. At Princeton, I received several teaching awards, both from students and from the School of Engineering. I have also taught advanced microprocessor architecture to industry professionals in seminars sponsored by the Institute of Electrical and Electronics Engineers (“IEEE”) and the Association for Computing Machinery (“ACM”).

20. In 1997, I was hired to run the research efforts at S3, Inc. (which later became SONIC|blue, Inc.) and served as Chief Technology Officer there from 1999 until 2002. At the time, S3 produced the 3D graphics hardware and driver software for more than half of new PCs, worldwide. I managed our display technology group. I led the development of components for laptop computers including the IBM Thinkpad and for desktop computers as well. I also worked with the teams that developed the FrontPath wireless tablet PC and the Diamond Mako handheld computer. My work at S3/Sonic included extensive work with and development of numerous serial data communications protocols, including HDMI, DVI, LVDS, USB, RS-232, 1394, 802.3, 802.11, RAMBUS, and SD cards, among others. My experience with serial data communications protocols throughout my career has also included SATA, PCI-E, and DisplayPort. This work has included the subjects of encoding, transmitting, and storing audio and video data, as well as control data, for

transmission over a serial link and packet encapsulation and decapsulation, encryption, and decryption, among many other similar subjects.

21. From 1999 through 2002, while a Consulting Professor at Stanford University, I taught computer architecture and microprocessor design to both undergraduate and graduate students.

22. Since 2002, I have operated Wolfe Consulting to provide services as a consultant on processor technology, computer systems, consumer electronics, software, design tools, and intellectual property issues. My consulting work relating to intellectual property has included serving as a testifying and consulting expert for IP and other technology-related litigation matters.

23. From 2013 to the present, I have been serving as a lecturer at Santa Clara University teaching courses on Microprocessor Systems, Integrated Circuit Design, Real-Time Computing, and Mechatronics and recently served as Faculty Senate President.

24. I have published more than fifty peer-reviewed papers in computer architecture and computer systems and IC design. I also have chaired IEEE and ACM conferences in microarchitecture and integrated circuit design and served as an associate editor for IEEE and ACM journals. I served on the IEEE Computer Society Awards committee. I am an IEEE Fellow, an IEEE Computer Society Distinguished Contributor and a Member of ACM. I am a named inventor on at least

fifty-seven U.S. patents and thirty-seven foreign patents, which are listed in my curriculum vitae.

25. In 2002, I was the invited keynote speaker at the ACM/IEEE International Symposium on Microarchitecture and at the International Conference on Multimedia. From 1990 through 2005, I was an invited speaker on various aspects of technology and the PC industry at numerous industry events including the Intel Developer's Forum, Microsoft Windows Hardware Engineering Conference, Microprocessor Forum, Embedded Systems Conference, Comdex, and Consumer Electronics Show, as well as at the Harvard Business School and the University of Illinois Law School. I have been interviewed on subjects related to computer graphics and video technology and the electronics industry by publications such as the Wall Street Journal, New York Times, Los Angeles Times, Time, Newsweek, Forbes, and Fortune as well as on CNN, NPR, and the BBC. I have also spoken at dozens of universities including MIT, Stanford, University of Texas, Carnegie Mellon University, UCLA, University of Michigan, Rice University, and Duke University.

III. LEVEL OF ORDINARY SKILL IN THE ART

26. I understand that my assessment of the claims of the '307 Patent and the teachings of the prior art and my analysis and opinions herein must be undertaken from the perspective of what would have been known or understood by a person

having ordinary skill in the art, reading the '307 Patent on its priority date and in light of the specification and file history of the '307 Patent. I refer to such a person as a "POSITA."

27. I further understand there are multiple factors relevant to determining the level of ordinary skill in the pertinent art, including: (1) the levels of education and experience of persons working in the field at the time of the invention; (2) the sophistication of the technology; (3) the types of problems encountered in the field; and (4) the prior art solutions to those problems.

28. I have been instructed to assume that the priority date for the '307 Patent, which does not claim priority to any earlier application, is its June 12, 2003, filing date.

29. Based on my study of the '307 Patent and my experience working in the field during the relevant timeframe, including my experience working with other skilled artisans in the field and educating undergraduate and graduate students in the field at the time, a POSITA in the field of the '307 Patent, as of June 12, 2003, would have had a bachelor's degree in electrical engineering, computer engineering, computer science, or equivalent training, and three years of experience with data communications systems or interfaces, including at least some experience with serial or display data communications systems or interfaces. Additional education can substitute for less work experience, and vice versa.

30. I note that I met at least these minimum qualifications to be considered a POSITA as of the priority date of the '307 Patent of June 12, 2003.

31. I understand that a POSITA is not a specific real individual, but rather a hypothetical individual having the qualities reflected by the factors above. This hypothetical person has knowledge of all prior art in the relevant field and takes from each reference what it would teach to a person having the skills of a POSITA.

32. For purposes of this Declaration, in general, and unless otherwise noted, my statement and opinions, such as those regarding my experience and the understanding of a POSITA generally (and specifically related to the references I consulted herein), reflect the knowledge that existed in the field as of the alleged priority date of the '307 Patent (*i.e.*, June 12, 2003). Unless otherwise stated, when I provide my understanding and analysis below, it is consistent with the skill level of a POSITA as of the alleged priority date of the '307 Patent.

IV. RELEVANT LEGAL STANDARDS

33. I am not an attorney. In preparing and expressing my opinions and considering the subject matter of the '307 Patent, I am relying on certain basic legal principles that counsel have explained to me. These principles are discussed below.

34. I understand that prior art to the '307 Patent includes patents and printed publications in the relevant art that predate the priority date of the alleged invention

recited in the '307 Patent. As noted, here, I have been instructed to assume June 12, 2003 is the priority date of the '307 Patent.

35. I have been informed and understand that a patent claim is unpatentable as “anticipated” under 35 U.S.C. § 102 if each element of that claim is disclosed either explicitly or inherently in a single prior art reference. I understand that the question of anticipation is whether a POSITA would have reasonably understood or inferred from the prior art reference’s teachings that every claim element was disclosed in that single reference. I further understand that a disclosure is “inherent” if the missing element is necessarily present in view of the explicit disclosure. The fact that the reference might possibly practice or contain a claimed limitation is insufficient to establish that the reference inherently discloses the limitation. For anticipation by a prior art patent or printed publication, I further understand that the reference’s description must enable a POSITA to practice the claimed invention without undue experimentation.

36. I have been informed that a patent claim is unpatentable as “obvious” under 35 U.S.C. § 103 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 having ordinary skill in the art to which the subject matter pertains. I have also been informed by counsel that the obviousness analysis takes into account factual inquiries including the level of ordinary skill in

the art, the scope and content of the prior art, and the differences between the prior art and the claimed subject matter.

37. I understand that for a claim to be unpatentable as obvious, the party asserting unpatentability must identify prior art references that alone or in combination with one or more other prior art references would have rendered the claim obvious to one of ordinary skill in the art at the time of the invention.

38. I understand that a single prior art reference can render a claim obvious if it would have been obvious to modify that reference to arrive at the patented invention. I understand that it is appropriate to look to the knowledge of a POSITA in assessing whether it would have been obvious to modify a single prior art reference to produce the claimed invention.

39. I understand that a conclusion of obviousness may be based on a combination of prior art references, particularly if the combination of elements does no more than yield predictable results. I understand that it may be helpful to identify a reason that would have prompted a POSITA in the relevant field to combine the elements in a way the claimed invention does. I understand that an invention would have been obvious where a POSITA pursues known options from a finite number of identified, predictable solutions. I understand that if trying a limited number of solutions leads to expected success, it is likely the product is not one of innovation but of ordinary skill and common sense.

40. I understand that in order for a reference to be proper for use in an obviousness combination, the reference must be analogous art to the claimed invention. I understand a reference is analogous art to the claimed invention if: (1) the reference is from the same field of endeavor as the claimed invention (even if it addresses a different problem); or (2) the 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). I understand that the “same field of endeavor” and “reasonably pertinent” tests are two separate tests for establishing analogous art and it is not necessary for a reference to fulfill both tests in order to qualify as analogous art.

41. I have been informed by counsel that the Supreme Court has recognized several rationales for combining references or modifying a reference to show obviousness of claimed subject matter. Some of these rationales include the following: (a) combining prior art elements according to known methods to yield predictable results; (b) simple substitution of one known element for another to obtain predictable results; (c) use of a known technique to improve a similar device (method, or product) in the same way; (d) applying a known technique to a known device (method, or product) ready for improvement to yield predictable results; (e) choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success; and (f) some teaching, suggestion, or motivation in the prior

art that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention.

42. I understand that in evaluating whether patent claims are obvious, secondary considerations of nonobviousness are considered. I understand that the secondary considerations of nonobviousness include commercial success, long-felt but unsolved needs, copying, praise, teaching away from the claimed invention by the prior art, unexpected results, industry acceptance, failure of others, and skepticism by experts. I understand that the secondary considerations are considered with the balance of obviousness evidence in the record to act as a check against impermissible hindsight bias. I further understand that the burden of presenting evidence of secondary considerations is on the patent owner, and the patent owner must establish a nexus between the merits of the claimed invention and the evidence of secondary considerations.

V. BACKGROUND

43. The '307 Patent describes a communication system that uses block encoding to packetize and transmit data between devices. '307 Patent at 8:38-48. The devices within its system operate using layers of the OSI model, such as a physical layer, link layer, and higher-level layers.'307 Patent at 1:24-34; 4:56-5:14. A receiver receives the block-encoded data and decodes it at the physical and link layers to recover application data and control bits. '307 Patent at 4:56-63. The

control bits have multiple levels of communication protocol functionality. '307 Patent at 4:66-5:14.

44. However, the multifunctional control bits of the purported invention—which the Applicant argued during prosecution was a main point of novelty—were not new. Prosecution History of the '307 Patent (Ex.2002) at 183-184. Multiple prior art references confirm this, including Shin and Yusairi. Shin describes bits recovered from a synchronization primitive and a preempt primitive that cause more than one layer of a receiver to execute certain functions. Shin at [0096], [0111]-[0115]. Likewise, Yusairi describes a data packet with a link header, which includes bits of a transaction code and bits of a type field that cause functions at multiple levels. Yusairi at 54-55, Sec. 5.2.

45. As explained in further detail below, the purported novel features of the '307 Patent, including the multifunctional control bits, were not novel. Each of the limitations in the Challenged Claims is found in prior art references before the earliest priority date of the '307 Patent.

VI. OVERVIEW OF THE '307 PATENT

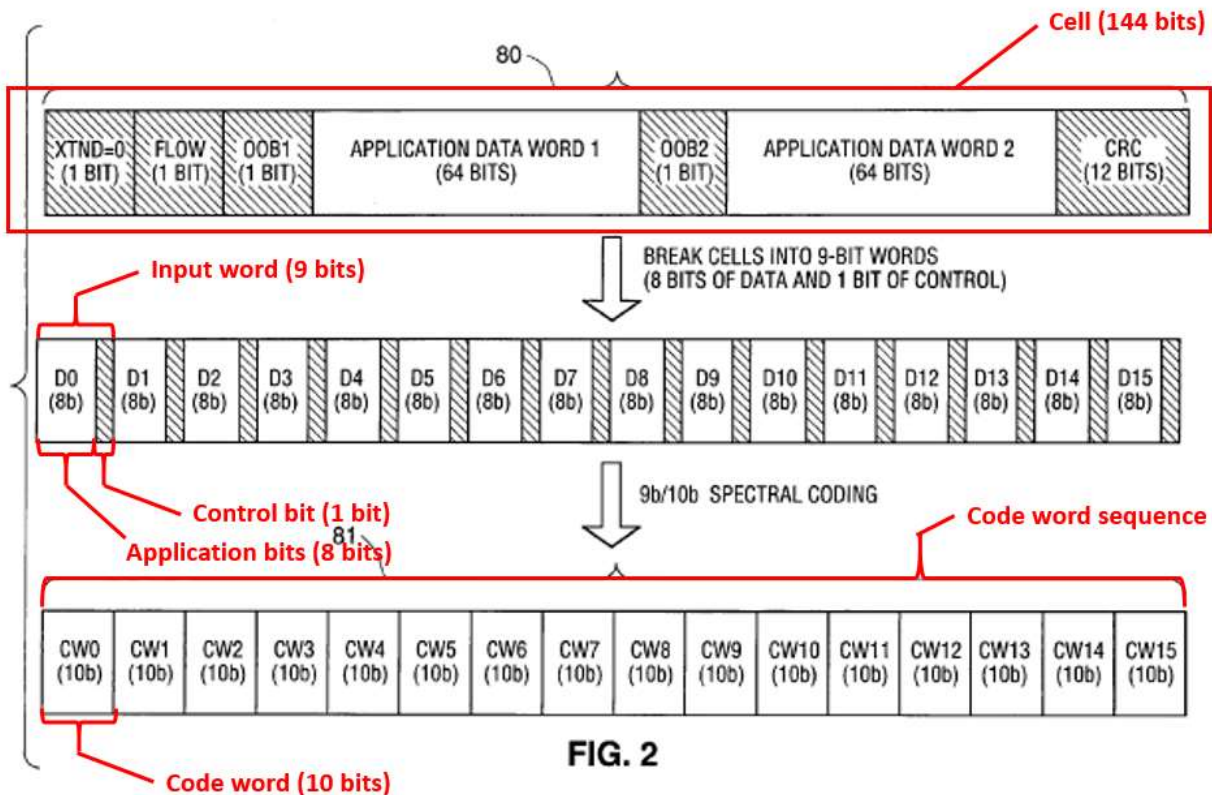
46. The '307 Patent relates to a communication system with at least one transmitter and one receiver. The transmitter is configured to generate encoded data in accordance with a line code for transmission over a serial link. '307 Patent (Ex.1001), Abstract. The line code specifies a block code for encoding application

data and control bits, and the block code can encode user data bits as a larger number of bits. '307 Patent, at Abstract. The receiver is configured to decode the encoded data to recover the application data and control bits. '307 Patent at Abstract; '307 Patent at 3:6-52, 8:39-42. The application data can include data such as video or audio data. '307 Patent at 6:36-38. Control data causes an endpoint, e.g., the transmitter or receiver, of the communication system to perform a function. '307 Patent at 4:63-66, 6:39-43.

47. The transmitter and receiver operate according to a stacked communication protocol architecture, such as, for example, the OSI reference model. '307 Patent at 1:24-43, 10:20-28, 12:42-50. The OSI model has seven layers, including the (1) physical layer; (2) link layer; (3) network layer; (4) transport layer; (5) session layer; (6) presentation layer; and (7) application layer. '307 Patent 1:24-34. With respect to the '307 Patent, the physical and link layers are responsible for transmitting and receiving the encoded line code data. '307 Patent at 8:24-25. Any layer above the link layer is referred to as “higher level circuitry,” and the control bits have multiple levels of communication protocol functionality, which allows them to be used at the link layer and a higher layer. '307 Patent at 4:63-66, 5:1-14, 6:39-43, 54:62-67.

48. An exemplary “cell,” along with “input words” and encoded “code words,” is shown below in Figure 2. '307 Patent at 4:38, 13:57-61. To encode the

data, a cell of 144 bits can be grouped into 16, nine-bit input words, each having eight application bits and one control bit. '307 Patent at 14:26-31. The transmitter then applies "block coding" to encode each "input word" as a "code word." '307 Patent at 14:48-52. In the present example, the nine-bit input words are encoded into ten-bit code words. '307 Patent at 14:48-52. The code words form a "code word sequence," which the patent describes as indicative of at least one cell of the input words. '307 Patent at 14:48-56.



49. The block encoding scheme of the '307 Patent encodes M-bit input words into N-bit code words where $N > M$. '307 Patent at 9:19-36; 4:53-59. One example of the scheme is the conventional 8B/10B block code where 8B/10B block

encoding results in $M=8$ and $N=10$ such that $N>M$. '307 Patent at 3:62-64, 56:10-14. Another example is the 9B/10B block code. '307 Patent at 11:45-47. After the transmitter encodes and transmits the line code, the receiver decodes the code words into application data and control bits at its physical layer. '307 Patent at 4:56-66. For instance, referring back to Figure 2, the 10-bit code words are decoded into nine-bit words each having eight application bits and one control bit. '307 Patent 11:16-20.

50. The '307 Patent also suggests that the transmitter can transmit special characters. Special characters “are typically transmitted between cells” and are “distinguishable from bit sequences of encoded cells.” '307 Patent at 5:17-20, 8:45-48. That is, “no special character matches any of the code words.” '307 Patent at 5:22-23, 10:55-60. The patent describes special characters as “idle characters,” “link training characters,” or “flow control characters.” '307 Patent at 9:19-22, 9:33-37, 51:2-4. Idle characters can be inserted between code word sequences for clock synchronization. '307 Patent at 49:17-22. Link training characters can be inserted after a link error or startup to establish symbol framing. '307 Patent at 17:1-8. Flow control characters can be used to help ensure transmitters and receivers will not overflow their buffers. '307 Patent 53:34-43.

VII. CLAIM CONSTRUCTION

51. It is my understanding that in order to properly evaluate the '307 Patent, the terms of the claims must first be construed. It is my understanding that for the

purposes of this *inter partes* review, the claims are to be construed under the so-called *Phillips* standard, under which claim terms are given their ordinary and customary meaning as would be understood by one of ordinary skill in the art in light of the specification and prosecution history, unless the inventor has set forth a special meaning for a term. For purposes of my analysis below, I do not believe any claim terms require explicit construction.

VIII. IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE

52. I have been asked to provide my opinions as to whether the Challenged Claims of the '307 Patent would have been anticipated and/or obvious in view of the prior art. The discussion below provides a detailed analysis of how the prior art references identified below teach the limitations of the Challenged Claims of the '307 Patent.

53. As part of my analysis, I have considered the scope and content of the prior art and any differences between the alleged invention and the prior art. For each challenged claim of the '307 Patent, I provide an element-by-element analysis of the scope and content of the prior art and any differences between the alleged invention and the prior art.

54. As described in detail below, the alleged invention of the Challenged Claims would have been anticipated and/or obvious in view of the teachings of the

identified prior art references as well as the knowledge of a POSITA as of the time of the assumed priority date of the '307 Patent.

A. Grounds 1 and 2: Claims 1, 2, 6, 7, and 53 are Anticipated by Shin and Claims 1, 2, 6, 7, and 53 are Obvious over Shin

1. Summary of Shin

55. U.S. Patent Application Publication No. 2002/0159450 to Shin et al. (“**Shin**,” Ex.1004), Shin discloses a serial data communications architecture for communication between hosts and data store devices. Shin at Abstract, [0070]. Like the '307 Patent, the Shin architecture is a stacked communication protocol architecture, including “transport, link, and physical layers.” Shin at [0071]. “[E]ach layer in the hierarchy is aware of and can take advantage of characteristics of lower layers in the hierarchy.” Shin at [0071]. For example, the transport layer “may receive signals from the link layer that indicate when an end of packet is received.” Shin at [0071]. The link layer can also process control information for functions performed at the link layer. Shin at [0096] (discussing synchronization primitives).

56. The Shin architecture is depicted in Figure 1 below. Shin includes hosts 110 and data store devices 120 interconnected by a switching network 130 via serial communications links 140. Shin at [0075], Fig. 1. Each of the hosts 110, data store devices 120, and switching network 130 includes one or more communication

interfaces 115, 125, and 135, each including a transport layer, link layer, and physical layer.

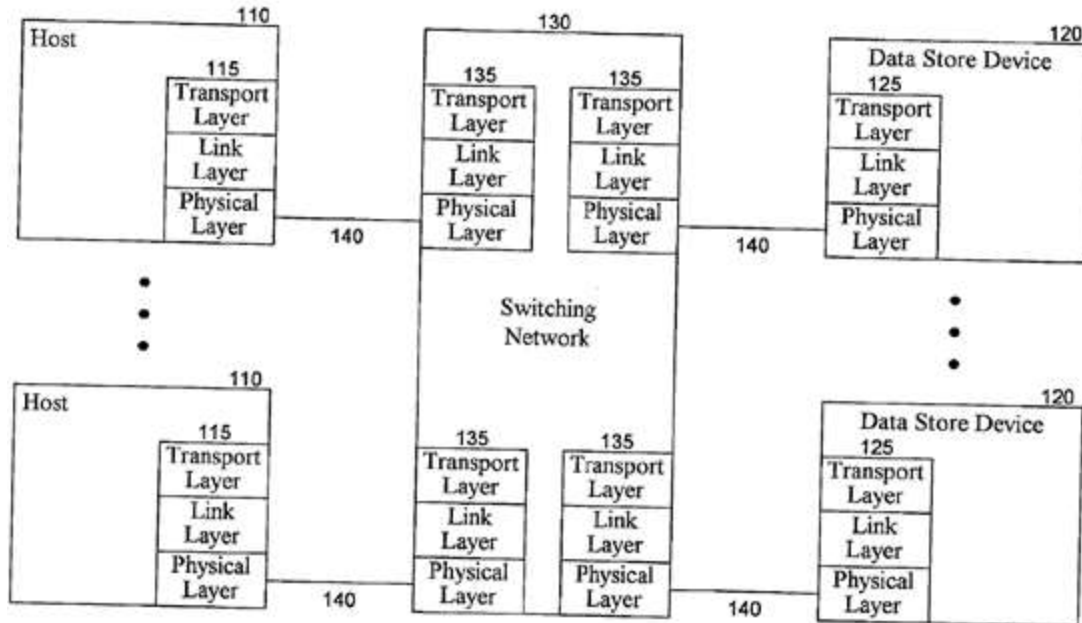
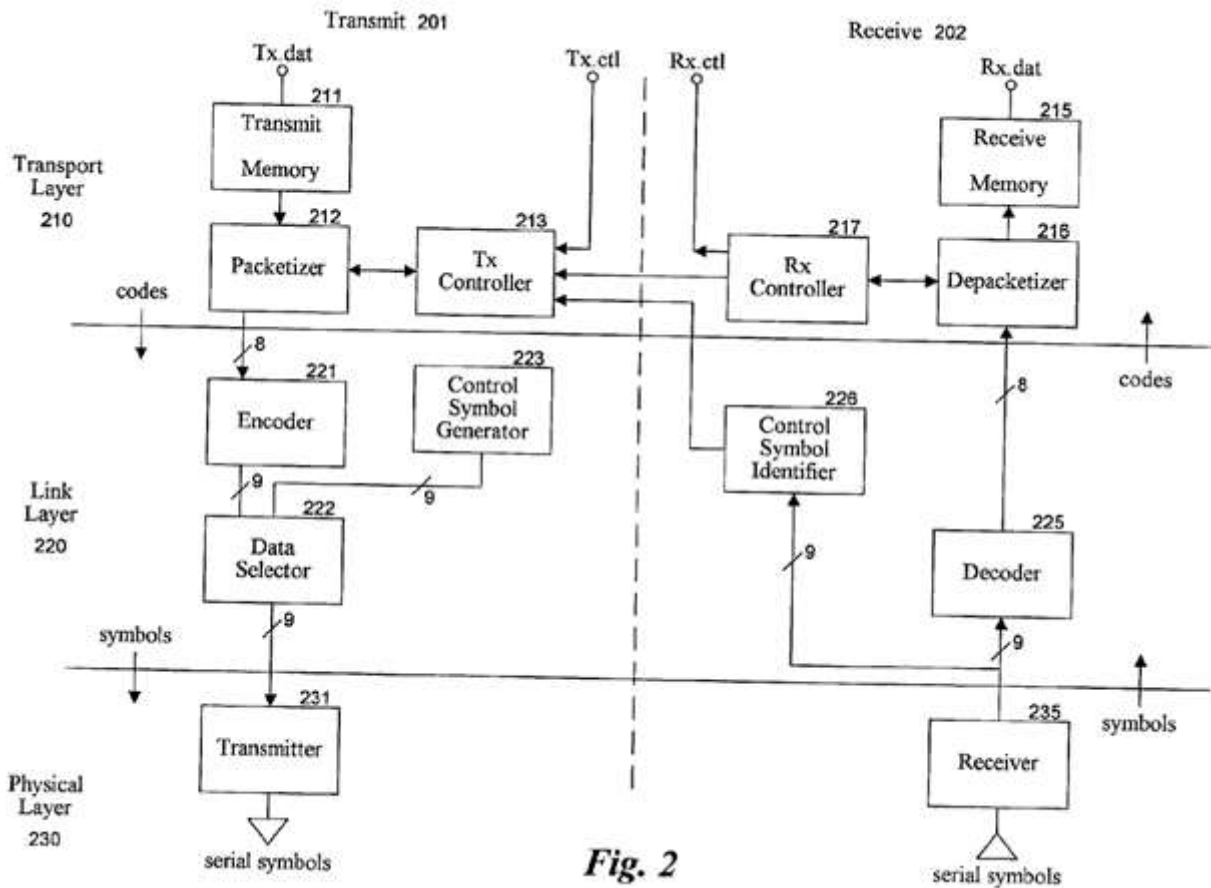


Fig. 1

Ex.1004 at Fig. 1. The host 110 and data store device 120 also include an “application upper layer.” Shin at [0076].

57. The transmit components of the link layer include an encoder 221, a data selector 222, and a control symbol generator 223, and the receive components of the link layer include a decoder 225 and a control symbol identifier 226, as shown below in Figure 2 of Shin. Shin at [0079].

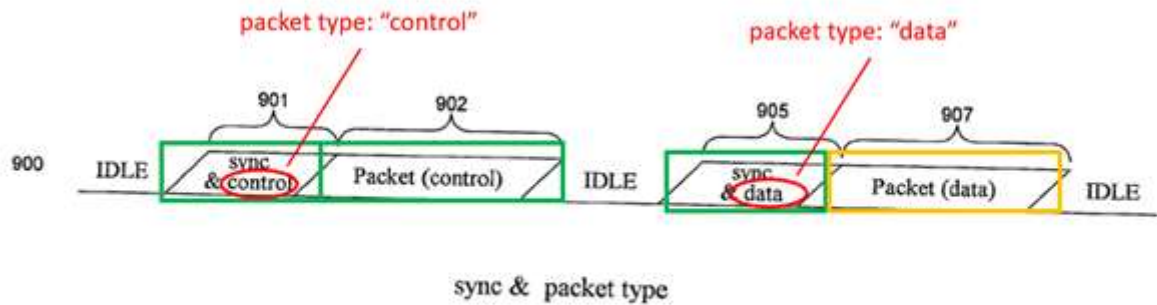


Shin at Fig. 2. The data selector 222 “inserts control symbols and synchronization symbols into the sequence of symbols that are to be transmitted.” Shin at [0079]. On the receive side 202, the decoder 225 of the link layer decodes symbols into codes, and the control symbol identifier 226 identifies control symbols and signals the transport layer as appropriate. *Id.*

58. For example, Shin describes synchronization primitives that the transmitter periodically transmits for proper alignment of symbols at the receiver.¹ Shin at [0096]. Packet type may be encoded in a synchronization primitive, which allows the link layer to transmit a synchronization primitive specifying the packet type before sending the packet itself. Shin at [0096]. For example, as shown in Figure 9A below, a synchronization primitive 901 encoded with a packet type of “control” is transmitted immediately prior to a control packet 902, and a synchronization primitive 905 encoded with a packet type of “data” is transmitted immediately prior to a data packet 907.²

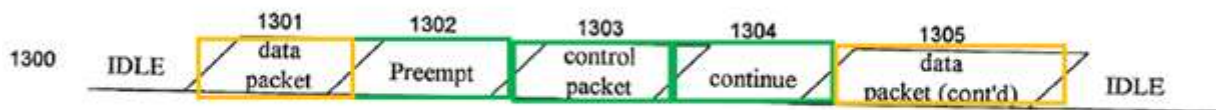
¹ Shin describes that “primitives are pairs of out-of-band symbols” and states that “a symbol is assumed to have nine bits.” Ex.1004 at [0098], [0138]-[0139].

² Petitioner notes that Shin’s specification refers to a “synchronization primitive 903” encoded with a packet type of “data” and “a data packet 904,” but in light of the specification as a whole, a POSITA would have understood that these refer to reference numerals 905 and 907, respectively.

**Fig. 9A**

Thus, the link layer of the receiving node can rapidly detect packet type based on decoding the synchronization primitive, rather than a packet header, which can reduce processing time and enhance overall processing speed. Shin at [0096].

59. In addition to synchronization primitives, Shin states that the link layer can detect a Preempt primitive to interrupt transmission of a packet, as shown in Figure 13 below. Shin at [0111]-[0115].

**Fig. 13**

60. Shin at Fig. 13 (annotated). The link layer of the receiving node detects the Preempt primitive and signals to the transport layer of the receiving node, causing the transport layer to “save current packet information” for the data packet “so that it can continue storing codes in the [data] packet after the preemption is

complete.” Shin at [0115]. The transport layer also “starts storing the subsequent codes as a control packet” (“the preempting packet”) until the link layer indicates it has received a continue primitive for resuming the data packet. Shin at [0115], [0112].

2. Claim 1

[1(pre)] A communication system, including:

61. To the extent the preamble is limiting, Shin discloses this limitation.

62. As provided below, Figure 1 illustrates a *communication system* that includes a storage link architecture having hosts and data store devices that communicate with each other. Shin at [0075], FIG. 1.

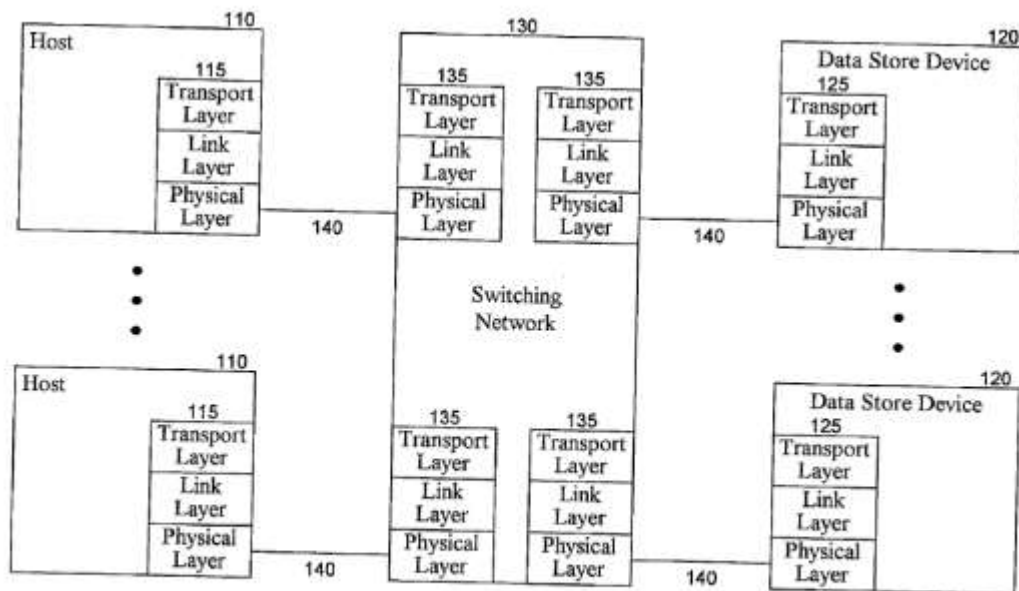


Fig. 1

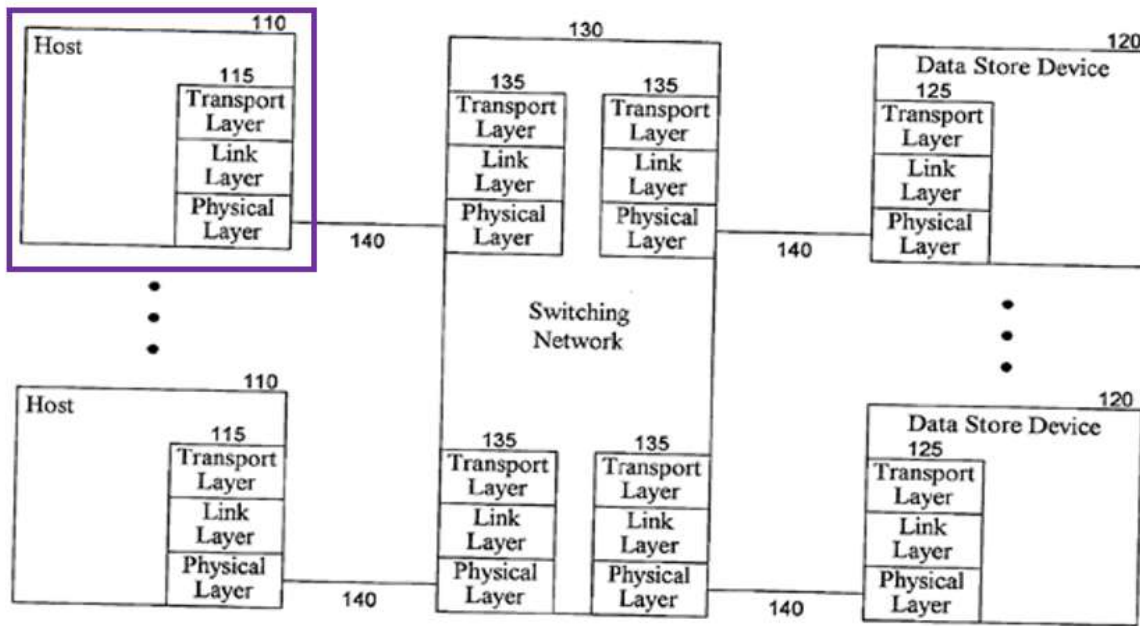
Shin states, “FIG. 1 is a block diagram illustrating components of the storage link architecture in one embodiment. **The hosts 110 and data store devices 120 are**

interconnected to switching network 130 via serial communications links 140.

The hosts may include computer systems that access the data store devices. The data store devices may include storage area network devices (e.g., disk drives), high-speed memory devices, and other devices for storing data.” Shin at [0075].

[1(a)] a first endpoint;

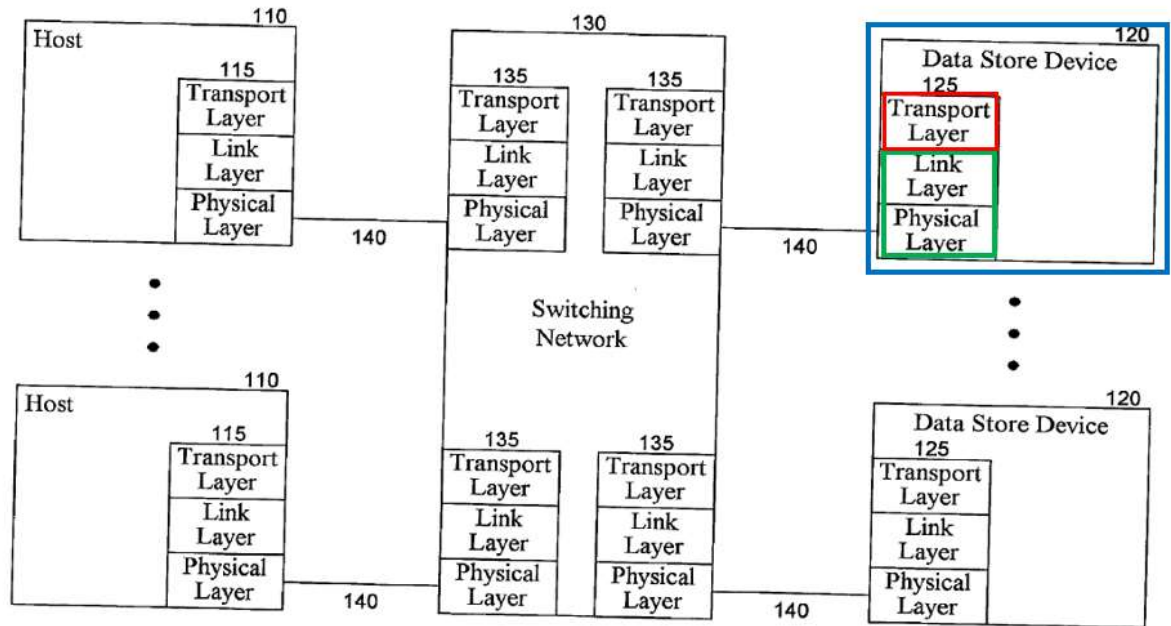
63. Shin discloses this limitation. Shin’s storage link architecture supports packet-based communications between hosts and storage devices, where one host acts as a transmitter (*a first endpoint*) to transmit data that is received at a data store device. Shin at [0071], [0076], [0077]. Shin states, “Each host, data store device, and switch is a communications node that includes one or more communications interfaces 115, 1257 and 135 with a transport layer, a link layer, and a physical layer. **The hosts and data store devices have an upper layer that communicate to each other** via transactions.” Shin at [0076]. Annotated Figure 1 below depicts a host, in **purple**, acting as a *first endpoint*.

**Fig. 1**

[1(b)] a second endpoint, including physical and link layer circuitry and higher level circuitry coupled to the physical and link layer circuitry; and

64. Shin discloses this limitation.

65. “Each host, data store device, and switch is a communications node that includes one or more communications interfaces 115, 125 and 135 with a transport layer, a link layer, and a physical layer,” respectively supporting higher-level circuitry, link layer circuitry, and physical layer circuitry. Shin at [0076], FIG. 1. A host communicates to a data store device 120 (*the second endpoint*), annotated below in blue. Shin at FIG. 1 (annotated). As illustrated, the data store device, in blue, acts as the second endpoint and includes the transport layer (*transport layer circuitry*), in red, and the link layer (*link layer circuitry*) and physical layer (*physical layer circuitry*), both in green. Shin at [0076], FIG. 1.

**Fig. 1**

66. The transport layer (*higher level circuitry*) is coupled to the physical and link layer circuitry so that “[t]he transport layer may indicate to the link layer when to transmit control symbols, and the link layer may indicate to the transport layer when control symbols are received.” Shin at [0079]. Further, the higher-level transport layer is coupled to the physical layer and the link layer so that the “transport layer [can] provide[] each code (e.g., each byte) of the packets to the link layer,” which “encodes the codes into symbols and, and provides those symbols to the physical layer.” Shin at [0076], [0077].

[1(c)] a serial link between the first endpoint and the second endpoint,

67. Shin discloses this limitation.

68. Shin provides that a host (*the first endpoint*) and a data store device (*the second endpoint*) “are interconnected to switching network 130 via serial communication links 140.” Shin at [0075]. Figure 1 below illustrates a serial communication link (*a serial link*), annotated in **blue**, between the host (*the first endpoint*) and the data store device (*the second endpoint*). Shin at [0075], FIG. 1.

69. As further noted by Shin, “a serial communications link,” such as the annotated link below, may be connected to “each port of [a] switch” to communication between a host and data store. Shin at [0076]. As such, the serial communications link 140 extends between and connects at least host 110 (*the first endpoint*) and data store device 120 (*the second endpoint*).

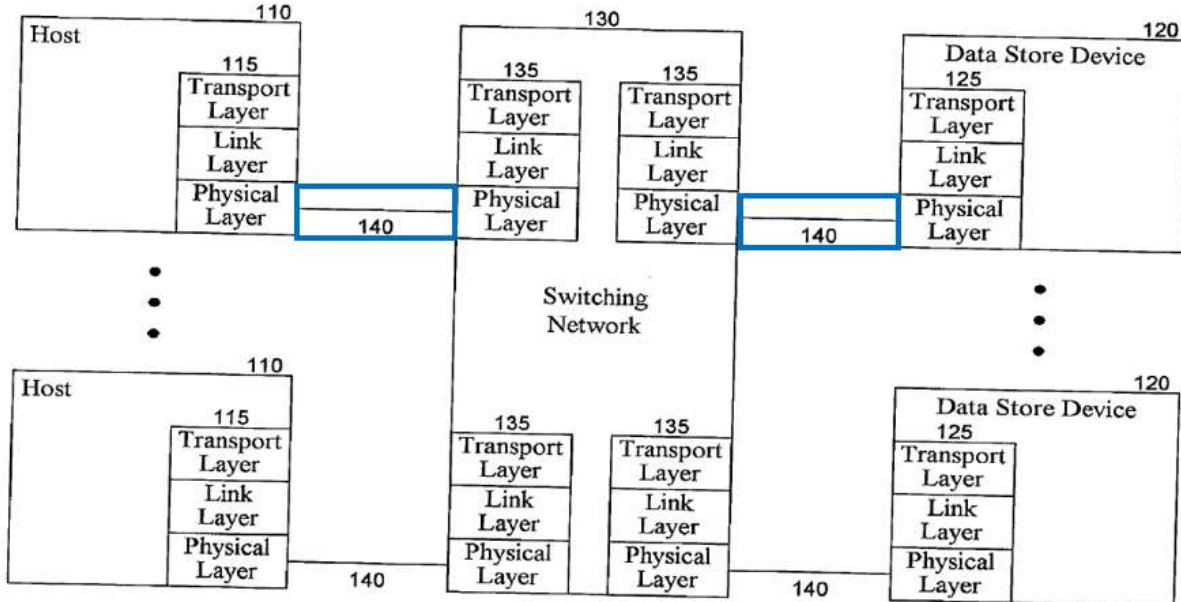


Fig. 1

[1(d)] wherein the first endpoint is configured to generate encoded data in accordance with a line code, and to transmit the encoded data over the link to the second endpoint, and

70. Shin discloses and renders obvious this limitation.

71. Shin teaches that “[e]ach host [*the first endpoint*], data store device [*the second endpoint*], and switch is a communications node.” Shin at [0076]. And Shin teaches that a communications node (a host for example) “encodes codes into symbols in order to optimize transition control and to ensure DC-balance.” Shin at [0128].

72. A POSITA would have understood that Shin’s description of generating symbols to “optimize transition control and to ensure DC-balance” discloses generating data that is encoded in accordance with a line code, since one function of line code is to transmit data in manner that helps ensure DC-balance between the receiver and the transmitter. Shin at [0071]. For instance, Shin’s host’s (*first endpoint*) function of “encod[ing] codes into symbols” describes a block encoding method that generates encoded symbols from the codes, as described by Shin’s example of encoding “8-bit code[s] to . . . 9-bit symbol[s]” using “8b/9b” block encoding. Shin at [0129], [0131]. As even noted by the Background of the Invention for the ’307 Patent:

Line coding techniques are often employed in serial, digital transmission (at the physical layer) to transform the frequency spectrum of the transmitted data stream to adapt to channel characteristics and facilitate clock/data

recovery by receiver systems. **This is typically accomplished using a block code which encodes a number of user data bits as a larger number of bits,** according to some pre-determined table or function. The added bits of overhead typically **ensure a high transition density to facilitate clock/data recovery, and accomplish DC balancing** to maintain running disparity near zero, which enables the use of AC-coupling.

'307 Patent 1:34-44 (Background of the Invention). Thus, it would have been understood by a POSITA that the host communication node (*the first endpoint*) is generating encoded data *in accordance with line code* by using a block code to generate symbols in a manner that “ensure[s] DC-balance.” Shin at [0128].

73. A POSITA also would have understood that Shin discloses generating encoded data in accordance with a line code, because Shin discloses digitally encoding binary data to be transmitted over a line (e.g, link 140). “Line coding” is “the process of converting binary data,” e.g., “sequences of bits,” into “a digital signal” to be transmitted from a transmitter over a line to a receiver. Ex.1018, at 2. Thus, “line code” is the code transmitted digitally over a serial link from a transmitter to a receiver. “In telecommunications, a line code is a pattern of voltage, current, or photons used to represent digital data transmitted down a communication channel or written to a storage medium.” Ex.1019, at 1. Thus, by transmitting data across link 140, Shin discloses generating *encoded data in accordance with a line code* simply by transmitting data. Shin at [0077] (“transmits the serialized symbols via the communications link.”). When used with block encoding methods, encoded data

blocks (like Shin's symbols) can be transmitted as line code. *See, e.g., '307 Patent*, at Abstract ("The line code specifies a block code"), 4:53-56 ("a block code" can encode "M-bit input words" into "N-bit code words" where " $M < N$ "), 5:15-18 ("the line code employed to transmit encoded data over a serial link specifies . . . a block code").

74. Additionally, an example encoding disclosed in Shin is 8B/9B encoding, which encodes groups or sequences (i.e., blocks) of 8 bits of data into 9-bit symbols (*encoded data*). Shin at [0131]. A POSITA would have understood that Shin's 8B/9B block encoded data, which includes symbols, is generated *in accordance with a line code* because Shin discloses using 8B/9B encoding that maintains the benefits of DC-balance, clock recovery, and error detection—which uses a line code. Shin at Abstract, [0071]. [0077], [0123]-[0124], [0128].

75. Further, Shin discloses a specific example line encoding technique. As also noted in Shin, selective block inversion can be performed to transition optimize a block code. Shin at [0131]-[0132] ("The encoding technique may also help ensure DC-balance by using selective block inversion encoding on symbols that have been transition optimized"). A POSITA would have understood that selectively inverting the encoded data blocks (Shin's "symbols") to generate DC-balance uses a line code. *Id.* at [0132]. This is because the selective block inversion is done to generate bits values that are transmitted across the line. Shin at [0128] ("For example, the result

of alternate bit inversion of a code with zero transitions (e.g., "11111111) is a code with seven transitions (e.g., "01010101") The encoding technique adds a ninth bit (i.e., a transition inversion indicating bit) to the 8-bit code to indicate whether the alternate bits of the code were inverted, which results in a 9-bit symbol.”). As noted above, the line code is the code transmitted across the line. Thus, Shin discloses an example of generating line code by inverting block code for transmission. As such, Shin’s host transmitter (*the first endpoint*) is configured to generate encoded data in accordance with a line code (at least by selectively inverting 9-bit symbols), and to transmit the encoded data (the selectively inverted symbols) over the link (the serial communication link) to the second endpoint (the data store device).

[1(e)] wherein the physical and link layer circuitry of the second endpoint is coupled to the link and configured to decode the encoded data

76. Shin discloses this limitation.

77. “The physical layer [of *the second endpoint*] receives serialized symbols via the communications link [*the link*], deserializes the symbols, and provides the symbols to the link layer.” Shin at [0077]. At the link layer, “[t]he data decoder receives symbols [*the encoded data*] from the physical layer and decodes them.” Shin at [0079], Fig. 1.

78. Thus, Shin discloses *physical and link layer circuitry of the second endpoint is coupled to the link and configured to decode the encoded data* at least

by deserializing encoded data at the physical layer and passing it to the link layer to decode it. Shin at [0079].

[1(f)] to recover application data and control bits,

79. Shin discloses this limitation.

80. Shin teaches that its transmitting node (e.g., a host) transmits “data packets” that include “blocks of data to be stored” at the receiving node (e.g., a data store device). Shin at [0090], [0088], Fig. 5, [0093]-[0094], Figs. 7-8. In addition to data packets, the host also transmits primitives, such as synchronization primitives that encode packet type, which may be transmitted immediately prior to packets, and the host may also transmit control packets. Shin at [0097]. As shown in the annotated version of Figure 9A below, which depicts information transmitted on a serial link, a synchronization primitive 901 encoded with a packet type of “control” is transmitted immediately prior to a control packet 902, and a synchronization primitive 905 encoded with a packet type of “data” is transmitted immediately prior to a data packet 907.

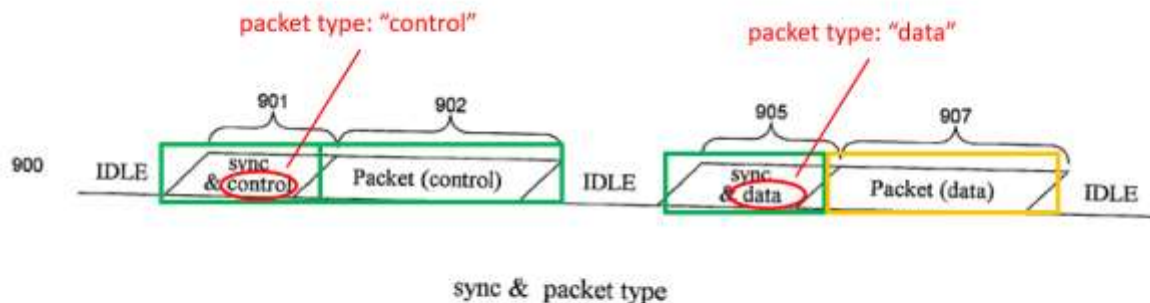


Fig. 9A

Shin at Fig. 9A (annotated), [0097]. “The receiving communications node uses the synchronization primitive encoded with packet type to align its symbol boundaries and to rapidly identify the packet type.” Shin at [0097]. As shown in the annotated version of Figure 9C below, a link layer 910 of a transmitter is provided with bits of the code (labeled “Data”) (**orange**) that is encoded to be transmitted and with an indication of a control primitive (labeled “CTL-S”) (**green**) that is generated to be transmitted. Shin at [0105].

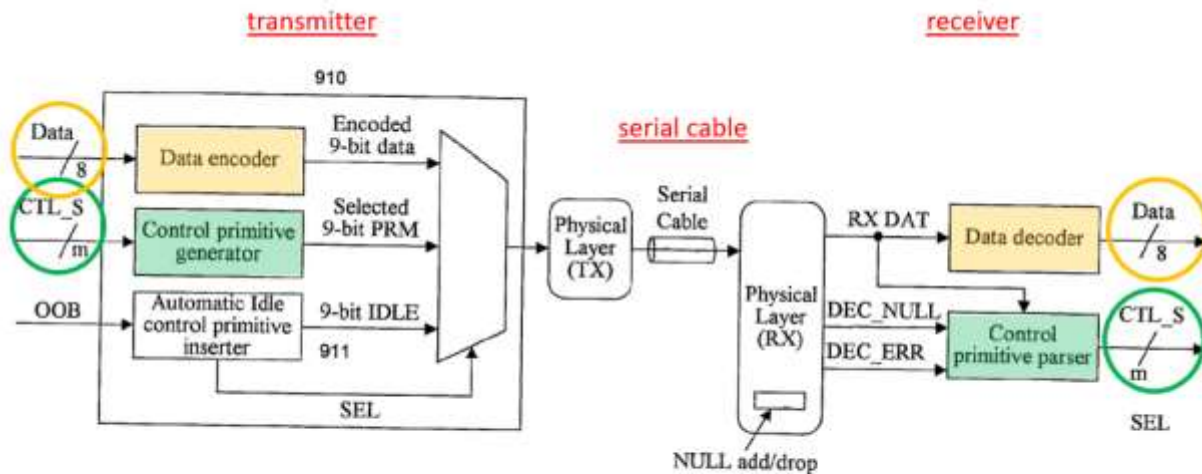


Fig. 9C

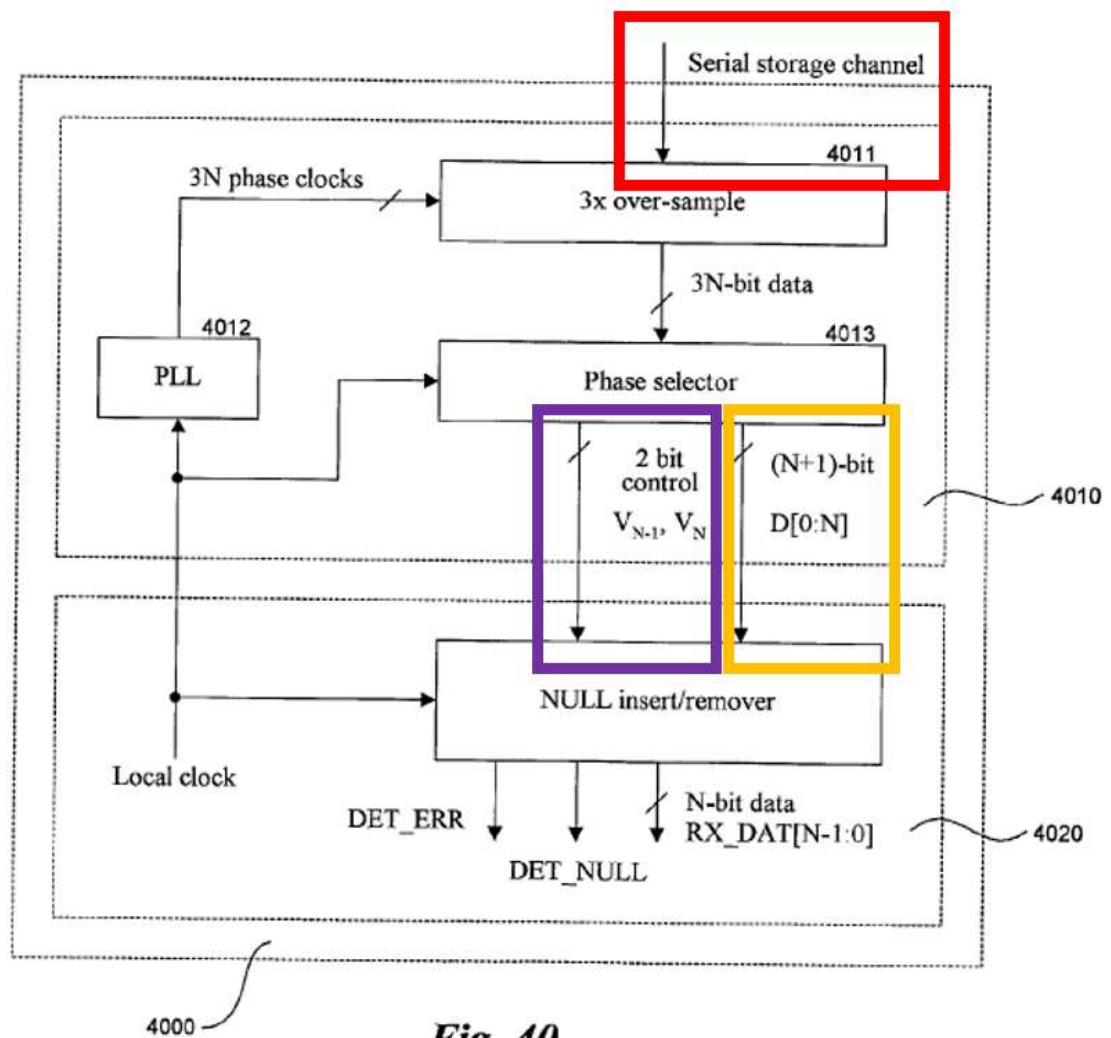
Shin at Fig. 9C (annotated). As shown, after receipt, the encoded data is decoded to recover bits of data (e.g., 8 bits) and the control primitives are parsed to recover “m” control bits. Shin at Fig. 9C, [0077] (describing decoding at the receiver), [0079] (same), [0146] (describing decoding primitives), Fig. 29. A POSITA would recognize the “slash” notation, illustrated in Figure 9C as “/m,” to mean “m bits.”

81. The decoded “Data” that is transmitted via “data packets” (**orange**) constitutes *recover[ed] application data*, and the “m” control bits recovered from parsing control primitives (“CTL_S”) (**green**) and from decoding control packets (**green**) constitute *recover[ed] control bits*, since a POSITA have understood the notation “/m” as recovering “m bits.” A POSITA would have further understood that the a preempt primitive would be recovered as “m” control bits because a preempt primitive is a control primitive. Shin at [0112] (“The preempt and continue primitives, which are control primitives, serve to delimit transmission of a preempting packet.”).

82. Shin further teaches that synchronization primitives encoding packet type are recovered as control bits. A POSITA would have understood that the bits of the synchronization primitive are recovered by a receiver (a second endpoint) because the receiver buffers the bits within the synchronization primitive. Shin at [0104] (“In one embodiment, the synchronization primitives are detected by buffering three symbols (e.g., 27 bits). The synchronization primitives are selected So that any time the buffer has exactly three of the synchronization detection mode bits (e.g., ‘10’ for falling edge) that are each 9 bits apart then the bits are at the beginning of a synchronization symbol”). Shin teaches this is done at the physical layer because the “receiver maintains a small buffer (e.g., 27 bits of the received data) at the physical layer. A POSITA would have understood that, for the receiver

to buffer the bits of the synchronization primitive (*control bits*), the receiver would have *recovered* the bits of the control primitive.

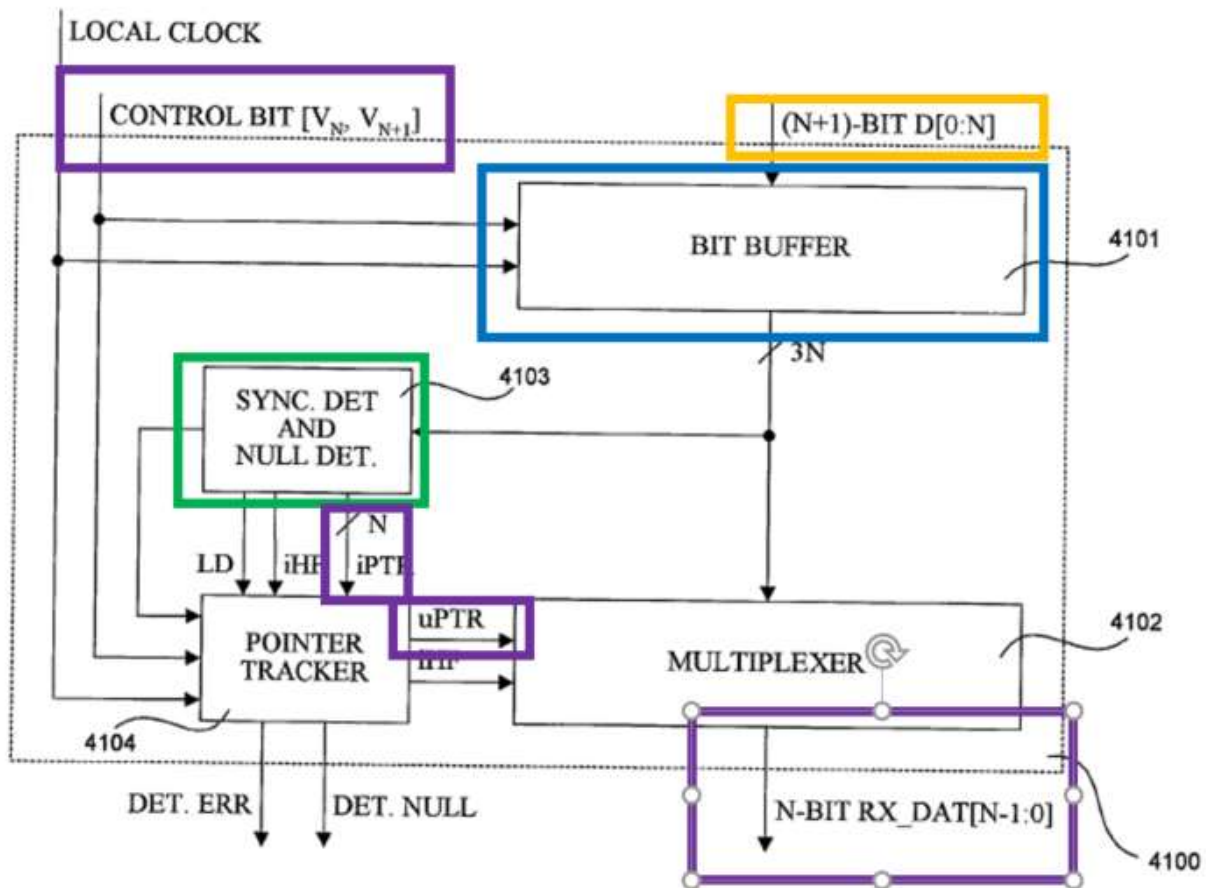
83. Further, Shin discloses that a “CONTROL BIT $[V_n, V_{n+1}]$ ” (Shin at Fig. 41) may be recovered from a synchronization primitive, as depicted in Figures 38A, 39B, 40, and 41, and described in [0163], [0165], [0168], and [0170]-[0171]. As illustrated below in Figure 40 from Shin, a POSITA have understood that encoded data from a transmitter is received at the physical layer of a receiver via the serial storage channel annotated with **red**. Shin at [0169]. The encoded data would includes synchronization primitives inserted by the transmitter. Shin at [0097]. As illustrated in Figure 40, sampler 4011 receives the encoded data with the synchronization primitives and outputs valid bits $[V_{N-1}, V_N]$ (**purple**) to frame aligner 1420. Shin at [0169]. Sampler 4011 outputs the data bits from the encoded data as data lines $D[O:N]$ (**orange**). Shin at [0169].

**Fig. 40**

A more detailed schematic of the sampler is provided by Figure 41 of Shin, which shows frame aligner 4100 of Figure 41. Shin, at [0069]. Below, Figure 41 of Shin

shows the control bits $[V_N, V_{N+1}]$ (**purple**)³ being received and placed in bit buffer 4101 (**blue**). Frame aligner 4100 also receives the data bits from the data lines $D[0:N]$ (**orange**). Shin, at [0169] (“The sampler outputs $N+1$ bits (where N is the number of bits in a symbol) of data lines $D[0:N]$ and two valid bits V_{N-1} and V_N .”).

³ A POSITA would have appreciated that Figure 40 illustrates an underrun ($N-1$) clock scenario, while Figure 41 illustrates an overrun clock scenario ($N+1$). Shin at [0169] (“If, however, there is an overrun, then the sampler provides $N+1$ data bits; that is, $D[N-1]$ and $D[N]$ are both valid. If there is an underrun, then the sampler provides $N-1$ data bits; that is, $D[N-1]$ and $D[N]$ are both invalid.”).

**Fig. 41**

The control bits $[V_N, V_{N+1}]$ (purple) from the synchronization primitive are provided to frame aligner 4100 and placed in bit buffer 4101 (blue). Shin at [0170], Fig. 41. The sync and null detector (green) detects a synchronization symbol by scanning the bit buffer (blue). Shin at [0170] (“The sync and null detector scans the bit buffer to detect whether it contains a NULL primitive and, in particular, a synchronization primitive.”). “If a synchronization primitive is detected, [the sync and null detector] signals the symbol pointer tracker” with a “9-bit synchronization symbol pointer

(‘iPTR’).” Shin at [0170]-[0171], Fig. 41. In response, the pointer tracker outputs a uPTR signal that is a “9-bit start of symbols pointer” to a multiplexer, which in response, outputs a 9-bit RX_DAT having the synchronization primitives. Shin [0171]-[0172]. The flow of the valid control bits, through the iPRT, the uPTR, and the N-Bit RX_DAT is illustrated using **purple**.

84. In Figure 38A, illustrated below, the RX_DAT signal now having the bits of the preempt primitive is provided from the physical layer 3813 to the link layer within 3812, annotated in **purple**. Shin at [0164].

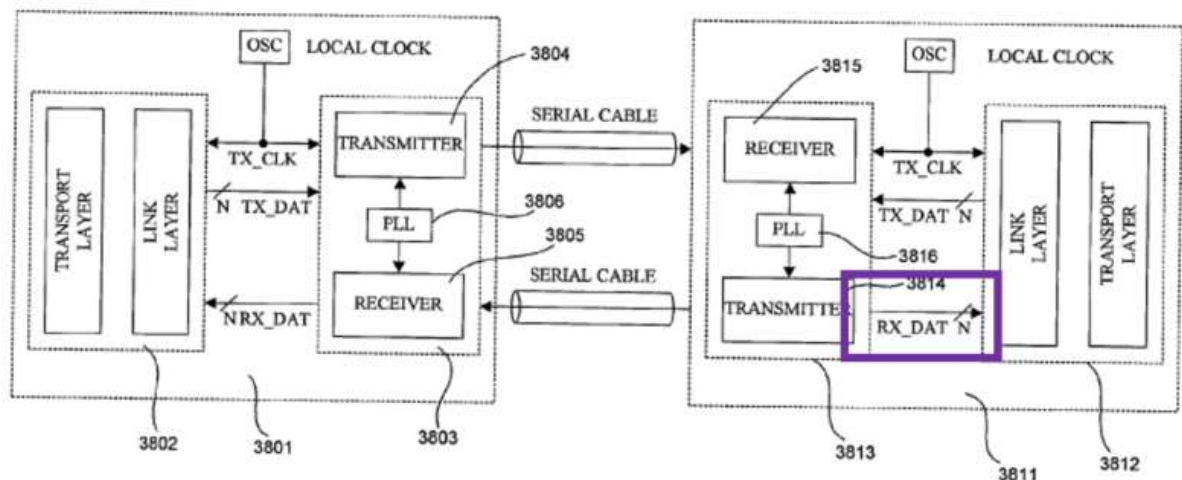


Fig. 38A

85. RX_DAT is similarly illustrated being transmitted from the link layer to the physical layer in FIG. 39B, annotated using **purple**. Shin at 168 (“The clock and data recover (‘CDR’) processing of the physical layer 3950 provides to the link layer RX_DAT lines that are synchronized to the local clock.”); Shin at [0167] (“In the link layer, there are two clock domains, the TX clock domain 3921 that is the

same as the local clock domain.”). As such, a POSITA would have understood that synchronization primitives are recovered as control bits at the receiver (the second endpoint).

86. Further, Shin discloses that all of its primitives are comprised of “bits” that are recovered from the primitives when received, and these bits are “control bits” because they cause functions to be performed by layers of the receiver. Shin at [0098], [0096]-[0097], [0115]; ’307 Patent at 5:29-32 (describing control bits as out-of-band bits), 6:45-49, 6:66-7:9, 7:60-64 (discussing how control bits cause functions at any OSI model layer, including the network layer, application layer, and link layer).

87. Additionally, a POSITA would have understood that, since the 8-bits of data (**orange**) and the “m” control bits (**green**) and the “CONTROL BIT[S]” are each recovered by decoding packets and parsing primitives, they are recovered from the line code so as to allow the digital signal transmitted across the line to be converted back to bits. Since the line code is the code that is transmitted (as note previously) recovery of the data bits and control bits as illustrated in Figure 9C also constitutes recovery from the line code because the physical layer of the receiver (RX) can be seen receiving the encoded data from the serial cable, as discussed at limitation [1(d)], noting that line code is encoded data transmitted across a line to a receiver).

88. Shin also discloses that the physical and link layer circuitry of the second endpoint is configured to decode the encoded data to recover application data and control bits, because Shin teaches that the receiver's (second endpoint's) physical layer deserializes the received symbols and provides them to the receiver's link layer, which decodes those symbols. Shin at [0077].

[1(g)] to perform at least one link level function in response to at least one of the control bits, and

89. Shin discloses this limitation.

90. Specifically, Shin discloses that the link layer of its receiving node can perform the function of detecting packet type (*perform[ing] at least one link level function*) in response to receiving a synchronization primitive encoded with packet type (*at least one of the control bits*). Shin at [0096]-[0097].

The link layer can then transmit a synchronization primitive that indicates packet type immediately before the packet itself. **The receiving communications node can take the appropriate action based on the packet type encoded in the synchronization primitive without having to parse the packet header. . . .** The rapid detection of packet type through a synchronization primitive, rather than through a packet header, can reduce the processing time of a Switch. Also, when a synchronization primitive encodes the packet type, **then the link layer, rather than the transport layer, can detect packet type**, which can enhance the overall processing speed.

Shin at [0096].

91. Because Shin's receiving node performs the link level function of detecting packet type in response to receiving a synchronization primitive encoded with packet type, this function is performed *in response to at least one of the control bits*. This is because, the function is performed in response to the control bits from the synchronization primitive, such as in response to the control bits recovered from received primitives (Figure 9C, Figures 40-41), as discussed above for limitation [1(f)]. *See also* Shin at Fig. 9C, [0098]. A POSITA, therefore, would have understood that these link level functions, such as identifying packet type, are performed in response to at least one control bit generated in response to parsing a received synchronization primitive encoded with packet type.

[1(h)] to assert at least one control signal indicative of at least another one of the control bits to the higher level circuitry, and

92. Shin discloses and renders obvious this limitation.

93. Shin discloses that when the link layer receives and parses a preempt primitive to recover bits (*another one of the control bits*), the link layer "signals" to the transport layer component that a preempt primitive has been received (*assert at least one control signal indicative of at least another one of the control bits to the higher-level circuitry*). Shin at [0115]. The transport layer component then performs the function of saving current packet information so it can resume storing codes in the packet after preemption is complete (*wherein the higher-level circuitry is*

configured to perform at least one higher level function in response to the at least one control signal). Shin at [0115], Fig. 15. Specifically, Shin states:

FIG. 15 is a logic diagram illustrating processing of packet preemption by a receiving communications node in one embodiment. **In block 1501, the component of the transport layer retrieves the next code from the link layer. In decision block 1502, if the link layer signals that a preempt primitive has been received, then the component continues at block 1504 . . . In block 1504, the component saves current packet information so that it can continue storing codes in the packet after the preemption is complete.** In block 1505, the component processes the preempting packet. The processing includes receiving each code and storing the codes in a preempting packet until the link layer indicates that it received a continue primitive.

Shin at [0115]. Thus, in Shin, the link layer *assert[s] at least one control signal* to the transport layer component (*higher level circuitry*) where the signal is *indicative of at least another one of the control bits*, because the link layer “signals” the transport layer component in response to receipt of a preempt primitive, which, as discussed above for limitation [1(f)], generates “m” control bits upon the primitive being received and parsed. *See* [1(f)]. Looking at annotated Figure 9C, reproduced below, a POSITA would have understood the preempt primitive is recovered as control bits because a preempt primitive is a control primitive, and the control primitives are recovered by the illustrated control primitive parser as discussed

above at limitation [1(f)]. Shin at [0112] (“The preempt and continue primitives, which are control primitives, serve to delimit transmission of a preempting packet.”).

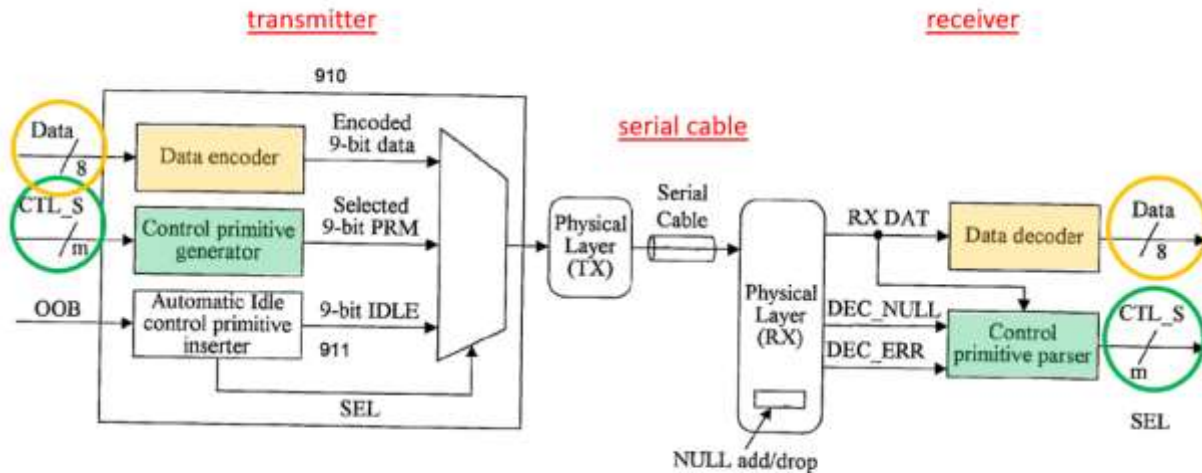


Fig. 9C

Thus, a POSITA would have understood that the “signal” asserted by the link layer to the transport layer is inductive of control bits because the signal occurs in response to the control bits generated from parsing the preempt primitive. Shin at [0115] (discussing the transport layer component performing the function of saving current packet information based on signal asserted by the link layer that is sent upon receiving the preempt primitive).

94. A POSITA would have further understood that the control bits recovered from the preempt primitive would constitute *at least another one of the control bits*, as compared to the control bits (*at least one of the control bits*) recovered from the synchronization primitive encoded with packet type, at least to the extent that the term “another one of the control bits” encompasses one or more

different control bit values and does not require one or more different control bit fields. This is because a POSITA would have understood that these two primitives, at a minimum, generate different control bit values when received, because the preempt primitive is a different primitive having different functionality as compared to the synchronization primitive encoded with packet type. See Shin at [0112] (“The preempt and continue primitives . . . are control primitives.”), [0101] (“The synchronization primitives are distinct from the control primitives.”). Thus, the control bits of the preempt primitive constitute *at least another one of the control bits* relative to control bits from the synchronization primitive when they are recovered.

95. To the extent the term “*at least another one of the control bits*” is interpreted to require one or more different control bit fields as compared to the first “*at least one of the control bits*,” then a POSITA would have found it obvious and would have been motivated to implement Shin’s synchronization primitive encoded with packet type and the preempt primitive using different control bit fields for several reasons. First, a POSITA would have known that there would have been a finite number of foreseeable solutions—either i) encode the primitives using separate control bit values in common bit fields, or ii) encode the primitives using separate bit fields. As such, it would have amounted to a routine design choice between two alternatives that would have been obvious to a POSITA.

96. Second, encoding control primitives using separate bit fields was a well-known solution in the field at the time, as demonstrated by, for example, the SATA 1.0a standard that had been adopted and publicly released in early 2003, prior to the filing of the '307 Patent, and that was intended for providing serial connections to storage devices (like Shin). SATA 1.0a specification (Ex.1009) at 23 (“Serial ATA [SATA] is a high-speed serial link replacement for the parallel ATA attachment of mass storage devices. The serial link employed is a high-speed differential layer that utilizes Gigabit technology and 8b/10b encoding.”), 143-144, Fig. 57. That it was well known to encode control primitives in this field of art by using separate bit fields is demonstrated by SATA’s teaching of using primitives both within and outside of frames and including primitives in dedicated bit fields, such as the SOF primitive that always starts a frame and marks the start of a frame and the EOF primitive that always ends a frame and marks the end of a frame. SATA 1.0a at 143-144, 146-147, Fig. 57. These SATA transmission structures are shown below:

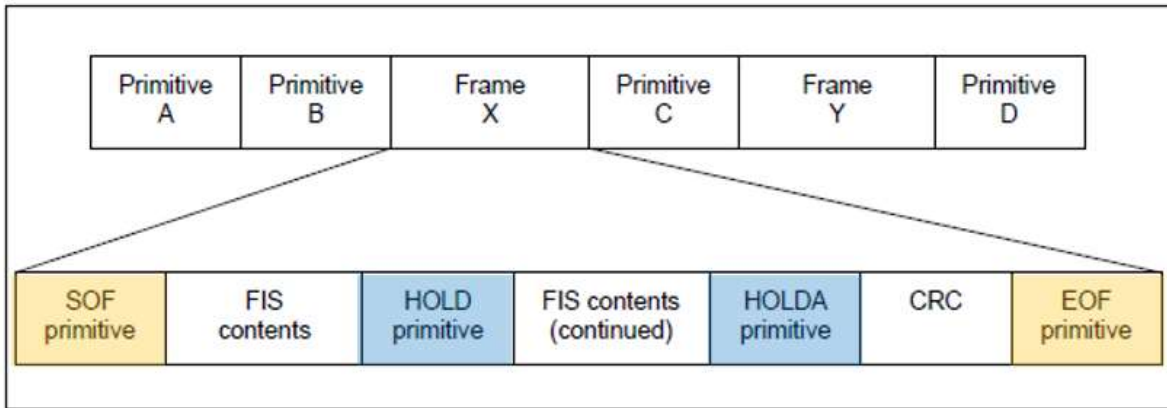


Figure 57 – Transmission structures

SATA 1.0a at Fig. 57 (annotated), 144. As illustrated using FIG. 57, the SOF primitive and the EOF primitive separate field, i.e., bit fields of the same frame (annotated in **yellow**). This is further illustrated by separate fields the HOLD primitive and the HOLDA primitive (annotated in **blue**). That it was well known in this field to encode control primitives using separate bit fields is also demonstrated by the HDMI 1.0 standard released in 2002, which also teaches using separate bit fields to encode control information, such as the bit fields CTL0, CTL1, CTL2, and CTL3, as shown below. HDMI 1.0 (Ex.1010) at 42-43, 43 (Table 5-2); Ex.1013.

Table 5-2 Preambles for Each Data Period Type

CTL0	CTL1	CTL2	CTL3	Data Period Type
1	0	0	0	Video Data Period
1	0	1	0	Data Island Period

HDMI 1.0 at 43 (Table 5-2). For example, CTL2 was known to encode packet type, while CTL3 was known to encode encryption status. HDMI at Table 42-43. Thus,

a POSITA would have been motivated to encode control primitives using separate control bit fields because it was well-known in the art.

97. Third, Shin discloses that there are up to 18 bits available for control bit encoding that can represent up to 256 out-of-band symbols Shin at [0158]. Thus, a POSITA would have had a reasonable expectation of success of encoding the synchronization primitive and preempt primitive using separate control bit fields, because this was a well-known technique and Shin provides more than enough control bits allowing such to be implemented. For instance, primitives are pairs of 9-bit symbols (18 bits). Shin at [0098], [0100], Table 2. Since only 8 bits are needed to encode the 256 out-of-band symbols, the remaining bit fields could be used to separately encode the primitives.

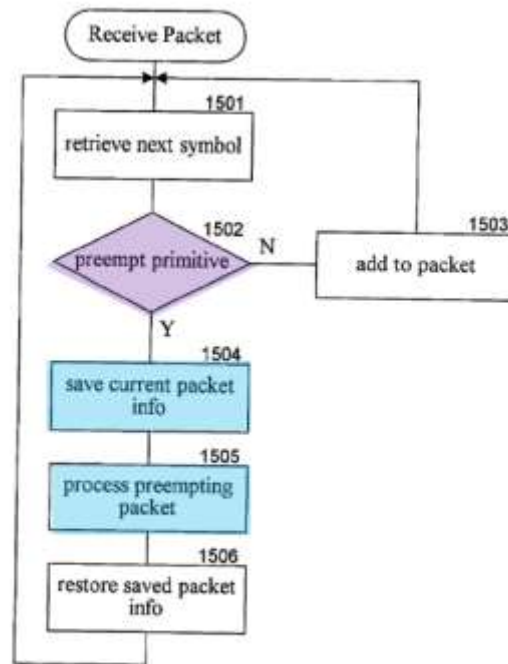
98. Therefore, it would have been an obvious and a well-known design choice from among two finite options for a POSITA to encode Shin's primitives in separate control bit fields, to the extent such is deemed to be required for this limitation [1(h)]. Shin, therefore, discloses and renders obvious this limitation.

[1(i)] wherein the higher level circuitry is configured to perform at least one higher level function in response to the at least one control signal.

99. Shin discloses this limitation.

100. As discussed for limitation [1(h)], "if the link layer signals" to the component of the transport layer (*higher level circuitry*) "that a preempt primitive has been received" in "decision block 1502," then "the component continues at block

1504 . . . [and] the component saves current packet information [*perform[s] at least one higher level function*] so that it can continue storing codes in the packet after the preemption is complete.” Shin at [0115], [0112], Fig. 15. The transport layer component then, in block 1505, also “processes the preempting packet,” which “includes receiving each code and storing the codes in a preempting packet until the link layer indicates that it received a continue primitive,” which also constitutes *perform[ing] at least one higher level function in response to the at least one control signal*, because it is done in response to the link layer signaling the transport layer that the preempt primitive has been received. Shin at [0115], Fig. 15. A POSITA would have understood that the transport layer’s function of receiving and storing codes of the preempting packet is a higher level function because the transport layer is higher than the link layer. ’307 Patent at 5:8-8 (Higher level circuitry is “circuitry that operates at [] least one level of communication protocol functionality higher than the link level.”). Figure 15 of Shin is shown below, with the block 1502 step of the link layer asserting the control signal to the transport layer highlighted in purple, and with the blocks 1504 and 1505 showing the higher level functions performed by the higher transport layer in response to the control signal highlighted blue:

**Fig. 15**

Ex.1004 at Fig. 15 (annotated).

3. Claim 2

[2] The system of claim 1, wherein the line code specifies a block code and special characters, the first endpoint is configured to encode M-bit input words in accordance with the block code to generate a code word sequence of N-bit code words, where $N > M$, the code word sequence is indicative of at least one cell of the input words, and the encoded data are a sequence of at least one of the special characters and the code words of the code word sequence.

101. Shin discloses and renders obvious this limitation.

102. As described above with respect to limitation [1(d)], Shin discloses a first endpoint (a host transmitter) that generates encoded data in accordance with line code.

103. Shin further discloses that *the line code specifies a block code*. A POSITA would have understood that Shin's communications node transmits line code that includes encoded symbols of a block code, because Shin's 8b/9b encoding encodes groups or sequences (i.e., blocks) of 8 bits of data into 9-bit symbols that are transmitted digitally across a line. Shin at [0128]. ("The link layer of a communications node encodes codes into symbols in order to optimize transition control and to ensure DC-balance."). A POSITA would have understood that codes are encoded into symbols using block encoding because Shin describes an example 8B/9B block encoding method where "[t]he encoding technique adds a ninth bit . . . to the 8-bit code . . . which results in a 9-bit symbol," which is then transmitted digitally across a line. Shin at [0129]. As such, Shin's *line code specifies a block code*. Further, as noted above with respect to limitation [1(d)], the line code is the code digitally transmitted from a transmitter to a receiver across a line. So, because Shin's transmitted line code includes blocks resulting from 8b/9b block encoding, as described by Shin, a POSITA would have understood that the line code specifies a block code during transmission.

104. Further, Shin discloses that line code also specifies *special characters*. Shin describes out-of-band symbols that are *special characters* because an out-of-band symbol, such as an IDLE symbol, is "inserted into a sequence of in-band symbols" and "does not have the same desirable characteristics as in-band symbols."

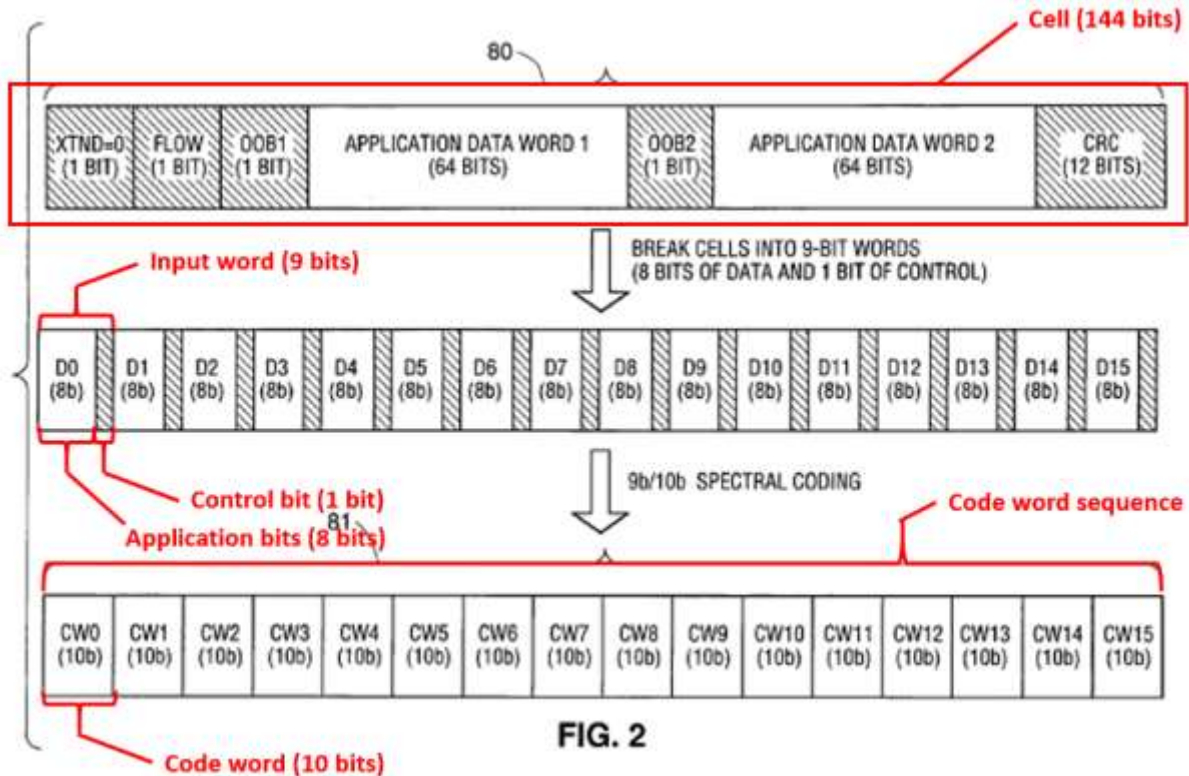
Shin at [0086]. Likewise, the *special characters* described by the '307 Patent are “inserted in a stream of code words” and are “distinguishable from . . . code word sequences.” '307 Patent at 18:48-50, 27:66-28:2. A POSITA would have understood that Shin's out-of-band symbols are *special characters* because both are being inserted into encoded data blocks and because the out-of-band symbols do not have the same characteristics as the encoded data blocks. Shin at [0086] (describing inserting out-of-band symbols into a sequence of in-band symbols; '307 Patent at 27:66-28:2 (“special characters (‘out-of-band’ or ‘OOB’ characters) . . . can be inserted in a stream of code words generated in accordance with the 9B/10B code.”)). A POSITA would have further understood that *the line code specifies a block code and special characters* because the out-of-band symbols (*special characters*) can be inserted “into the sequence of symbols that are to be transmitted” using the line code. Shin at [0079]. Moreover, a POSITA would have understood, because Shin's special characters are transmitted with the sequence of symbols, the special characters are also specified by the line code, since it is the line code that is responsible for the digital transmission across the line.

105. Shin also teaches the first endpoint is configured to encode M-bit input words in accordance with the block code to generate a code word sequence of N-bit code words, where $N > M$. As described above and with respect to limitation [1(d)], Shin teaches an 8B/9B encoding method for the first endpoint to encode data. Shin,

at [0129] (“The encoding technique adds a ninth bit . . . to the 8-bit code . . . which results in a 9-bit symbol.”). A POSITA would have understood that the 8B/9B encoding scheme used by the first endpoint is a block encoding scheme that encodes eight-bit (M-bit) input words into a code word sequence of nine-bit (N-bit) code words in accordance with the 8B/9B encoding, and thus $N > M$. The M-bit input word to N-bit symbols is a common notation that would have been understood by a POSITA. The background of the ’307 Patent provides another example block encoding technique where $N > M$ where it discusses IBM’s 8B/10B encoding. ’307 Patent at 3:62-64. A POSITA would have understood that, in the 8B/10B encoding, “eight bits of data are transmitted as a 10-bit entity called a symbol, or character,” and like Shin’s 8B/9B, the M-input words have a number of bits (8) that is less than the N-output symbols (10). Ex.1020, at 1. Such encodings were well known in the art. Ex.1020.

106. Shin further teaches that *the code word sequence is indicative of at least one cell of the input words*. According to the ’307 Patent, a “cell,” in the context of “packet-based (cell-based) communication,” includes “application and control bits.” ’307 Patent at 4:37-38. “Each cell has a predetermined structure, so that the position of a bit within the cell determines whether the bit is an application data bit or a control bit and determines the type (function) of each control bit.” ’307 Patent at

4:63-66. Annotated Figure 2 “shows an example of such a cell, consisting of 144 bits.” ’307 Patent at 14:23-25.



’307 Patent at Fig. 2 (annotated). “The bits of each cell are grouped into input words” having “application data bits . . . [and] control bit[s].” ’307 Patent at 14:26-31. “[I]nput words indicative of cell 80 are encoded . . . using a 9-Bit-to-10-Bit (‘9B/10B’) block code to produce the sixteen 10-bit code words (CWO—CW15) shown. The code words are transmitted in an appropriate sequence over a serial link.” ’307 Patent at 14:48-53.

107. Likewise, Shin describes a packet (*cell*) that is encoded and transmitted using block code. Shin at [0076]. For example, a “packet includes a header section

and a payload section” Shin at [0076]. Further, like a “cell” Shin’s packet has predetermine structure, such as specific bit fields. Shin at [0084] (describing various bit fields in the packet). “Packet” and “cell” are similarly used by the ’307 in this context, as it says in the Summary of the Invention that “line code provides packet-based (cell-based) communication.” ’307 Patent at 4:37-40.

108. “The transport layer receives the data of the transaction and generates packets (i.e., packetizes the data) for transmission. The transport layer then provides each code (e.g., each byte) of the packets to the link layer.” Shin at [0076]. Therefore, the “codes” are the pre-encoded data from the packet and include the M-bit input words.

109. “The link layer receives codes to be transmitted from the transport layer encodes the codes into symbols, and provides those symbols to the physical layer. Shin at [0077]. The link layer can use block encoding to encode the words into symbols, such as an “encoding technique [that] adds a ninth bit . . . to the 8-bit code . . . , which results in a 9-bit symbol.” Shin at [0129]. This generates “a sequence of symbols that are to be transmitted.” Shin at [0079]. “The physical layer receives the symbols from the link layer, serializes the symbols, and transmits the serialized symbols via the communications link.” Shin at [0077]. Therefore, the “symbols” are the codes that have been encoded using block encoding, and thus, the symbols are the N-bit output of the encoded M-bit codes.

110. Thus, Shin discloses generating codes (*input words*) from a packet (*cell*) and encoding the codes (*input words*) into symbols (*code words*) using block encoding techniques. As such, Shin's sequence of symbols (*code word sequence*) is indicative of at least one packet (*cell*) of the codes (*input words*).

111. Lastly, Shin teaches that the encoded data are a sequence of at least one of the special characters and the code words of the code word sequence. Shin discloses a data selector at the link level that "effectively inserts control symbols [special characters] and synchronization symbols into the sequence of symbols [code word sequence] that are to be transmitted." Shin at [0079]. As noted, out-of-band symbols are special characters, and Shin also refers to out-of-band symbols as "control symbols." Shin at [0086] ("Because control information is encoded as out-of-band symbols, the control symbols can be inserted into a sequence of in-band symbols and readily detected as control symbols when received.").

112. In a specific example, Shin discloses a special character called an "IDLE symbol" (*at least one of the special characters*), also referred to as an "IDLE primitive." Shin at [0086] ("an IDLE symbol is actually a sequence of two out-of-band symbols"). IDLE symbols (*at least one of the special characters*) can be inserted "after every block of a packet" of in-band symbols, since "a receiving link identifies the start of a packet by a transmission from receiving IDLE symbols to receiving in-band symbols. . . . [and] identifies the end of a packet by a transmission

from receiving in-band symbols to receiving IDLE symbols.” Shin at [0086]; [0105].

Figure 9A from Shin illustrates IDLE symbols, also called “primitives” between packets.

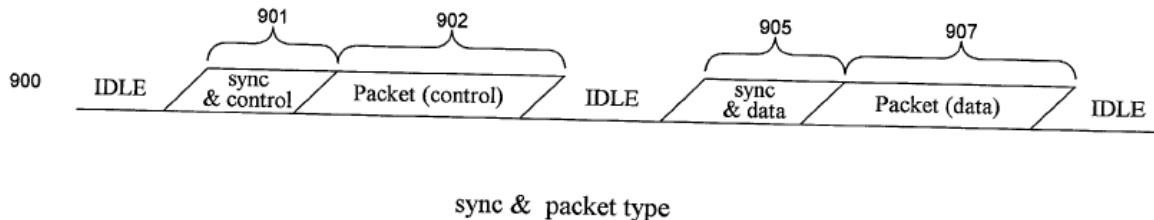


Fig. 9A

As noted, these IDLE symbols are *special characters*, and they are between the packets, which when transmitted are encoded into a sequence of symbols as discussed above. Thus, what gets transmitted as the encoded data is a sequence of the IDLE symbols (*special characters*) and the symbols generated from the packet (*the code words of the code word sequence*). Thus, because a control symbol, such as an IDLE symbol, (*at least one of the special characters*) is inserted into the sequence of symbols (*code words of the code word sequence*) for transmission, such as inserting IDLE symbols between transmitted packets, Shin describes encoded data that is a sequence of at least one special character and the code words of the code word sequence.

4. Claim 6

[6] The system of claim 2, wherein said at least one of the special characters is an idle character.

113. As described above with respect to limitation [2], Shin discloses that *at least one of the special characters is an idle character* (the IDLE symbol). As noted, with respect to limitation [2], special characters are out-of-band symbols, and “an IDLE symbol is an out-of-band symbol.” Shin at [0086].

5. Claim 7

[7] The system of claim 6, wherein the physical and link layer circuitry is configured to perform simplex link flow control by inserting idle characters between subsequences of the code word sequence, where each of the subsequences is indicative of a different cell of the input words.

114. Shin discloses and renders this limitation obvious.

115. Specifically, Shin discloses a receiver (the second endpoint) that inserts a NULL primitive, which is also an *idle character*. A POSITA would have understood that Shin’s NULL primitive (also called a “NULL symbol”) discloses an *idle character* because, like idle characters, the NULL symbols are inserted from symbol streams for clock synchronization using plesiosynchronous communication. ’307 Patent at 49:4-7. The ’307 Patent states, “[A] special idle character” can be inserted “between valid code words (each indicative of contents of a cell) for plesiosynchronous communication.”). ’307 Patent at 49:4-7. The ’307 Patent also states “A slow receiver can drop (ignore) the idle characters to catch up to a faster transmitter.”). ’307 Patent at 49:4-7. Likewise, Shin discloses, “In one embodiment, the serial communications architecture specifies that NULL symbols or NULL primitives are inserted into or removed from a stream of symbols to account for

variations in clock frequency between the transmitting and receiving communications nodes when using plesiosynchronous clocking.” Shin, at [0163]. Generally, “a plesiochronous system is one where different parts of the system are almost, but not quite, perfectly synchronized. . . . the mismatch between the two systems' clocks is known as the plesiochronous difference. . . . [P]lesiochronous systems . . . must employ some means in order to cope with ‘sync slips’, which will happen at intervals due to the plesiochronous nature of the system.” Ex.1022, at 1. A POSITA would have understood that Shin’s use of the NULL primitive “to account for variations in clock frequency between the transmitting and receiving communications nodes” is a form of flow control in a plesiochronous system. Shin at [0163]. Shin’s use of NULL primitives inserted as *idle characters* at the receiver as a form of flow control is further supported by Figure 41 of Shin, which illustrates a single component as both a synchronization primitive detector (“SYNC DET”) and a NULL detector (“NULL DET”). Shin at Fig. 41. The figure is illustrated below and annotated with purple to show the single synchronization primitive and null primitive detector.

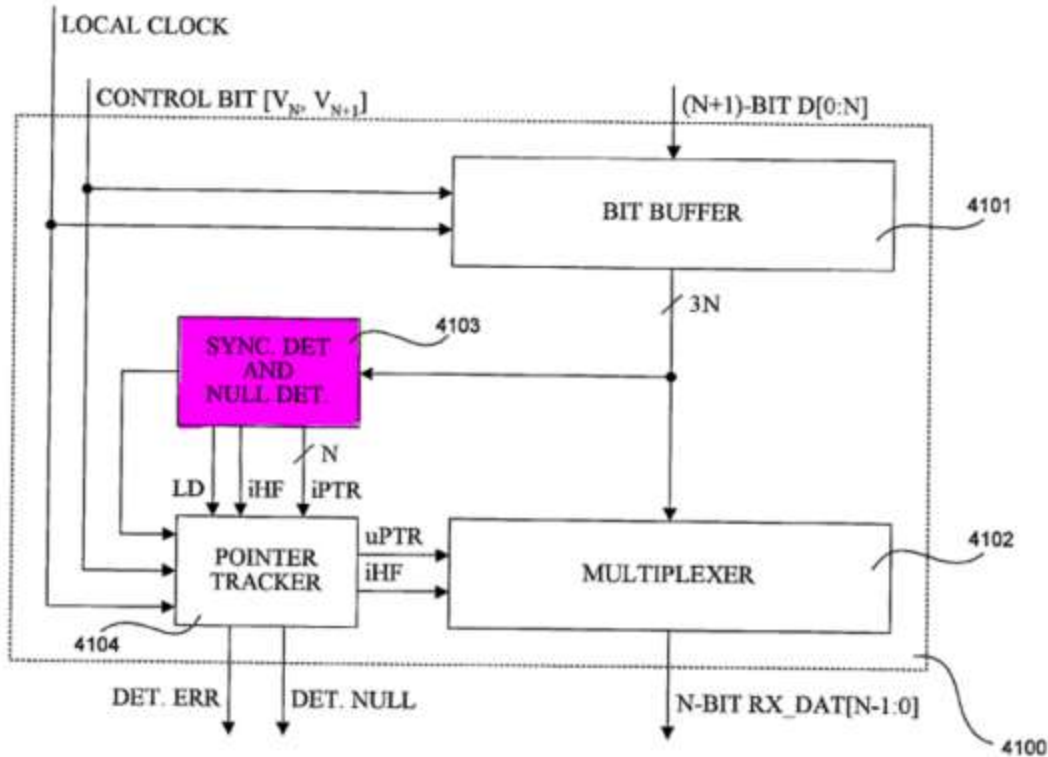


Fig. 41

Since Shin states that synchronization primitives are an example of IDLE primitives, a POSITA would have understood that Shin describes a single component for detecting IDLE primitives and NULL primitives. Thus, a POSITA would have understood Shin's NULL primitive to be a type of *idle character* that is being inserted for flow control.

116. Shin further discloses that *the physical and link layer circuitry is configured to perform simplex link flow control by inserting idle characters*. The NULL primitives are inserted at the physical layer. Shin at [0169] ("NULL insertion and deletion [occurs] at the physical layer."). This is illustrated in annotated Figure

9C below, which shows the *physical layer* of a receiver (RX) (**green**) adding or dropping NULL primitives in a sequence of symbols, illustrated as “DEC_NULL,” (**orange**).

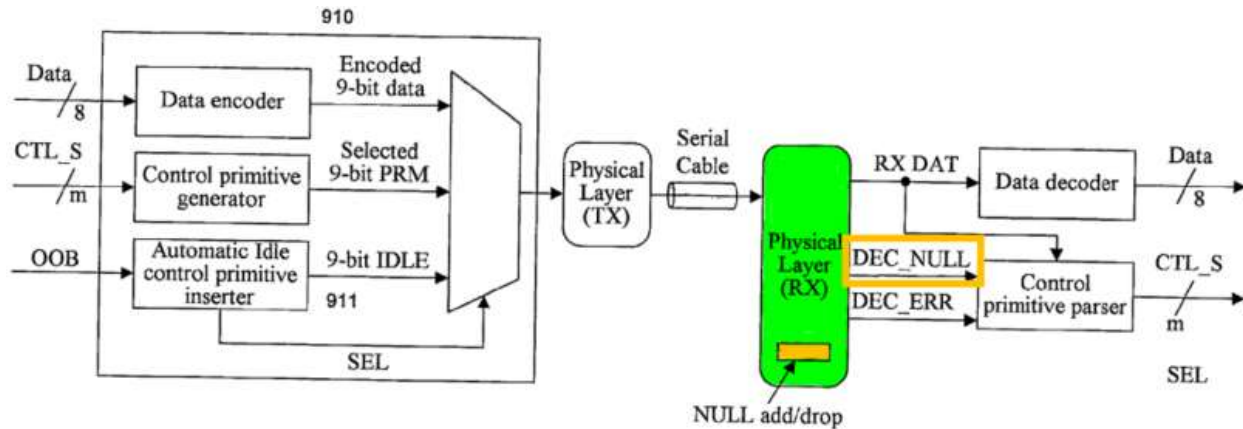


Fig. 9C

117. A POSITA would have understood that Shin inserts the NULL primitive in one direction (i.e., simplex link) for flow control because the receiver inserts the NULL primitive to align the receiver’s clock with the frequency of the transmitter, thus synchronizing the receiver with the transmitter. Shin at [0163] (“These underrun and overrun conditions were typically handled at the link layer. These complexities and costs may, however, be avoided by the physical layer of a receiving communications node periodically inserting or removing NULL symbols or NULL primitives from the received stream of symbols at a rate sufficient to compensate for variations in clock frequencies.”); Figure 9C above also shows a one-directional insertion of NULL primitives at the receiver’s physical layer, as

illustrated by the one-directional arrow from the physical layer to the control primitive parser annotated in **orange**, thus illustrating the use of the NULL primitives as a form of simplex link flow control because it is in one direction.

118. Shin discloses and renders obvious insertion of NULL primitives (idle characters) between subsequences of the code word sequence, where each of the subsequences is indicative of a different cell of the input words. Shin states:

The sync and null detector scans the bit buffer to detect whether it contains a NULL primitive and, in particular, a synchronization primitive. **If a synchronization primitive is detected**, it signals the symbol pointer tracker to align itself on the synchronization primitive. **The sync and null detector also signals the symbol pointer tracker whether a NULL symbol should be inserted or removed from the symbol stream.** The symbol tracker pointer tracks the symbol alignment over time.

Ex.1004, at [0170].

119. Further, as noted above and as illustrated by Shin in 9A, an IDLE primitive (which “may be a synchronization primitive”) can be inserted between packets.

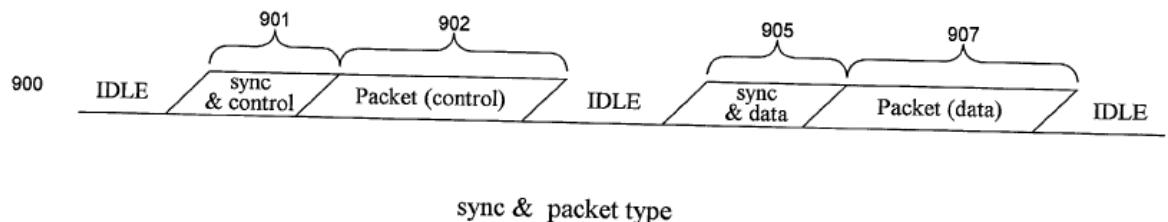


Fig. 9A

Shin at Fig. 9A. As discussed with respect to limitation [2], the packets are transmitted as a sequence of symbols, which is a code word sequence because both are N-bit outputs of the block encoding. Shin at [0079], [0097]. Since Shin's packets are transmitted as a sequence of symbols (*code word sequence*), A POSITA would have understood that Shin's sync and null detector signals for the NULL symbol to be inserted within the symbol stream between packets because it receives IDLE primitives between packets and the NULL symbols are inserted when the IDLE primitive (the synchronization primitive) are detected. Shin at [0170], Fig. 9A. Thus, the NULL symbol (*idle character*) is being inserted within the symbol stream between different packets (*cells*). Therefore, the NULL symbols are inserted between subsequences of symbols indicative different packets (*cells*) encoded from each packets respective codes (*input words*).

120. Additionally, Shin also discloses that “[W]hen the receiver detects a synchronization symbol, it initializes a start-of-symbol pointer in the buffer.” Shin at [0073]. A “tracker . . . inserts a NULL symbol at the start-of-symbol pointer. A POSITA would have understood that since the NULL symbol is inserted at the start-of-symbol pointer upon detecting the synchronization symbol, the NULL symbol is being inserted between packets, since as described above, the synchronization symbol (or IDLE symbols) is transmitted between packets. Thus, for the same reasons as stated above, Shin's insertion of NULL symbols between packets

discloses *inserting idle characters between subsequences of the code word sequence, where each of the subsequences is indicative of a different cell of the input words.*

6. Claim 53

[53(pre)] A receiver for receiving encoded data that have propagated over a serial link, the encoded data having been generated in accordance with a line code, said receiver including:

121. To the extent the preamble is limiting, Shin discloses this limitation for the reasons set forth in [1(d)].

[53(a)] physical and link layer circuitry, coupled and configured to receive and decode the encoded data

122. Shin discloses this limitation.

123. See [1(b)], describing a second endpoint acting as a receiver and including physical and link layer circuitry coupled to higher level circuitry.

124. See [1(e)], describing the physical and link layer circuitry of the second endpoint receives and decodes the encoded data.

[53(b)] to recover application data and control bits

125. Shin discloses this limitation.

126. See [1(f)]

[53(c)] to perform at least one link level function in response to at least one of the control bits; and

127. Shin discloses this limitation.

128. See [1(g)]

[53(d)] higher level circuitry coupled to the physical and link layer circuitry, wherein the physical and link layer circuitry is coupled and configured to assert at least one control signal indicative of at least another one of the control bits to the higher level circuitry,

129. Shin discloses and renders obvious this limitation.

130. See [1(h)], describing the link layer of the second endpoint asserts a control signal to higher level circuitry, and the control signal is indicative of at least another one of the control bits.

131. See [1(b)], describing the second endpoint having higher level circuitry that is coupled to the physical and link layer circuitry.

132. Further, a POSITA would have understood that *the physical and link layer circuitry is coupled to the higher level circuitry* of the receiver because “the link layer . . . signals the transport layer [*higher level circuitry*].” Shin at [0112]. It would have been understood that the higher-level transport layer is coupled to the physical and link layers at least to receive the signal from the link layer.

[53(e)] and the higher level circuitry is configured to perform at least one higher level function in response to the at least one control signal.

133. Shin discloses this limitation

134. See [1(i)]

B. Ground 3: Claims 1, 2, 6, 7, 23, 25, 31, 53, 68, 70, 71, 72, 74, and 75 are Obvious over Yusairi in view of Shin

1. Summary of Yusairi

135. The article entitled “Hardware Design and Implementation of IP-over-1394 Protocol Stack and Its Evaluation,” by Mohd Yusairi, Bin Abu Hassan, and Koki Abe, ITE Technical Report Vol.27, No. 18, pp. 51-85 (March 2003) (“**Yusairi**,” Ex.1005) generally describes using an IPv4-over-1394 protocol stack for communication between devices. Yusairi at 51, Sec. 1. Each device includes a physical layer and a link layer, along with two higher-level network and transport layers. Yusairi at 53-54, Sec. 5, 5.1. The link layer and physical layer, which are part of the IEEE1394 interface (also known as “FireWire”), transmit and receive operations between devices. Yusairi at 51, Sec. 1; Yusairi at 52-53, Sec. 3.2, Fig. 1.

136. When packets are generated by stream_trans at the link layer of the transmitter, a link header is attached to the data. Yusairi at 54, Sec. 5.2. The link header includes a transaction code. Yusairi at 54, Sec. 5.2. When received at the transmitter by stream_rec, the link layer performs a header check process based on the value of the transaction code. Yusairi at 54, Sec. 5.2. During the header check process, stream_rec checks the type field. Yusairi at 55, Sec. 5.2. The link layer then signals the network layer based on the value of the type field. Yusairi at 55, Sec. 5.2. When ip_rec of the network layer detects the ready signal from stream_rec of the link layer, it performs several functions, including reading fifo, checking a destination address of the packet, and calculating an IP checksum, each of which are network layer functions. Yusairi at 55, Sec. 5.3.

2. Summary of Shin

137. See description at ¶¶55-60.

3. Claim 1

[1(pre)] A communication system, including:

138. Yusairi discloses this limitation.

139. Yusairi discloses *a communication system* between two boards, SOPC Board A and SOPC Board B, illustrated Figure 6, below. Yusairi (Ex.1005) at 56, Sec. 7.1.1, FIG. 6.

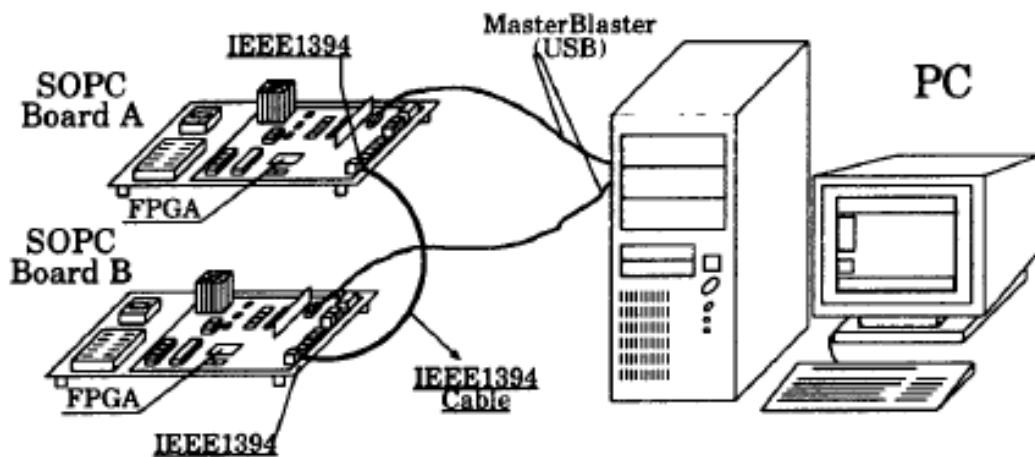


Fig. 6 System used for experiment (1).

140. “The application module on the SOPC board A generates data (0x610A) and SOPC board B’s IP address, and passes it to fire IP module where the data is encapsulated and transmitted to the SOPC board B.” Yusairi at 56, Sec. 7.1.2.

[1(a)] a first endpoint;

141. Yusairi discloses this limitation.

142. Figure 6 below is annotated to show SOPC Board A, which acts as a *first endpoint* because “SOPC board A generates data . . . [that is] transmitted to the SOPC board B.” Ex.1005 at 56, Sec. 7.1.2.

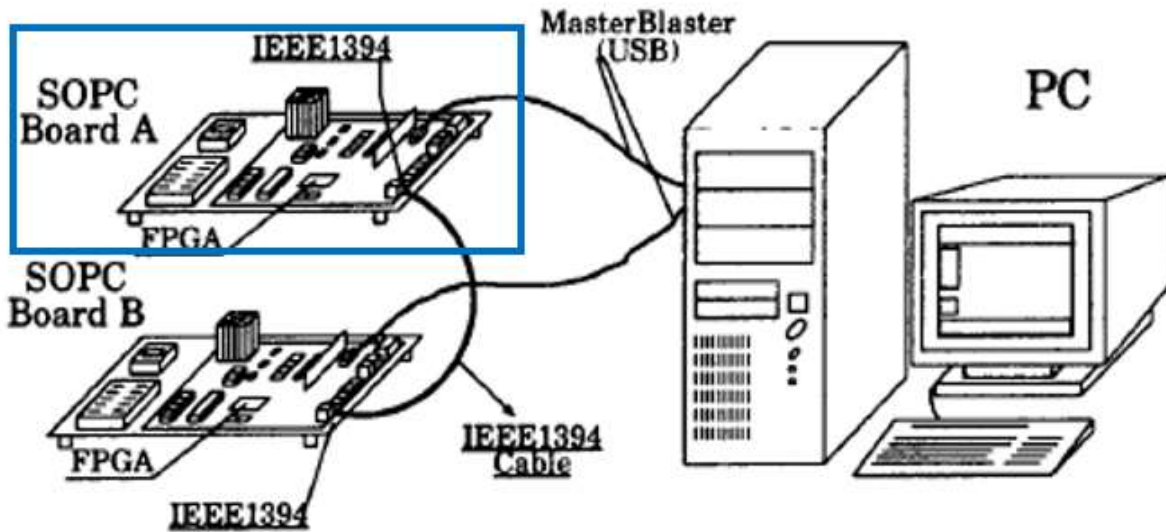


Fig. 6 System used for experiment (1).

[1(b)] a second endpoint, including physical and link layer circuitry and higher level circuitry coupled to the physical and link layer circuitry; and

143. Yusairi discloses this limitation.

144. Figure 6 below is annotated to show SOPC Board B, which acts as a *second endpoint* because “The SOPC board B waits for the incoming packet” and “[r]eceive[s] packets” from SOPC Board A. Yusairi at 56, Sec. 7.1.2.

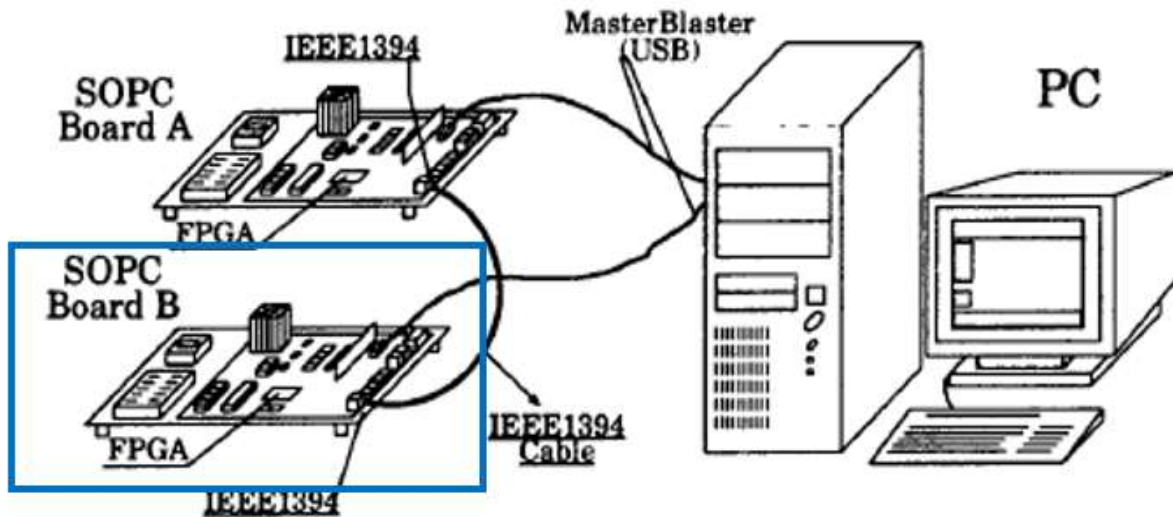


Fig. 6 System used for experiment (1).

145. The *second endpoint*, SOPC board B, includes *physical and link layer circuitry* because it includes a “fire IP” module having an “LLC-PHY interface.” Yusairi at 54, Sec. 5.1; *see also* Yusairi at 53, Sec. 3.2 (“The IEEE134 interface consists of link and physical layers which are usually called LLC and PHY, respectively.”).

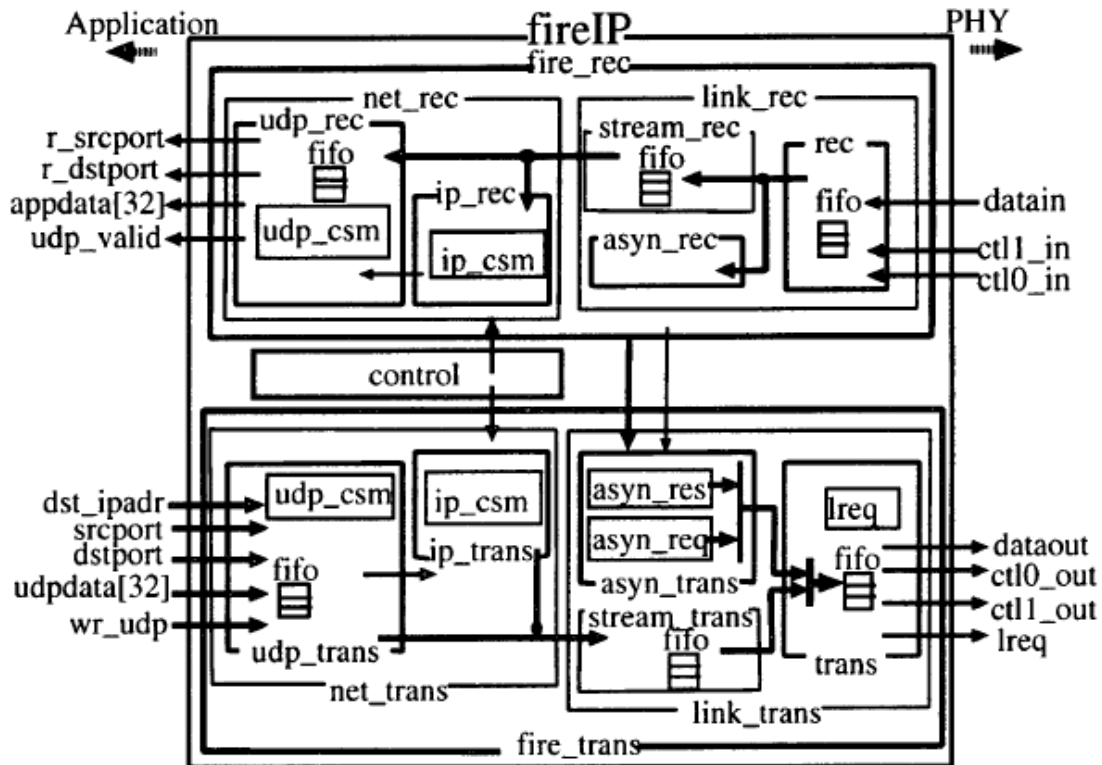


Fig. 5 Block diagram of fireIP module.

146. Further, the FireIP module includes circuitry corresponding to modules controlling the link layer and physical layer interface (respectively, *link layer circuitry* and *physical layer circuitry*), including “fifo,” “crc,” “lreq,” “trans,” and “rec.” Yusairi at 54, Sec. 5.1.

147. The FireIP module also shows connected *higher level circuitry*. The FireIP modules above include circuitry that correspond to functions at the network layer (*higher level circuitry*), which “consist of two module[] **ip_trans** and **ip_rec** for transmitting and receiving.” *Id.* at 54, Sec. 5.3. This is further illustrated by Figure 4 of Yusairi, copied below, which illustrates the various layers of a transmitter (a first endpoint, such as SOPC Board A) and receiver (a second

endpoint, such as SOPC Board B), as understood by the arrow from the IEEE1394 chip to the left to the IEEE1394 chip on the right.. Yusairi at 56, Sec. 4.1.1, Fig. 4.

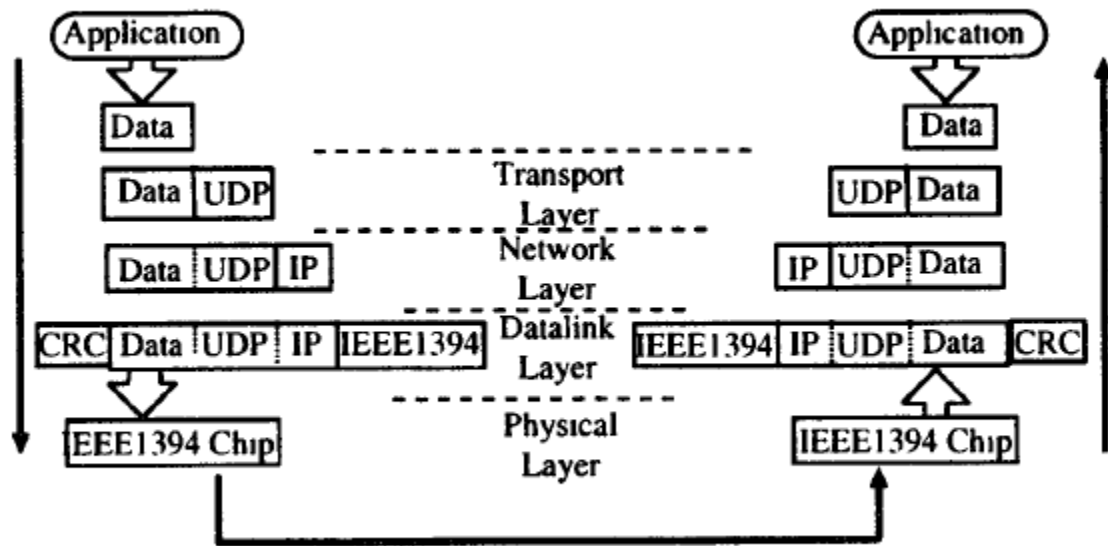


Fig. 4 UDP/IP protocol suite.

Looking at Figure 4, a POSITA would have understood that the physical layer and datalink layer (the link layer) are coupled the higher level network and transport layers on the right side illustrating a receiver since data is transmitted upward from the physical and datalink (link) layers to the upper transport and network layers.

148. A POSITA would have understood that the link layer and physical layer of Yusairi are coupled to the higher-level network and transport layers because the link layer within the PHY-LLC interface can communicate packets to the higher-level network and layer, thus disclosing *higher level circuitry coupled to the physical and link layer circuitry*. Yusairi at 55, Sec 5.2 (At the link layer, “The header check process checks header and data CRCs and the type field value. If both of the CRCs

are correct and the type field equals to 0x0800 (IPv4) the packet is accepted and handed to the **ip_rec**” module of the network layer) (emphasis in original).

[1(c)] a serial link between the first endpoint and the second endpoint,

149. Yusairi discloses this limitation.

150. An IEEE1394 cable (*the serial link*) is used to connect the SOPC Board A (*the first endpoint*) and the SOPC Board B (*the second endpoint*) as annotated below in Figure 6 from Yusairi. Ex.1005, at 56, Sec. 7.1.1, Fig. 6. A POSITA would have understood the IEEE1394 cable to be a *serial link* because since it transmits data from SOPC Board A to SOPC Board B. See IEEE 1394 (Ex.1008) at 46 (describing an IEEE 1394 bus that transmits data between nodes).

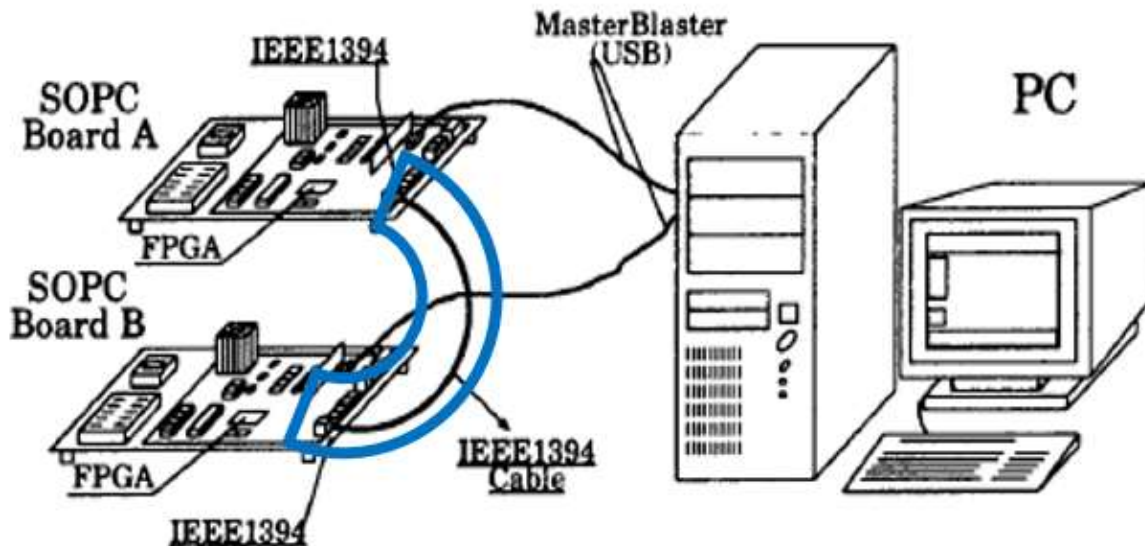


Fig. 6 System used for experiment (1).

[1(d)] wherein the first endpoint is configured to generate encoded data in accordance with a line code, and to transmit the encoded data over the link to the second endpoint, and

151. Yusairi discloses this limitation.

152. Yusairi discloses that at SOPC board A (*the first endpoint*), packets are “encapsulated and transmitted to the SOPC board B” (*generate encoded data . . . and to transmit the encoded data over the link to the second endpoint*). Ex.1005, at 56, Sec. 7.1.2. At SOPC board B, “[r]eceived packets [are] decapsulated.” *Id.* A POSITA would have understood that Yusairi is encoding data in accordance with line code by encapsulating it at SOPC board A. Generally, “‘encapsulation’ is the computer-networking process of concatenating layer-specific headers or trailers with a service data unit (i.e. a payload) for transmitting information over computer networks.” Ex.1023 at 1. By encapsulating the higher level IP data with the packet header, Yusairi is encoding data in accordance with line code, since the encapsulated packet having the header and IP data will be transmitted across a link (such as the IEEE1394 cable). Yusairi at 54, Sec. 5.2 (“generates a link header, attaches it to IP data”), Yusairi, at 53, Sec. 4 (“The protocol stack we are going to implement is shown in Fig. 4 together with data encapsulation and decapsulation”). As noted previously with regard to the discussion at limitation [1(d)], line code is the information transmitted across the line. Thus, a POSITA would have understood that

encapsulating the packet for transmission across the IEEE1394 cable generates encoded data in accordance with line code.

153. Further, Yusairi also discloses using IEEE1394, which a POSITA would have understood encodes data *in accordance with a line code*. Yusairi, at 56, Sec. 7.1.1 (“Each of the boards [is] equip[ed] with the FPGA and an IEEE1394 PHY chip . . .”). IEEE1394 standards describe a high performance serial bus. IEEE1394 (Ex.1008) at 1. The IEEE1394 uses an “NRZ” (non-return to zero) signaling technique, which transmits encoded NRZ data, which is encoded line code. IEEE 1394 at 15, 81 (“[P]hysical layer entities on the bus communicate via NRZ data. The NRZ data is transmitted and received as Data_Tx and Data_Rx and is accompanied by a strobe signal, Strb_Tx and Strb_Rx. This strobe signal changes state whenever two consecutive NRZ data bits are the same, ensuring that a transition occurs on either Data or Strb.”). A POSITA would have understood an NRZ signaling technique to be line code. Ex.1024 at 1 (“In telecommunications, a non-return-to-zero (NRZ) line code is a binary code in which ones are represented by one significant condition, usually a positive voltage, while zeros are represented by some other significant condition, usually a negative voltage, with no other neutral or rest condition.”). The NRZ line code is illustrated by the signal changes shown in Figure 4.17 of the IEEE1394 standard below.

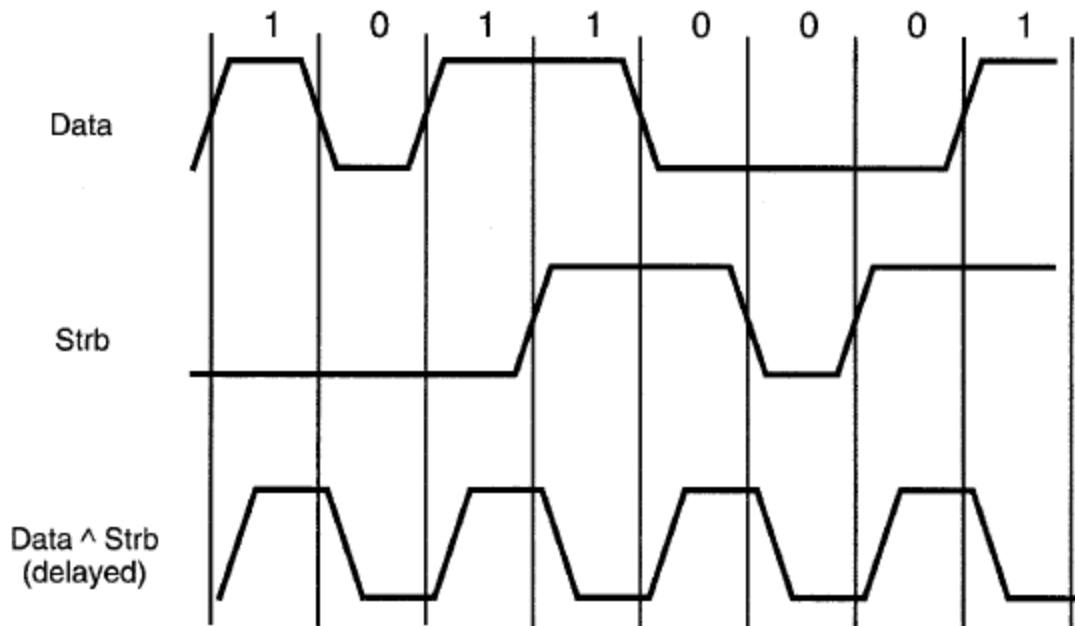


Figure 4.17—Data-strobe coding

IEEE1394 at 82. The encoding method is further described in Section 4.4.1.1, which states, “Data transmission is a straightforward function: the data bits are sent to the attached peer PHY along with the appropriately encoded strobe signal using the timing provided by the PHY transmit clock.” IEEE1394 at 95, Sec. 4.4.1.1, 82 (“The procedure for encoding data during transmission is described in 4.4.1.1 and decoding during reception in 4.4.1.2.”). A POSITA would have understood that Yusairi discloses encoding data in accordance with line code at a transmitter (such as SOPC Board A, the first endpoint) because the IEEE1394 standard on which Yusairi was designed uses NRZ line code to transmit data.

[1(e)] wherein the physical and link layer circuitry of the second endpoint is coupled to the link and configured to decode the encoded data

154. Yusairi discloses this limitation, and Yusairi in view of Shin renders obvious this limitation.

155. As illustrated in Figure 6, Yusairi's SOPC Board B (*the second endpoint*) is coupled to the IEEE1394 cable (*the link*) over which it receives the packets from SOPC Board A. Ex.1005, at 56, Sec. 7.1.1, Fig. 6.

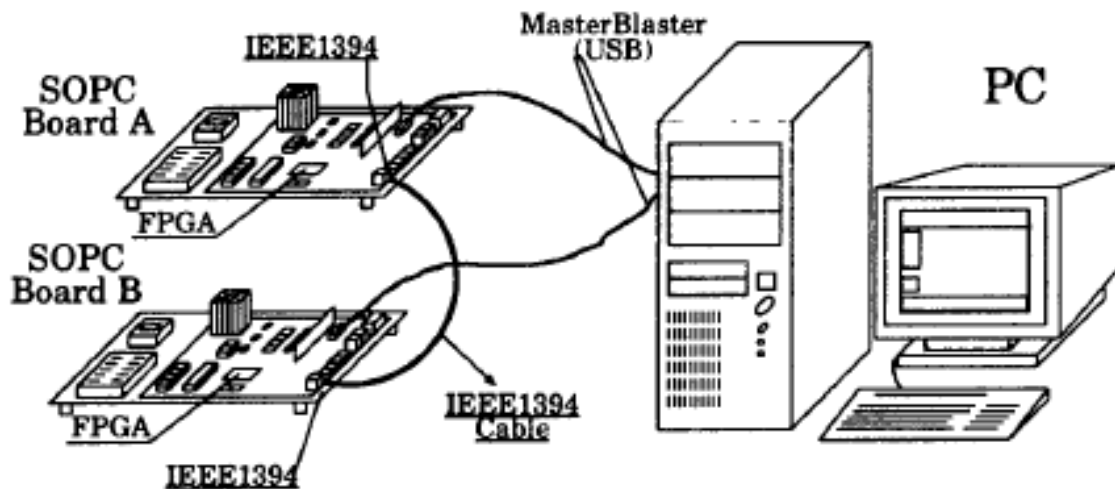


Fig. 6 System used for experiment (1).

156. Further, as illustrated in Figure 4, the physical layer of the LLC-PHY interface at SOPC Board B receives the packets. *Id.* at 53, Fig. 4. A POSITA would have understood that the right side of Figure 4 illustrates layers of a receiver (such as SOPC Board B, the second endpoint), since the illustration shows an arrow

representing the data transmission from the physical layer of the left side transmitter to the physical layer of the right side receiver.

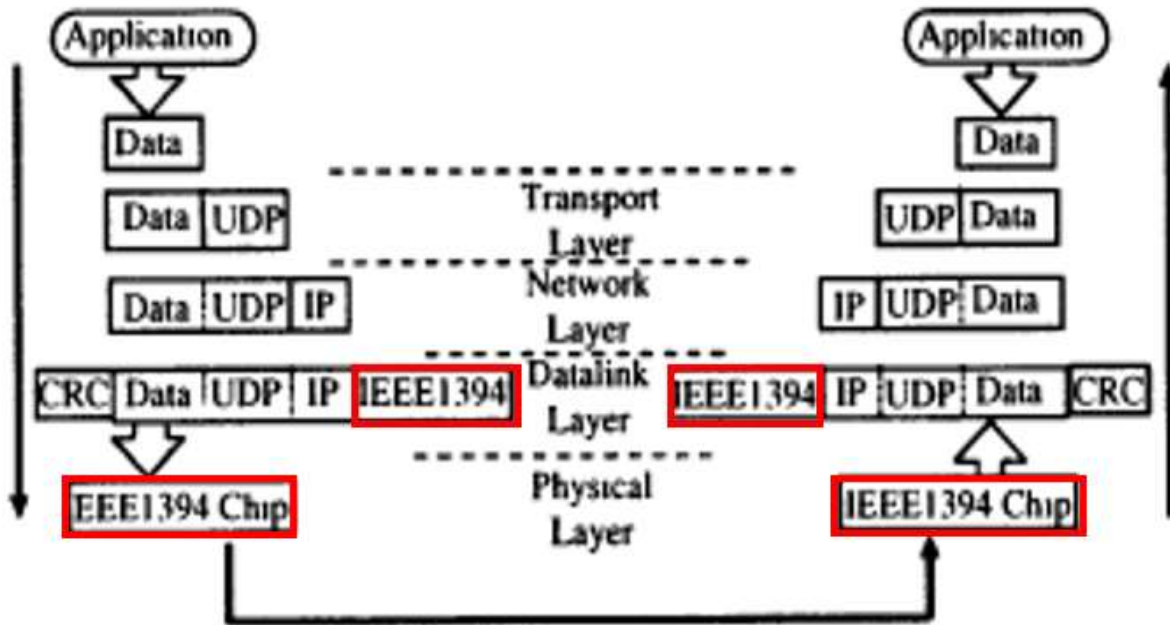


Fig. 4 UDP/IP protocol suite.

A POSITA would have understood that the physical and link layer circuitry of the SOPC Board B (*the second endpoint*) is coupled to the IEEE1394 cable (*the link*) to receive the packets from SOPC Board A at least because Figure 4 shows the transmission of the encoded data to the IEEE1394 physical layer chip on the right side receiver. To receive the data at the physical layer of the receiver (such as SOPC Board B, the second endpoint) the physical layer is coupled to the link transmitting the data (such as the IEEE1394 cable). A POSITA would have also understood that the physical link and datalink (link) layer circuitry are couple to the link (the

IEEE1394 cable) because, as illustrated and annotated in Figure 4, IEEE1394 circuitry is used across the physical layer and the datalink (link) layers.

157. Yusairi further discloses that the physical and link layer circuitry of the second *is configured to decode* the encoded data packets once received. At SOPC Board A, packets are “encapsulated and transmitted to the SOPC board B.” Yusairi at 56, Sec. 7.1.2. At SOPC Board B, “[r]eceived packets [are] decapsulated.” Yusairi at 56, Sec. 7.1.2. A POSITA would have understood that Yusairi is encoding by encapsulating it at SOPC Board A, as described above with respect to limitation [1(d)], and decoding the encoded data by decapsulating it at SOPC Board B (*the second endpoint*). A POSITA would have understood that decapsulation at the receiving SOPC Board B is the reverse of the encapsulation at SOPB Board A, where concatenated headers are removed from the IP data, thereby decoding the received packets. Ex.1023 at 1 (“Deencapsulation (or de-encapsulation) is the reverse computer-networking process for receiving information; it removes from the protocol data unit (PDU) a previously concatenated header or trailer that an underlying communications layer transmitted.”).

158. Further, A POSITA would have understood that Yusairi’s IEEE1394 chip at SOPC Board B decodes the encoded data, and this would be done using the circuitry of the physical and link layers since, as illustrated and annotated above at Figure 4, the IEEE1394 chip provides the physical and link layer circuitry.

IEEE1394 at 96, Sec. 4.4.1.2 (“Data reception for the cable environment physical layer has three major functions” including “decoding.”); Yusairi at Fig. 4 (annotated with **red** to show IEEE1394 circuitry at the datalink (link) layer and physical layer.). Figure. POSITA would have understood that, because Yusairi uses the IEEE1394 PHY chip for the physical layer circuitry of the SOPC Board B, the encoded data received by the SOPC Board B would be decoded at the physical layer. For instance, Figure 5.7 of the IEEE1394 standard illustrates a “data resynch” (annotated in **green**) component in the physical layer.

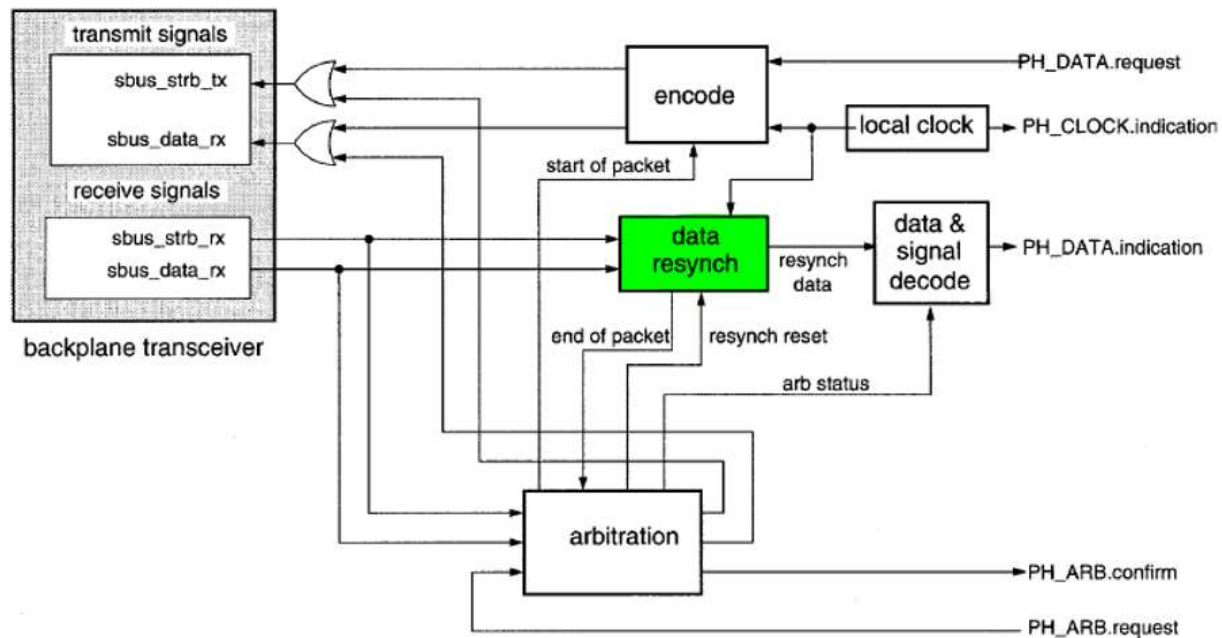


Figure 5.7—Backplane PHY architecture

IEEE1394 at 132, FIG 5.7 (annotated). “The “data resynch” block decodes the data-strobe signal and retimes the received data to a local fixed-frequency clock provided by the ‘local clock’ block.” Because the data resynch component of the physical

layer “decodes” data, a POSITA would have understood that the physical and link layers of the receiver (such as SOPC Board B, the second endpoint) are configured to decode the received encoded data.

159. Further, a POSITA would have also understood that the encoded data is decoded by *the physical and link layer circuitry of SOPC Board B (the second endpoint)* because the “PHY [physical layer] hands a received packet to the module rec” and the “rec module then puts the packet into a fifo.” Yusairi at 54, Sec. 5.1. At the link layer, “stream_rec takes the data from the fifo and checks the link header” to “start the header check process.” Yusairi at 54, Sec. 5.2. As such, a POSITA would have understood that the encoded data is being decoded by the *physical and link layer circuitry* since it is being received by the physical layer and used at least by the link layer. The physical and link layer circuitry would decode the encoded data from the line (the IEEE1394 cable) by retrieving it at the physical layer and using it at the link layer.

[1(f)] to recover application data and control bits,

160. Yusairi discloses this limitation.

161. At SOPC Board A (the first endpoint), the stream_trans module “generates a link header, [and] attaches it to IP data.” Yusairi at 54, Sec. 5.2. Within the link header, there is “a transaction code” and “a type field.” Yusairi at 54-55,

Sec. 5.2. The link header and IP data packet “is transmitted to the SOPC Board B [the second endpoint]” Yusairi at 56, Sec. 7.1.2.

162. A POSITA would have understood the “transaction code” and the “type field” each include *control bits* because each causes performance of a function at one of the layers, such as the link layer and the network layer. ’307 Patent at 7:55-60 (describing control bits as “providing transport layer functionality”); ’307 Patent at 7:60-65 (describing control bits as providing link layer functionality). A POSITA would have understood that the transaction code comprises control bits because “If the transaction code is equal to 0xA . . . stream_rec will start the header check process.” Yusairi at 54, Sec. 5.2. Additionally, a POSITA would have also understood that the type field comprises different control bits because if the “type field equals 0x0800 (IPv4) the packet is accepted and handed to the ip_rec [of the network layer]” where ip_rec performs higher network level functions, including “check[ing] the destination address,” “accept[ing] the packet and calculate[ing] the IP checksum.” Yusairi at 55, Sec. 5.2-5.3.

163. As noted above at limitation [1(e)], the physical and link layers of SOPC Board B decode the received packet, which would “recover” the control bits. A POSITA would understand that the *control bits* of the transaction code and type field are “recovered” because the link layer uses them to determine the value of the transaction code “to start the header check process” and the value of the type field

to assert a “ready signal” to the ip_rec module of the network layer. Yusairi at 55, Sec. 5.2-5.3. A POSITA would have understood that the control bits of the transaction code are recovered because the value of the control bits is used to determine whether to start a header check process. Yusairi at 54, Sec. 5.2. Likewise, a POSITA would have understood the control bits of the type field are recovered because the value of these control bits is used to determine whether to assert a ready signal to the network layer. Yusairi at 55, Sec. 5.2-5.3.

164. Further, Yusairi also discloses recovering *application data*. As noted, the header is attached to “IP data.” Yusairi at 54, Sec. 5.2. A POSITA would have understood that Yusairi’s IP data is application data because the IP data is attached to the link header. Yusairi at 54, Sec. 5.2; IEEE1394 at 147 (“An asynchronous packet shall consist of a packet header, and it may also include a data block.”). A POSITA would have understood that the IP data (*the application data*) is recovered because the IP data is included in the packet, and the packet is decoded so that it can be “accepted and handed to the ip_rec [module of the network layer].” Yusairi at 55, Sec. 5.2.

[1(g)] to perform at least one link level function in response to at least one of the control bits, and

165. Yusairi discloses this limitation.

166. For the reasons discussed with respect limitation [1(f)], Yusairi discloses “a transaction code” and “a field type” of a packet header that each

comprise *control bits*. As also noted above at [1(f)], a POSITA would have understood the transaction code and field type are control bits because they cause a function to be performed when received. In response to at least one of the control bits of the transaction code, “stream_rec [a link level module] will start the header check process [a link level function]. Yusairi, at 54, Sec 5.2. A POSITA would have understood that the header check process (*the link level function*) occurs in response to at least one of the control bits because the header check process (*link level function*) is started when “the transaction code is equal to 0xA.” Yusairi at 54, Sec 5.2.

[1(h)] to assert at least one control signal indicative of at least another one of the control bits to the higher level circuitry, and

167. Yusairi discloses this limitation.

168. As discussed in above with respect to limitation [1(g)], Yusairi discloses that a link level function (the header check process) is performed in response to at least one of the control bits from the transaction code. Yusairi, at 54, Sec 5.2.

169. Additionally, Yusairi further discloses *to assert at least one control signal indicative of at least another one of the control bits to the higher level circuitry*. Yusairi discloses that, at the link layer, “the packet is accepted and handed to the ip_rec [module of the network layer]” when “the type field [is] 0x0800.” Yusairi at 55, Sec. 5.2-5.3. Accordingly, the ip_rec “module [of the network layer]

reads its fifo when it detects the ready signal [*the control signal*] asserted by stream_rec [a link layer module].” Yusairi at 55, Sec. 5.3. Thus, A POSITA would have understood that Yusairi asserts *at least one control signal* (the ready signal from stream_rec) *to the higher level circuitry* (the network layer).

170. Further, the ready signal (*the control signal*) is *indicative of* the bits of the type field (*control bits*) at least because it occurs in response to “the type field equal[] to 0x800,” thus indicating the presence and value of the type field control bits *Id.* at 55, Sec. 5.2.

171. Additionally, a POSITA would have understood that the bit(s) of the transaction code are different from the type field bit(s) because the type field value is determined during the header check process that starts in response to the transaction code bit value. Yusairi at 54-55, Sec. 5.2 (“If the transaction code is equal to 0xA of the GASP type, stream_rec will start the header check process,” and “[t]he header check process checks the header and data CRC’s and the type field value.”). Thus, the transaction code includes at least one of the control bits, while separately, the type field includes *at least another one of the control bits*.

[1(i)] wherein the higher level circuitry is configured to perform at least one higher level function in response to the at least one control signal.

172. Yusairi discloses and renders obvious this limitation.

173. As discussed with respect to limitation [1(h)], Yusairi's network layer includes *higher level circuitry*, and it receives a ready signal (*the control signal*) asserted from the link layer.

174. Additionally, Yusairi discloses that the network layer (*higher level circuitry*) is configured to perform several functions, including "reading fifo," "check[] the destination address," and "calculate[] the IP checksum." Ex.1005 at 55, Sec 5.3. A POSITA would recognize these are each a *higher level function* since they are performed by the network layer (*the higher level circuitry*). These network-layer functions are performed *in response to the at least one control signal* because they occur "when [ip_rec] detects the ready signal." Ex.1005 at 55, Sec 5.3.

4. Claim 2

[2] The system of claim 1, wherein the line code specifies a block code and special characters, the first endpoint is configured to encode M-bit input words in accordance with the block code to generate a code word sequence of N-bit code words, where $N > M$, the code word sequence is indicative of at least one cell of the input words, and the encoded data are a sequence of at least one of the special characters and the code words of the code word sequence.

175. Yusairi in view of Shin renders obvious this limitation.

176. Shin discloses this limitation for the reasons discussed with respect to limitation [2] of Ground 1 and 2, which discusses Shin's use of block code (such as 8B/9B), and transmitting block encoded in-band symbols and out-of-band special characters via the line code. A POSITA would have understood that applying the use of Shin's block code to encode Yusairi's packets (having the application data

and control bits) would generate encoded in-band symbols, which is the block code from the packets that is included with the special characters, like IDLE primitives, when transmitted in line code. Shin at [0086] (“A packet that is transmitted on a serial communications link typically comprises in-band symbols.”); Shin at [0131] (discussing 8B/9B encoding generates 9-bit symbols from 8-bit codes); Shin at [0086] (further discussing that out-of-band symbols, like IDLE characters, can be inserted into in-band symbols and transmitted between communications links).

177. A POSITA would have been motivated to modify Yusairi with Shin’s block encoding methods because Shin’s block encoding methods, having M-bit input words and N-bit code words, where $N > M$, would have provided Yusairi with known communication protocols that provide the benefit of having a neutral DC component to help ensure signal is accurately received and decoded by synchronizing the SOPC Board B receiver with the SOPC Board A transmitter. For example, a POSITA would have been motivated to use the block encoding method disclosed by Shin because it was known in prior art systems that block encoding methods with $N > M$, such as 8B/10B encoding methods, can provide DC balance. HDMI 1.0 (Ex.1010) at 55 (“During video data, where each 10-bit character represents 8 bits of pixel data, the encoded characters provide an approximate DC balance as well as a reduction in the number of transitions in the data stream.”). Further, a POSITA would have been motivated to modify Yusairi with Shin’s

encoding methods, because doing so would provide for flow control at the lower physical and link layers, as opposed to relying on higher level application functions to provide flow control. Yusairi at 53, Se. 4.1 (“For more sophisticated reliabilities or flow control necessary for each application, we rely on functions to be supplied by the application itself.”); SATA 1.0 at 129 (describing how the known prior art system SATA includes a link layer that employs 8B/10B encoding and also provides frame flow control.).

5. Claim 6

[6] The system of claim 2, wherein said at least one of the special characters is an idle character.

178. Yusairi in view of Shin renders obvious this limitation for the reasons discussed with respect to limitation [2].

179. A POSITA would have been motivated to modify Yusairi with Shin’s encoding methods to render this limitation obvious for the reasons discussed with respect to limitation [2]. Additionally, Shin disclose that IDLE primitives (*idle characters*) would help synchronize transmitters and receivers, and as such, a POSITA would have been further motivated to use Shin’s IDLE primitives (*idle characters*) with Yusairi’s system for synchronization between SOPC Board A and SOPC Board B. Shin at [0096] (“An IDLE primitive may be a synchronization primitive.”); Shin at [0096] (“When a communications node receives a

synchronization primitive, it knows that the primitive is correctly aligned on a symbol boundary.”).

6. Claim 7

[7] The system of claim 6, wherein the physical and link layer circuitry is configured to perform simplex link flow control by inserting idle characters between subsequences of the code word sequence, where each of the subsequences is indicative of a different cell of the input words.

180. Yusairi in view of Shin renders obvious this limitation. Shin discloses this limitation as discussed above with respect to the discussion at limitation [7] of Grounds 1 and 2. Further, a POSITA would have been motivated to modify Yusairi with a communication architecture that uses simplex link flow control for the reasons also stated at limitations [2] and [6], e.g., to provide Yusairi with lower physical and link layer flow control capabilities rather than relying on higher application layers for flow control.

7. Claim 23

[23(pre)] A communication system, including:

181. Yusairi discloses this limitation for the reasons set forth in [1(pre)].

[23(a)] a first endpoint;

182. Yusairi discloses this limitation for the reasons set forth in [1(a)].

[23(b)] a second endpoint; and

183. Yusairi discloses this limitation for the reasons set forth in [1(b)].

[23(c)] a serial link between the first endpoint and the second endpoint,

184. Yusairi discloses this limitation for the reasons set forth in [1(c)].

[23(d)] wherein the first endpoint is configured to encode M-bit input words in accordance with a block code to generate a code word sequence of N-bit code words, where $N > M$,

185. Yusairi in view of Shin renders obvious this limitation for the reasons set forth in [2].

186. A POSITA would have been motivated to modify Yusairi with Shin's encoding methods to teach this limitation for the reasons describe with respect to limitation [2].

[23(e)] the code word sequence is indicative of at least one cell of the input words, the input words of each said cell are indicative of application data and control bits,

187. Yusairi in view of Shin renders obvious this limitation.

188. First, Shin discloses *the code word sequence is indicative of at least one cell of the input words* for the reasons set forth for limitation [2] of Grounds 1 and 2.

189. Further, Yusairi in view of Shin renders obvious the input words of each said cell are indicative of application data and control bits. As also discussed at [1(f)], Yusairi discloses generating a packet having application data (IP data) and control bits (transaction code bits and field type bits) at SOPC Board B. Yusairi at 54, Sec. 5.2. Shin discloses that a packet (*cell*) can be formed into codes (*input words*) that are encoded by block encoding into symbols (*code words*), as discussed with respect to limitation [2] of Grounds 1 and 2. Accordingly, as would have been understood by a POSITA, when Shin's block encoding method is applied to Yusairi's packet, the IP data (*application data*) and transaction code bits and field

type bits (*control bits*) of the packet (*cell*) would have been used to form the codes (*input words*), and thus, *the input words of each said cell are indicative of application data and control bits.*

190. A POSITA would have been motivated to combine Shin’s teaching of using a code word sequence [] indicative of at least one cell of the input words with Yusairi’s teaching of the input words of [a] cell [being] indicative of application data and control bits because doing so would have allowed Yusairi’s packet having application data and control bits to be encoded and transmitted as part of a sequence of symbols (code word sequence) using known block encoding methods, such as Shin’s 8B/9B block encoding Shin at [0131]. A POSITA would have been further motivated to encode Yusair’s packets with Shin’s block encoding methods for the reasons stated at limitations [2] and [6].

[23(f)] the control bits have multiple levels of communication protocol functionality, and

191. Yusairi discloses this limitation.

192. As discussed at limitation [1(f)], Yusairi discloses a transaction code and a field type having *control bits*. As noted in [1(g)], the transaction code control bits are functional at the link level because, “If the transaction code is equal to 0xA [the link layer] will start the header check process [*a communication protocol functionality at the link layer*].” Further, as described in [1(i)], the transport layer performs any one of several functions—“checks the destination address,” “accept[s]

the packet and calculates the checksum”—responsive to a ready signal asserted because “the type field [is] equal[] to 0x0800.” *See also*, Yusairi at 55, Sec. 5.2, 5.3. A POSITA would have understood that because there is a function performed at the link layer and also a function performed at the higher transport layer based on values of the transaction code and the type field *control bits*, the *control bits have multiple levels of communication protocol functionality*.

[23(g)] the first endpoint is configured to transmit encoded data including the code words of the code word sequence over the link to the second endpoint.

193. Yusairi in view of Shin renders obvious this limitation.

194. As discussed in [1(d)], Yusairi discloses that the first endpoint is configured to transmit encoded data over the link to the second endpoint.

195. Further, Shin discloses that *encoded data includes the code words of the code word sequence* for the reasons discussed in [2] of Grounds 1 and 2. A POSITA would have been motivated to modify Yusairi with Shin’s encoding methods to teach transmitting encoded data that includes code words (symbols) of a code words sequence (sequence of symbols) for the reasons discussed with limitations [2] and [6].

8. Claim 25

[25] The system of claim 23, wherein the block code is specified by a line code that also specifies special characters, the encoded data are a sequence of at least one of the special characters and the code words of the code word sequence, and none of the special characters of the encoded data matches a bit sequence of the code word sequence.

196. Yusairi in view of Shin renders obvious this limitation.

197. As discussed with respect to limitation [2], Shin discloses wherein the block code is specified by a line code that also specifies special characters, the encoded data are a sequence of at least one of the special characters and the code words of the code word sequence.

198. Further Shin discloses or renders obvious that *none of the special characters of the encoded data matches a bit sequence of the code word sequence*. Shin's transmitted packets comprise in-band symbols (*the code word sequence*), and the in-band symbols include bits (*bit sequence of the code word sequence*). *Id.* at [0086]. Shin also teaches that control characters (*the special characters of the encoded data*), such as the synchronization primitive, "have a unique number of transitions that is not found in any . . . in-band symbol [*code word sequence*]." *Id.* at [0101]. A POSITA would understand that Shin's control characters with transitions that are not found in bits of any of the in-band symbols teaches that *none of the special characters of the encoded data matches a bit sequence of the code word sequence*. This would have been further understood by a POSITA because prior

serialization and encoding methods, such as SATA 1.0, used control primitives (*special characters* inserted to convey state information or control information transfer) that did not match bit sequences of code words (the encoded data). SATA 1.0 at 142-43 (“A primitive consists of a single Dword and is the simplest unit of information that may be exchanged between a host and a device. When the bytes of a primitive are encoded the resulting pattern is difficult to misinterpret as any other primitive or random pattern. Primitives are used primarily to convey real-time state information, to control the transfer of information and coordinate host/device communication. All bytes in a primitive are constants and the first byte is always a special character. Since all of the bytes are constants, a primitive cannot be used to convey variable information.”).

199. A POSITA would have been motivated to modify Yusairi with Shin’s block encoding methods, which includes out-of-band symbol that do not normally appear within a packet, to teach this limitation for the reasons discussed with limitations [2] and [6].

9. Claim 31

[31] The system of claim 23, wherein each of the first endpoint and the second endpoint is a transceiver, and the second endpoint is configured to generate additional encoded data in response to input words in accordance with the line code and to transmit the encoded data over the link to the first endpoint.

200. Yusairi discloses this limitation and Yusairi in view of Shin renders obvious this limitation.

201. Each of SOPC Board A (*the first endpoint*) and the SOPC Board B (*the second endpoint*) has the FireIP component that “consists of fire_trans and fire_rec for transmit and receive operations, respectively” Yusairi, at 53, Sec. 5; *see also* Yusairi at 55, Sec. 7.1 (“The purpose of this experiment is to verify the proper communications between two fireIP modules.”). A POSITA would have understood that a transceiver is capable of acting as both a transmitter and a receiver for packet transmission and reception, respectively. IEEE1394 (Ex.1008) at 322 (“The transceivers and the bus to which they are connected make up the electrical/mechanical interface of the Serial Bus. Other functions of the PHY layer (e.g., arbitration and packet transmission/reception) are performed within the backplane PHY logic.”). Because each of the SOPC Board A and the SOPC Board B can perform “transmit and receive operations,” A POSITA would have recognized Yusairi discloses that each of the endpoints is *a transceiver*. Yusairi at 53, Sec. 5. A POSITA would have further understood that Yusairi’s SOPC Board A and SOPC Board B (*the endpoints*) are transceivers because they include the IEEE1394 chip, and the IEEE1394 protocol operated on transceiver technology. IEEE1394 at iii (“There are two physical environments for the Serial Bus: backplane and cable. The backplane environment uses two single-ended signals on a broadcast multitapped bus using backplane transceiver logic (BTL) or emitter coupled logic (ECL)

transceiver technology at 49.152 Mbit/s or enhanced transistor-transistor logic (TTL) transceiver technology at 24.576 Mbit/s.”).

202. Shin discloses the second endpoint is configured to generate additional encoded data in response to input words in accordance with the line code and to transmit the encoded data over the link to the first endpoint. Shin describes a communication node 3801 (the first endpoint) and another communication node 3811 (the second endpoint) that are each a transceiver because both have a transmitter component and a receiver component and can communicate using a bidirectional serial link. Shin at [0164]. This is illustrated by FIG. 38A below, where the transmitter and receiver of each are indicated by annotation.

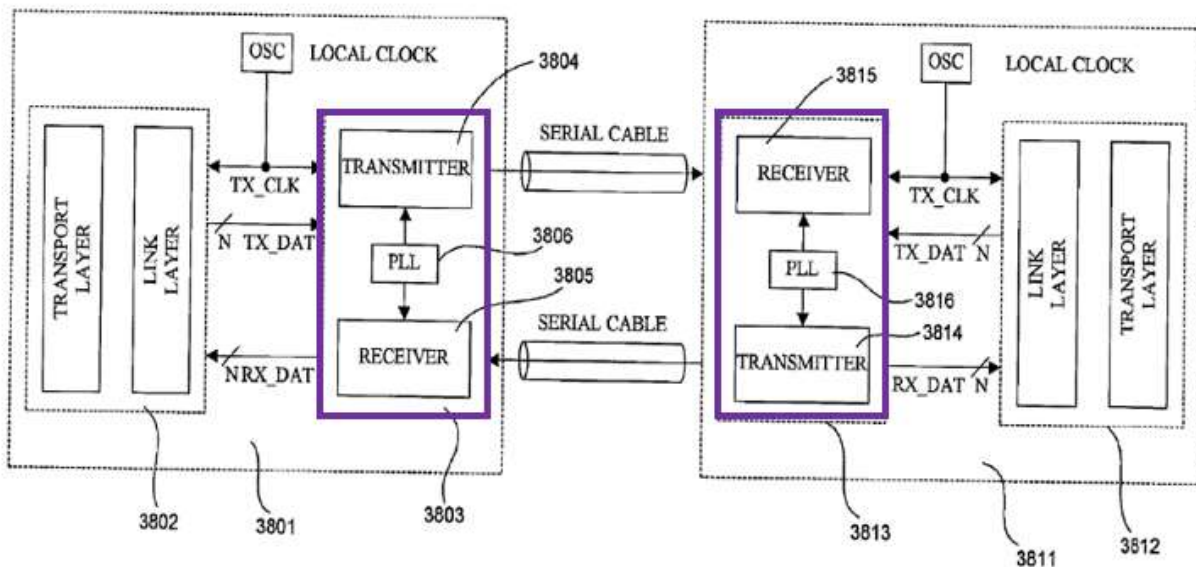


Fig. 38A

203. As illustrated in 38A, encoded data can be transmitted over a serial cable (*the link*) from communication node 3811 (*the second endpoint*) to

communication node 3801 (*the first endpoint*). Shin at Fig. 38A. Further, Shin discloses that nodes can transmit N-bit block *encoded in response* to M-bit *input words*, such as 8B/9B block encoding. Shin at [0128] (“In one embodiment, the link layer of a communications node encodes codes into symbols in order to optimize transition control and to ensure DC-balance.”); Shin at [0131] (“The alternate bit inversion encoding technique is referred to as an 8b/9b encoding”); Shin at [0165]. As discussed at [1(d)], A POSITA would understand that Shin’s 8B/9B block code is *generated in accordance with line code*. Thus, because communication node 3811 (*the second endpoint*) is a communication node that can encode codes (*input words*) to symbols (*code words*) that are transmitted by a line code to communication node 3801 (*the first endpoint*), A POSITA would have understood that Shin discloses this limitation.

204. A POSITA would have been motivated to combine Shin’s teaching of generating additional encoded data in response to input words in accordance with the line code at the second endpoint and transmitting it to the first endpoint because doing so would provide full-duplex communication, which was standard for IEEE1394, which Yusair uses for the link and physical layer circuitry and as the IEEE1394 cable communications link. IEEE1394 at 15 (defining a “physical connection” as “[t]he full-duplex physical layer association between directly connected nodes. In the case of the cable physical layer, this is a pair of physical

links running in opposite directions”; IEEE1394 at 34 (“The cable environment is a network of nodes connected by point-to-point links called physical connections. The physical connection consists of a port on the PHY of each node and the cable between them.”).

10. Claim 53

[53(pre)] A receiver for receiving encoded data that have propagated over a serial link, the encoded data having been generated in accordance with a line code, said receiver including:

205. To the extent the preamble is limiting, Yusairi in view of Shin renders obvious this limitation for the reasons set forth in [1(b)] and [1(d)].

[53(a)] physical and link layer circuitry, coupled and configured to receive and decode the encoded data

206. Yusairi discloses this limitation and renders this limitation obvious in view of Shin for the reasons set forth in [1(e)].

[53(b)] to recover application data and control bits

207. Yusairi discloses this limitation for the reasons set forth in [1(f)].

[53(c)] to perform at least one link level function in response to at least one of the control bits; and

208. Yusairi discloses this limitation for the reasons set forth in [1(g)].

[53(d)] higher level circuitry coupled to the physical and link layer circuitry, wherein the physical and link layer circuitry is coupled and configured to assert at least one control signal indicative of at least another one of the control bits to the higher level circuitry,

209. Yusairi discloses this limitation for the reasons set forth in [1(b)] and [1(h)].

[53(e)] and the higher level circuitry is configured to perform at least one higher level function in response to the at least one control signal.

210. Yusairi discloses this limitation for the reasons set forth in [1(i)].

11. Claim 68

[68(pre)] A transmitter including physical and link layer circuitry, wherein the physical and link layer circuitry include:

211. To the extent the preamble is limiting, Yusairi discloses this limitation.

212. Yusairi discloses this limitation. Yusairi's SOPC Board A is *a transmitter* because it "generates data ... [that is] transmitted to the SOPC board B." Ex.1005 at 56, Sec. 7.1.2. Further, SOPC Board A (*the transmitter*) includes a "Datalink"/"Link" layer using "IEEE1394" and a "physical layer" using an "IEEE1394 Chip" (*Id.* at 53, Fig. 4 & Table 1), any may be implemented on a "fireIP" module containing a "LLC-PHY interface" (*link layer and physical layer circuitry*). *Id.* at 53 (Sec. 5), 54 (Sec. 5.1); *see also id.* at 55, Sec. 7.1 ("two fireIP modules").

[68(a)] packetizing circuitry configured to generate M-bit input words,

213. Yusairi in view of Shin renders obvious this limitation.

214. Yusairi discloses the SOPC Board A (*the transmitter including physical and link layer circuitry*) comprises *packetizing circuitry*, as it includes the fireIP module, which includes a link layer and physical layer, as discussed for [68(pre)],

and the link layer encapsulates data into packets and passes them on for transmission. Yusairi at 54, Secs. 5, 5.2; 56, Secs. 7.1.2; 53, Secs. 4 and 4.3. Specifically “the module link_trans for the link layer” does so by “generat[ing] a link header, attach[ing] it to IP data, and put[ting] the data into a fifo of trans module.” Yusairi at 54, Secs. 5, 5.2. The header includes “encapsulation headers.” Yusairi at 54, Sec. 5.2). Then, “[w]hen the packet is ready to be transmitted, a signal is asserted to inform [the] trans module that IP data has been put into the fifo and [is] ready to transmit.” Yusairi at 54, Sec. 5.2. Then, trans “starts transmitting link packets ... to PHY [physical layer]” and “request[s] PHY that the data be transmitted.” Yusairi at 54, Sec. 5.1. The function of link_trans combining headers and IP data into packets aligns with the ’307 patent’s description of its “*packetizing circuit*,” which “group[s]” “data blocks (sometimes referred to as words) ... into cells,” where cells can be packets.’ 307 Patent at 14:8-10; 4:38-40. Yusairi, therefore, discloses *link layer circuitry* that *include[s] packetizing circuitry*.

215. Yusairi in view of Shin renders obvious the *packetizing circuitry* being *configured to generate M-bit input words*. As discussed for [2] in Grounds 1 and 2, Shin’s packet (*cell*) can be formed into codes (*M-bit input words*) that are encoded by block encoding into symbols (*code words*). Shin describes an example block encoding that uses “8b/9b encoding,” which encodes “8-bit codes [to] ... 9-bit symbols.” Shin at [0131]. A POSITA would have understood that block encoding

encodes M-bit codes (*input words*), such as the 8-bit codes, to N-bit symbols (*code words*), such as the 9-bit symbols. Shin at [0131].

216. A POSITA would have been motivated to use Shin's block encoding methods, which *generate M-bit input words*, with Yusairi's link-layer *packetizing circuitry* in generating packets for transmission, because it was known such block encoding methods help maintain DC balance between transmitters and receivers, such as Yusairi's SOPC Boards, which is a benefit taught by Shin. Shin at [0131]-[0132]. A POSITA would have also been motivated to use Shin's block encoding methods with Yusairi's link-level packetizing circuitry because it was a well-known industry practice to perform packetization and block encoding at the link layer. For instance, both HDMI 1.0 and 802.3 standards used this configuration. HDMI 1.0 describes a link layer encoder/serializer, illustrated and annotated in Figure 5-1 below, which generates an 8-bit input data set (the *M-bit input word*) for encoding to a 10-bit output value.

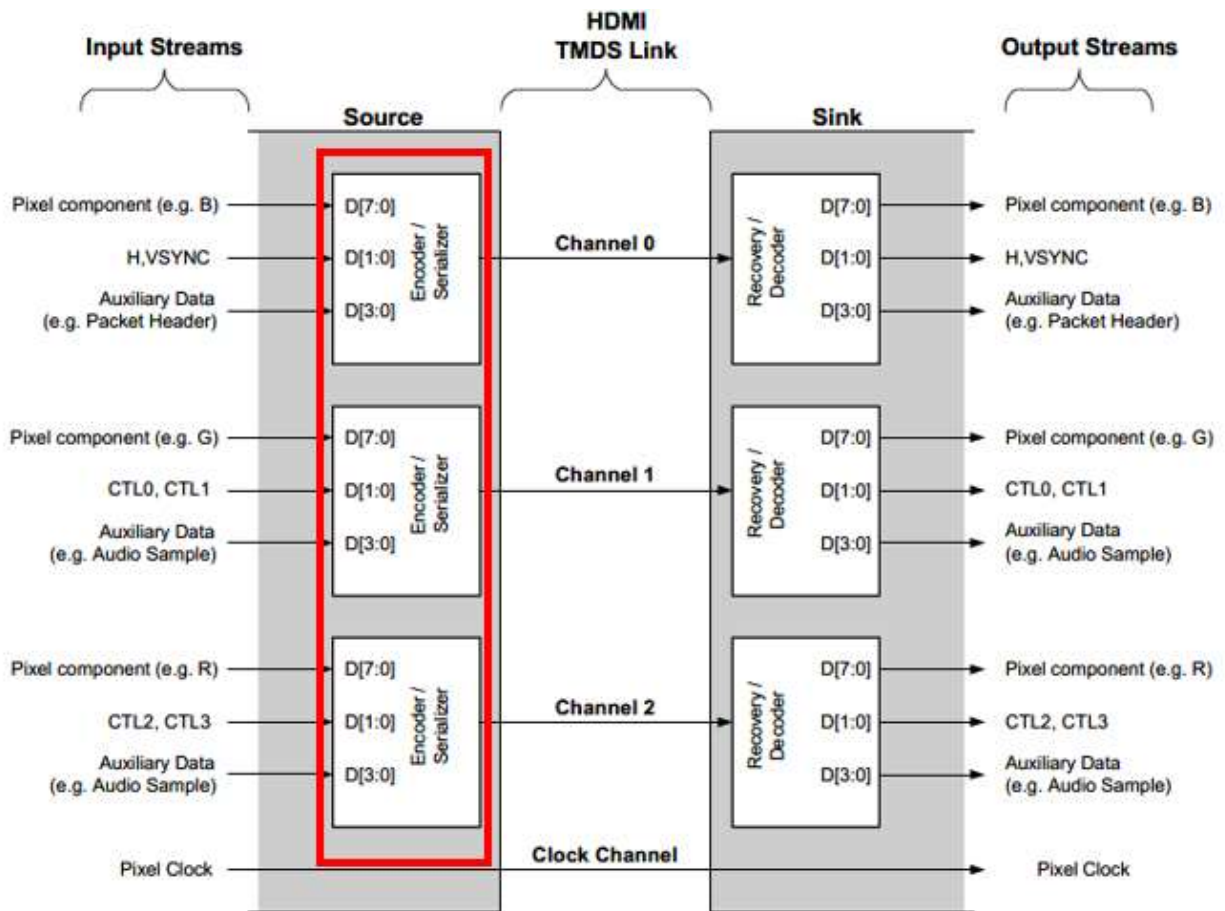


Figure 5-1 HDMI Encoder/Decoder Overview

HDMI 1.0 at 40, Fig. 5-1; HDMI. 1.0 at 56 (“D is 8-bit pixel data” that is “[t]he encoder input data set”). Since packet data is input to the “Source,” which is part of the link architecture illustrated in Figure 5-1, and the Encoder/Serializer uses the 8-bit pixel data from the packet data as an input, A POSITA would have understood that the 8-bit pixel data from is being generated from the packet data at the link layer. HDMI 1.0, at 40 (“The input stream to the Source’s encoding logic will contain video pixel, packet and control data. The packet data consists of audio and auxiliary data and associated error correction codes. These data items are processed in a

variety of ways and are presented to the TMDS encoder as either 2 bits of control data, 4 bits of packet data or 8 bits of video data per TMDS channel.”).

217. Further, 802.3 standards also illustrates that it was common practice to have packetizing circuitry within the link layer. In 802.3 standards, the MAC layer is part of the link layer (also called data link layer). Figure 35-1 from 802.3 is provided below with the MAC layers of the link layer annotated in **green**.

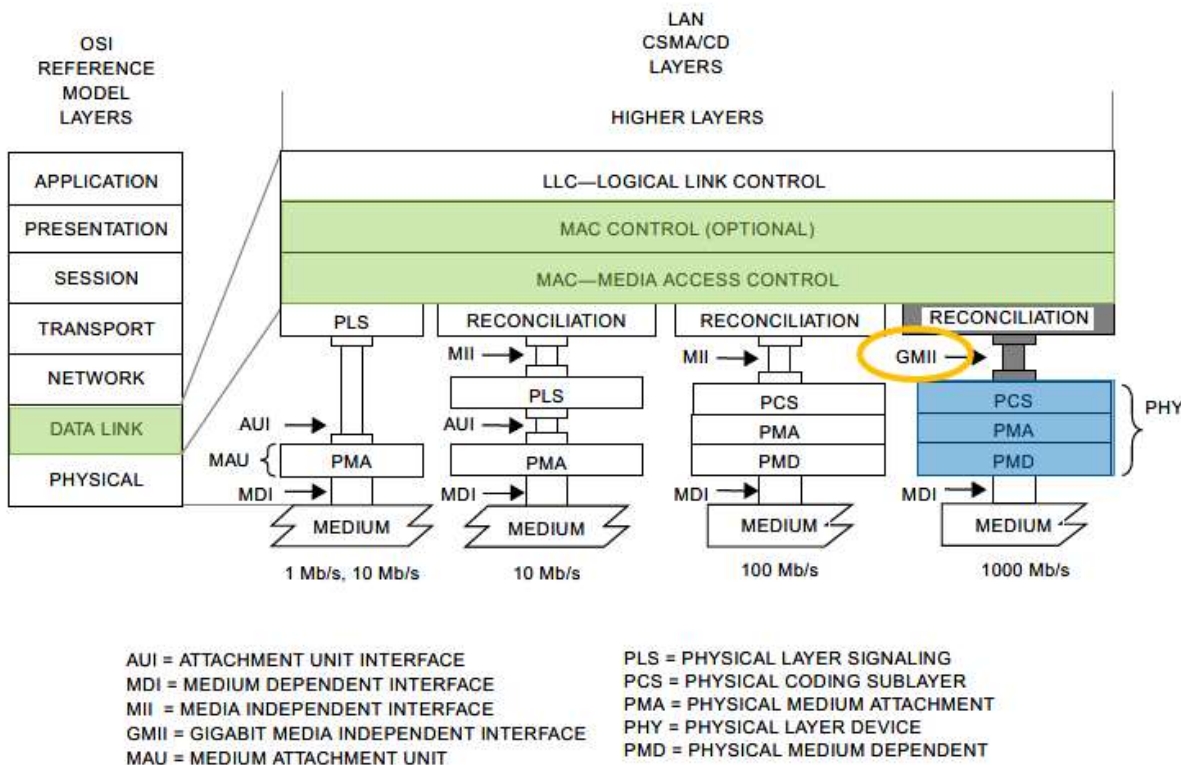


Figure 35-1—GMII location in the OSI protocol stack

802.3 (Ex.1011) at 896, Fig. 35-1. The reconciliation layer, in **gray**, serves as an interface between the MAC layer (link layer) and the physical layer, annotated in **blue**. A PLS_DATA primitive originates from the MAC Layer and causes at GMII

signal at the reconciliation layer interface. 802.3, at 98 (“The PLS_DATA primitives support the transfer of data from a single MAC sublayer entity to all other peer MAC sublayer entities contained within the same LAN defined by the broadcast medium.”). The GMII signals are indicated using **orange** above in Figure 35-1. Figure 35-2 provided below illustrates and annotates the PLS_DATA.request (the PLS_data primitive) provided to the reconciliation sublayer, which outputs the GMII signal in response. 802.3 at 98 (referring to the “PLS_DATA.request,” “This primitive is generated by the MAC sublayer to request the transmission of a single data bit on the physical medium or to stop transmission.”).

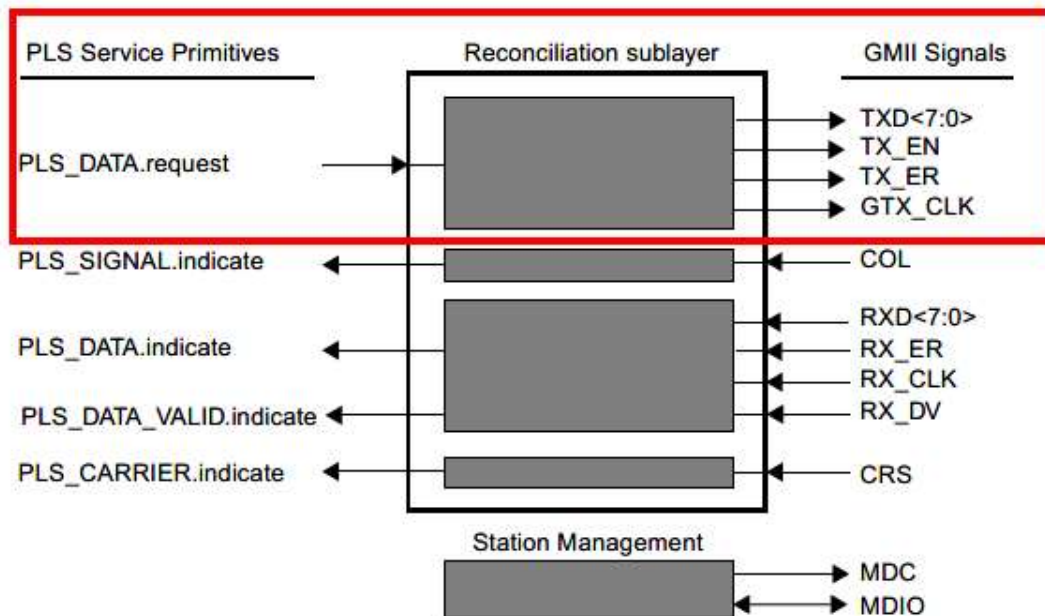


Figure 35-2—Reconciliation Sublayer (RS) inputs and outputs and STA connections to GMII

802.3 at 898, Fig. 35-2. During 8B/10B encoding, the GMII signal provides the M-bit inputs to a physical layer encoder. This path is illustrated below in Figure 36-3.

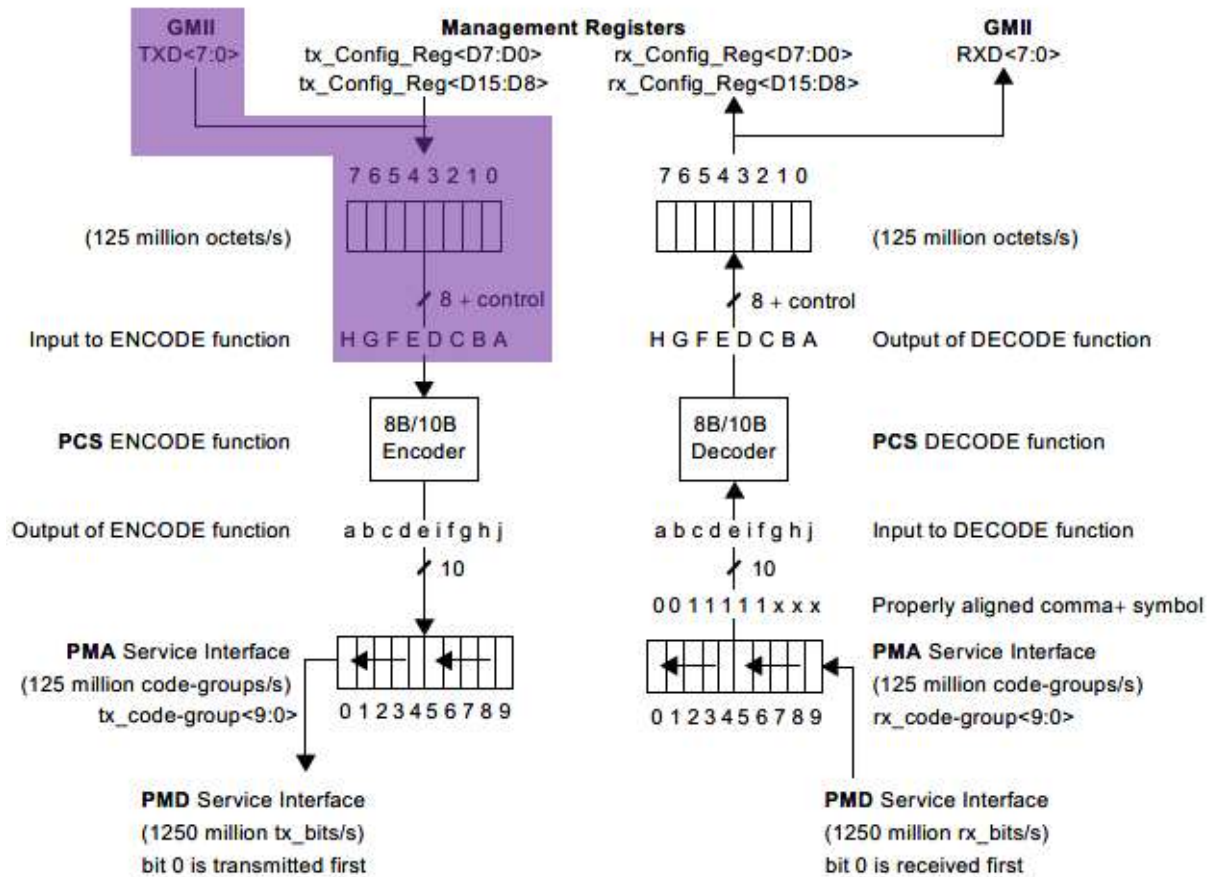


Figure 36-3—PCS reference diagram

802.3 at 928, Fig. 36-3. In view of this, a POSITA would have understood that the M-input bits of the block encoding (the *M-bit input words*) originate from the PLS_DATA primitive generated by *packetizing circuitry* at the MAC layer (the link layer). Thus, a POSITA would have understood that prior systems, such as HDMI 1.0 and 802.3, packetized (generated the M-bit input words) at a link layer, and thus, they would have motivated a POSITA to implement the same in Yusairi.

218. Further, Yusairi's teachings regarding performing packetization in the link layer would have motivated a POSITA to implement this technique of performing packetization and block encoding (such as the block encoding taught by Shin) in the link layer of Yusairi, given this technique's well-known use in well-known standards at the time, such as HDMI and Ethernet, as previously noted. HDMI 1.0 at 40, Fig. 5-1; 802.3 at 98, 896, 928, Figs. 35-1, 35-1, 36.3. A POSITA also would have had a reasonable expectation of success in making this combination because it would have been a minor modification involving changes only to the encoding scheme used and would have been implemented on Yusairi's existing link layer packetizing circuitry. A POSITA would have also been motivated to modify Yusairi with Shin's block encoding methods that use packetizing circuitry to generate M-bit input words for the reasons stated with respect to limitations [2] and [6].

[68(b)] wherein the input words are indicative of application data and control bits; and

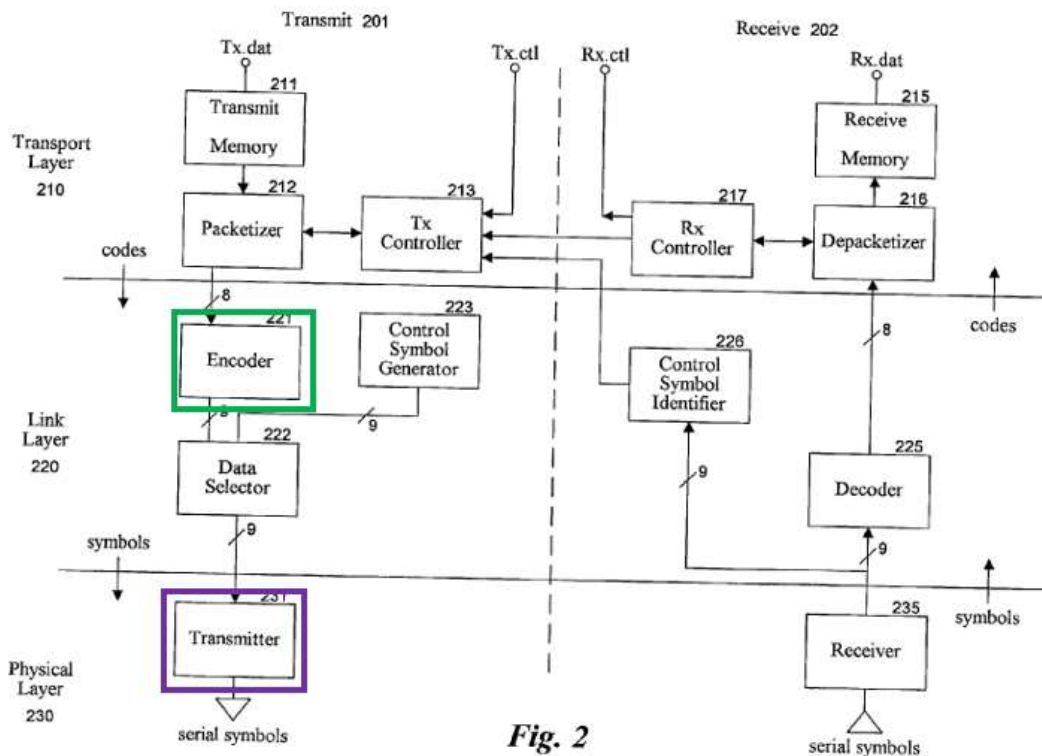
219. Yusairi in view of Shin renders obvious this limitation for the reasons discussed above at limitation [23(e)].

[68(c)] encoding and transmission circuitry coupled to the packetizing circuitry and configured to encode the input words in accordance with a block code to generate a code word sequence of N-bit code words, where $N > M$,

220. Yusairi in view of Shin renders obvious this limitation.

221. As discussed above for 68[a], Yusairi discloses link-layer packetizing circuitry, and it would have been obvious to a POSITA to use Shin's block encoding methods with Yusairi's link-layer packetizing circuitry. Further, as also discussed for [68(a)], Yusairi's stream_trans module generates packets and passes them to the trans module, which then transmits them to the physical layer with a request that they be transmitted out. Yusairi at 54. Secs. 5, 5.1, 5.2. Thus, Yusairi discloses *transmission circuitry* that is *coupled to the packetizing circuitry* by also being located in the link layer.

222. In addition, Shin discloses *encoding and transmission circuitry* in a link layer that is coupled to packetizing circuitry and that performs Shin's block encoding discussed above for 68[a], as illustrated below in Figure 2.



Specifically, Shin discloses an encoder 221 (*encoding circuitry*) (**green**) and transmitter 231 (*transmission circuitry*) (**purple**) coupled to a packetizer 212, where the encoder of “[t]he link layer encodes the codes to be transmitted via its communications link into symbols ... and provides those symbols the [transmitter of] the physical layer” that “transmits the serialized symbols via the communications link.” Shin at [0076], [0077]. Shin further discloses encoding methods that *encode the input words in accordance with a block code to generate a code word sequence of N-bit code words, where $N > M$* for the reasons discussed above at limitation [2] of Grounds 1 and 2.

223. A POSITA would have been motivated to modify Yusairi's link layer by coupling Yusairi's link-layer packetizing circuitry with Shin's link-layer encoder and transmitter (*encoding and transmission circuitry*) that encode and transmit in block code that generates N -bit code words from M -bits, where $N > M$, for the reasons discussed above for limitations [2] and [6]. Further, a POSITA would have also been motivated to make this combination because, like Yusairi's link layer packetizing circuitry, Shin discloses link-layer encoding and transmission circuitry, which would have motivated a POSITA to make this combination. Further, a POSITA would have had a reasonable expectation of success given that Shin's encoder and transmitters are both in the link layer already, like Yusairi's packetizing circuitry.

224. Shin further discloses encoding methods that *encode the input words in accordance with a block code to generate a code word sequence of N -bit code words, where $N > M$* for the reasons discussed above at limitation [2] of Grounds 1 and 2.

[68(d)] the code word sequence is indicative of at least one cell of the input words, the cell includes at least two of the control bits,

225. Yusairi in view of Shin renders obvious this limitation.

226. Yusairi discloses at least one packet (*cell*) that includes *at least two of the control bits*, because the transaction code bits and field type bits are within the same packet. Yusairi at 54, Sec. 5.2. Shin discloses that the code word sequence is

indicative of at least one cell of the input words, as described with respect to limitation [2] of Grounds 1 and 2.

227. A POSITA would have been motivated to encode Yusairi's packets using Shin's block encoding techniques to render this claim element obvious for the same reasons discussed with respect to limitations [2] and [6].

[68(e)] and said at least two of the control bits have multiple levels of communication protocol functionality,

228. Yusairi discloses this limitation for the reasons discussed with respect to [23(f)].

[68(f)] wherein the encoding and transmission circuitry is configured to be coupled to a serial link and to transmit encoded data over the serial link,

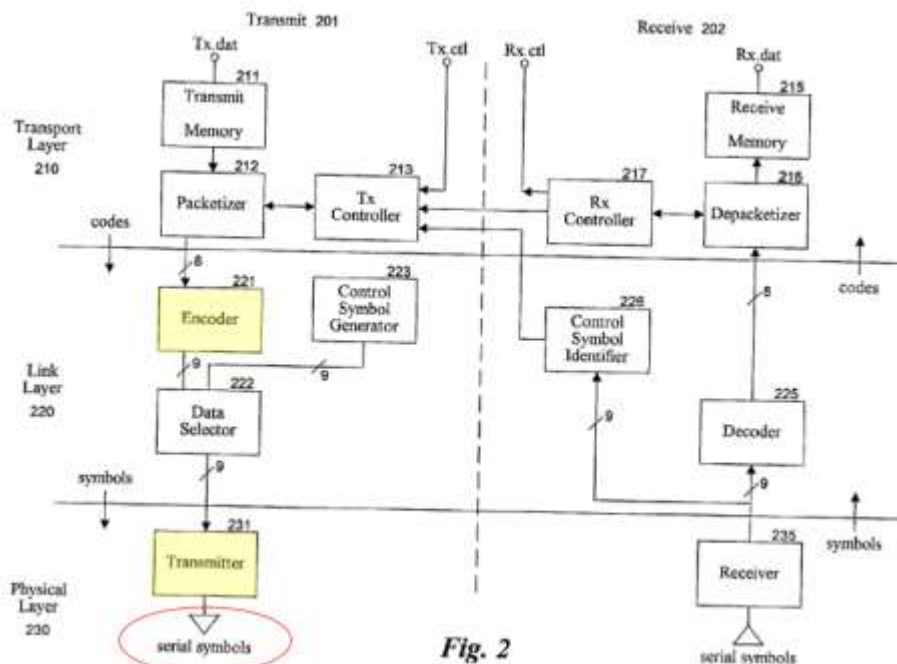
229. Yusairi in view of Shin renders obvious this limitation.

230. First, Yusairi discloses this limitation. As discussed for 68[c], Yusari discloses *encoding and transmission circuitry* through its disclosure that Yusairi's stream_trans module performs encapsulation of packets and passes them to the trans module, which then transmits them to the physical layer with a request that they be transmitted out. Yusairi at 54, Secs. 5, 5.1, 5.2.

231. Further, Yusairi discloses that its encoding and transmission circuitry is configured to be coupled to a serial link and to transmit encoded data over the serial link. Yusairi discloses that to send packet data a 'Bus request' must be performed by the link layer to the physical layer in which "[t]he Lreq terminal

controlled by LLC sends serial service requests to PHY for taking the access of the serial-bus [serial link] for packet transmission” (encoding and transmission circuitry is configured to be coupled to a serial link and to transmit encoded data over the serial link). Yusairi at 52-52, Secs. 3.2 3.2.1.

232. In addition, Yusairi in view of Shin renders this limitation obvious. As discussed for [68(c)], Shin discloses *encoding and transmission circuitry* in the form of its encoder 221 and transmitter 231 that are *configured to be coupled to a serial link and to transmit encoded data over the serial link*. Specifically, as shown in Figure 2 of Shin, the encoder 221 is coupled to the transmitter 231, which is coupled to a “communications link” over which it transmits “serial symbols” (*coupled to a serial link and to transmit encoded data over the serial link*). Shin at [0079], [0080], Fig. 2.



Shin at Fig. 2 (annotated).

233. A POSITA would have been motivated to combine Shin's encoder 221 and transmitter 231 (*encoding and transmission circuitry*) that are *configured to be coupled to a serial link and to transmit encoded data over the serial link* with Yusairi's link layer packetizing circuitry for the same reasons set forth above for [68(a)] and [68(c)].

[68(g)] said encoded data including the code words of the code word sequence.

234. Yusairi in view of Shin renders obvious this limitation for the reasons discussed with respect to limitation [2] of Grounds 1 and 2.

235. A POSITA would have been motivated to use Shin's block encoding with Yusairi's link layer packetizing circuitry this limitation obvious for the reasons discussed with respect to limitations [2] and [6].

12. Claim 70

[70] The transmitter of claim 68, wherein the block code is specified by a line code, the line code also specifies special characters, and the encoding and transmission circuitry is configured to generate the encoded data such that said encoded data are a sequence of at least one of the special characters and the code words of the code word sequence.

236. Yusairi in view of Shin renders obvious this limitation.

237. Shin discloses and renders obvious the block code is specified by a line code, the line code also specifies special characters and that said encoded data are a sequence of at least one of the special characters and the code words of the code word sequence for the reasons discussed with respect to [2] of Grounds 1 and 2.

238. Further, Shin discloses or renders obvious *the encoding and transmission circuitry is configured to generate the encoded data* for the reasons discussed with respect to limitations [68(c)] and transmit the encoded data as line code for the reasons discussed with respect to [68(f)].

239. A POSITA would have been motivated to modify Yusairi with Shin's encoding method to render this limitation obvious for the reasons discussed with respect to limitations [2] and [6].

13. Claim 71

[71] The transmitter of claim 70, wherein none of the special characters of the encoded data matches a bit sequence of the code word sequence.

240. Yusairi in view of Shin renders obvious this limitation for the reasons discussed with regard to limitation [25].

241. It would have been obvious to a POSITA to modify Yusairi with Shin's encoding methods that use special characters that do not match a bit sequence for the reasons discussed with respect to limitations [2] and [6]. Further a POSITA would have also been motivated to modify Yusairi with Shin's special characters that do not match bit sequences of symbols (codes within a code word sequence), the receiver can identify the special characters from among the transmitted code words and perform a function indicated by the special character, such as notifying a transmitter to stop communicating data using an XOFF special character. Shin at [0079].

14. Claim 72

[72] The transmitter of claim 70, wherein said at least one of the special characters is a link training character.

242. Yusairi in view of Shin renders obvious this limitation.

243. The '307 Patent discloses that a special character called a “link training character” can be transmitted “[t]o establish symbol framing” ’307 Patent, at 17:1-5, 17:22-23. Similarly, Shin teaches “[a] transmitting communications node periodically transmits synchronization primitives so that the receiving communications node can properly align with that synchronization primitive.” Shin at [0096]. A POSITA would understand that the use of the synchronization primitive establishes symbol framing to properly align the transmitting communications node and receiving communications node, since “the communications node . . . knows the primitive is correctly aligned on the symbol boundary.” Shin at [0096]. Therefore, Shin’s synchronization primitive teaches *a link training character*.

244. A POSITA would understand that the synchronization primitive is a *special character* because “[t]he synchronization primitives have a unique number of transitions that is not found in any control primitive or in-band symbol.” Shin at [0101].

245. A POSITA would have been motivated to combine Shin’s disclosure of an a synchronization primitive acting as *a link training character* with Yusairi because doing so would allow for the SOPC Board B (acting as a receiver) to align

symbol boundaries of transmitted encoded data. Shin at [0096] (“A transmitting communications node periodically transmits synchronization primitives so that the receiving communications node can properly align with that synchronization primitive. When a communications node receives a synchronization primitive, it knows that the primitive is correctly aligned on a symbol boundary. ”).

15. Claim 74

[74] The transmitter of claim 70, wherein said at least one of the special characters is an idle character.

246. Yusairi in view of Shin renders this limitation obvious for the reasons discussed with respect to limitation [2] of Grounds 1 and 2.

247. A POSITA would have been motivated to modify Yusairi to use IDLE characters because doing so was a known way to synchronize communication nodes, such as the SOPC Boards, during communication, since Shin discloses “an IDLE primitive may be a synchronization primitive,” helps “a receiving communications node [to] properly align with [the] synchronization primitive” transmitted by a transmitting communications node.” Shin at [0096], [0097].

16. Claim 75

[75] The transmitter of claim 70, wherein said at least one of the special characters is a flow control character.

248. Yusairi in view of Shin renders obvious this limitation.

249. Shin discloses *at least one of the special characters is a flow control character*. The '307 Patent states that an “idle character” can be for flow control.

'307 Patent at 9:32-37 (“For another example, special characters (e.g., N-bit code words known as ‘idle’ characters, where N=10 in preferred embodiments) which can be recognized by physical layer circuitry are used in accordance with the invention by link layer circuitry for flow control.”). As discussed above with respect to limitation [2], Shin discloses an IDLE primitive or symbol that is a *special character*. Shin’s “IDLE primitive may be a synchronization primitive” that “maintain proper synchronization” between transmitters and receivers. Shin at [0097], [0105]. A POSITA would have understood that Shin’s IDLE primitive performing a type of flow control when used for synchronization. Thus, Shin discloses at least one special character that is a flow control character (the IDLE primitive), since Shin’s IDLE primitive can also be used for flow control.

250. A POSITA would have been motivated to combine Shin’s disclosure of an IDLE primitive acting as a *flow control character* with Yusairi because doing so would allow for the SOPC Board A (acting as a transmitter) to be in synchronization with SOPC Board B (acting as the receiver) when transmitting encoded data with IDLE characters. Shin at [0096] (“A transmitting communications node periodically transmits synchronization primitives so that the receiving communications node can properly align with that synchronization primitive. When a communications node receives a synchronization primitive, it knows that the primitive is correctly aligned on a symbol boundary.

C. Secondary Considerations of Non-obviousness.

251. I am not aware of any evidence of the existence of any secondary considerations of non-obviousness regarding the Challenged Claims. In the event patent owner asserts any such evidence, I reserve the right to address it.

IX. CONCLUSION

This declaration and my opinions herein are made to the best of my knowledge and understanding, and based on the material available to me, at the time of signing this declaration. I declare that all statements made herein on my own knowledge are true and that all statements made on information and belief are believed to be true, and further, that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 or Title 18 of the United States Code.

Date: 10/1/2024



Andrew Wolfe