David C. Marcus (SBN 158704)
david.marcus@wilmerhale.com
James M. Dowd (SBN 259578)
james.dowd@wilmerhale.com
Matthew J. Hawkinson (SBN 248216)
matthew.hawkinson@wilmerhale.com
Aaron Thompson (SBN 272391)
aaron.thompson@wilmerhale.com
WILMER CUTLER PICKERING
HALE AND DORR LLP
350 South Grand Avenue, Suite 2100
Los Angeles, CA 90071
Telephone: (213) 443-5300
Facsimile: (213) 443-5400

William F. Lee (pro hac vice)
william.lee@wilmerhale.com
WILMER CUTLER PICKERING
HALE AND DORR LLP
60 State Street
Boston, MA 02109
Telephone: (617) 526-6000
Facsimile: (617) 526-5000

Attorneys for Defendants and Counterclaim-Plaintiffs
Hughes Communications Inc.
Hughes Network Systems LLC
DISH Network Corporation,
DISH Network LLC, and
dishNET Satellite Broadband LLC

Additional Counsel Listed on Signature Page

Apple 1115

UNITED STATES DISTRICT COURT
CENTRAL DISTRICT OF CALIFORNIA

THE CALIFORNIA INSTITUTE OF TECHNOLOGY, Plaintiff and Counter-Defendant, vs. HUGHES COMMUNICATIONS INC., HUGHES NETWORK SYSTEMS LLC, DISH NETWORK CORPORATION, DISH NETWORK LLC, and DISHNET SATELLITE BROADBAND LLC, Defendants and Counter-Plaintiffs.	Case No. 2:13-cv-07245-MRP-JEM **EXPERT REPORT OF DR. BRENDAN FREY REGARDING INVALIDITY OF PATENTS-IN-SUIT**

EXPERT REPORT OF DR. BRENDAN FREY REGARDING INVALIDITY OF PATENTS-IN-SUIT

I. SUMMARY OF REPORT

1. I have been retained as an expert in this case by counsel for Defendants and Counter-Plaintiffs Hughes Communications Inc., Hughes Network Systems LLC, DISH Network Corporation, DISH Network LLC, and dishNET Satellite Broadband LLC (collectively, "Defendants"). I expect to testify at trial about the matters set forth in this report, if asked about these matters by the Court or by the parties' attorneys.

2. I understand that the Plaintiff and Counter-Defendant in this proceeding, the California Institute of Technology ("Plaintiff" or "Caltech") has asserted against Defendants the following four patents:

- U.S. Patent No. 7,116,710 (the "'710 patent");
- U.S. Patent No. 7,421,032 (the "'032 patent");
- U.S. Patent No. 7,916,781 (the "'781 patent"); and
- U.S. Patent No. 8,284,833 (the "'833 patent").

3. I further understand that Plaintiff has asserted the following claims:

- claims 1, 4, 6, 15, 20, and 22 of the '710 patent;
- claims 1, 18, 19, and 22 of the '032 patent;
- claims 16 and 19 of the '781 patent; and
- claims 1, 2, 4, and 8 of the '833 patent.

4. I have been asked for my expert opinion on whether the claims listed in the preceding paragraph (the "asserted claims") are valid. In my opinion, all of the asserted claims are invalid for the reasons stated below.

5. I have also been asked for my opinion on whether various documents, including an email from an inventor dated March 7, 2000, demonstrate conception

of the claimed invention. In my opinion, these documents do not demonstrate conception for the reasons stated below.

6. I have also been asked for my opinion regarding whether three references (two by Luby et al. and one by Richardson et al.) were material to the claimed invention. In my opinion, as explained below, these three references, none of which were before the patent office during prosecution of the asserted patents, were material to the claimed invention.

BACKGROUND

A. Qualifications and Experience

7. I received a B.Sc. with Honors in Electrical Engineering from the University of Calgary in 1990, a M.Sc. in Electrical and Computer Engineering from the University of Manitoba in 1993, and a Ph.D. in Electrical and Computer Engineering from the University of Toronto in 1997. Since July 2001, I have been at the University of Toronto, where I am a Professor of Electrical and Computer Engineering and Computer Science.

8. During my career I have conducted research in the areas of graphical models, error-correcting coding, machine learning, genome biology and computer vision. I have authored more than 200 publications and am named as an inventor on nine patents issued by the U.S. Patent and Trademark Office.

9. I have received a number of honors and awards for the research I have conducted. In 2008, I was named a Fellow of the Institute for Electrical and Electronic Engineers (IEEE), an honor given to a person with an "extraordinary record or accomplishments" in the field of electrical engineering. In 2009, I was named a Fellow of the American Association for the Advancement of Science (AAAS), an honor that recognizes "efforts on behalf of the advancement of science or its applications which are scientifically or socially distinguished."

10. In 2009, I was awarded a Steacie Fellowship for my work on the theory and implementation of artificial and natural mechanisms for inferring patterns from data. The Steacie Fellowship is awarded by the Natural Sciences and Engineering Research Council of Canada (NSERC) to "outstanding and highly promising scientists and engineers" who are faculty members of Canadian universities. In 2011, I received the NSERC's John C. Polanyi Award, in recognition of my research on inferring genetic codes embedded in DNA that direct activities within cells.

11. Throughout my career I have received funding from various governmental agencies to support my research, including the Natural Sciences and Engineering Research Council of Canada, the Canadian Institutes of Health Research, and the Canadian Institute for Advanced Research.

12. A copy of my *curriculum vitæ* is attached to this report as Exhibit A.

B. Understanding of the Law

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

i) Invalidity in General

14. A patent is presumed valid, and a challenger to the validity of a patent must show invalidity of the patent by clear and convincing evidence. Clear and convincing evidence is evidence that makes a fact highly probable.

ii) Anticipation

15. A patent claim is invalid if it is "anticipated" by prior art. For the claim to be invalid because it is anticipated, all of its requirements must have existed in a single device or method that predates the claimed invention, or must have been described in a single publication or patent that predates the claimed invention.

16. The description in a written reference does not have to be in the same words as the claim, but all of the requirements of the claim must be there, either stated or necessarily implied, so that someone of ordinary skill in the art, looking at that one reference would be able to make and use the claimed invention.

17. A patent claim is also anticipated if there is clear and convincing proof that, more than one year before the filing date of the patent, the claimed invention was: in public use or on sale in the United States; patented anywhere in the world; or described in a printed publication anywhere in the world. This is called a statutory bar.

iii) Obviousness

18. A patent claim is invalid if the claimed invention would have been obvious to a person of ordinary skill in the art at the time the application was filed. This means that even if all of the requirements of a claim cannot be found in a single prior art reference that would anticipate the claim or constitute a statutory bar to that claim, the claim is invalid if it would have been obvious to a person of ordinary skill who knew about the prior art.

19. The determination of whether a claim is obvious should be based upon several factors, including:

- the level of ordinary skill in the art that someone would have had at the time the claimed invention was made;
- the scope and content of the prior art;
- what difference, if any, existed between the claimed invention and the prior art.

20. In considering the question of obviousness, it is also appropriate to consider any secondary considerations of obviousness or non-obviousness that may be shown. These include:

- commercial success of a product due to the merits of the claimed invention;

- a long felt need for the solution provided by the claimed invention;
- unsuccessful attempts by others to find the solution provided by the claimed invention;
- copying of the claimed invention by others;
- unexpected and superior results from the claimed invention;
- acceptance by others of the claimed invention as shown by praise from others in the field or from the licensing of the claimed invention; and
- independent invention of the claimed invention by others before or at about the same time as the named inventor thought of it.

21. A patent claim composed of several elements is not proved obvious merely by demonstrating that each of its elements was independently known in the prior art. In evaluating whether such a claim would have been obvious, it is relevant to consider if there would have been a reason that would have prompted a person of ordinary skill in the field to combine the elements or concepts from the prior art in the same way as in the claimed invention. For example, market forces or other design incentives may be what produced a change, rather than true inventiveness. It is also appropriate to consider:

- whether the change was merely the predictable result of using prior art elements according to their known functions, or whether it was the result of true inventiveness;
- whether there is some teaching or suggestion in the prior art to make the modification or combination of elements claimed in the patent;
- whether the innovation applies a known technique that had been used to improve a similar device or method in a similar way; or
- whether the claimed invention would have been obvious to try, meaning that the claimed innovation was one of a relatively small number of possible approaches to the problem with a reasonable expectation of success by those of ordinary skill in the art.

22. In considering obviousness, it is important to be careful not to determine obviousness using the benefit of hindsight; many true inventions might seem obvious after the fact.

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

iv) <u>The "Written Description" Requirement</u>

24. A patent claim is invalid if the patent specification does not contain a written description of the invention to which the claim is directed. To satisfy the written description requirement, a patent specification must describe the claimed invention in sufficient detail that one of ordinary skill in the art can reasonably conclude that the inventor had possession of the claimed invention.

25. An applicant shows possession of the claimed invention by describing the claimed invention with all of its limitations using such descriptive means as words, structures, figures, diagrams, and formulas that fully set forth the claimed invention. A description that merely renders the invention obvious does not satisfy the written description requirement.

v) <u>Inequitable Conduct and Materiality</u>

26. I have been informed that during prosecution, inventors have a duty to disclose to the Patent Office all information known to the inventors that is material to the patentability of the claims being examined.

27. Information is deemed to be material to patentability when it is not cumulative to information already before the Patent Office, and when: (1) it establishes, by itself or in combination with other information, that a claim was unpatentable; or (2) it refutes, or is inconsistent with, a position the applicant takes

in (a) opposing an argument of unpatentability relied on by the Patent Office, or (b) asserting an argument of patentability.

C. Materials Reviewed

28. Among the materials I have reviewed in forming my opinions are:

- The '710, '032, '781, and '833 patents;
- The prosecution histories of the '710, '032, '781, and '833 patents;
- The prior art of record that was available to the patent examiner;
- The prior art references discussed herein;
- Claim Construction Order dated August 6, 2014 (Dkt. No. 105);
- Declaration of Stephen B. Wicker, dated Oct. 6, 2014 (Dkt. No. 130-10);
- Transcript of the October 14, 2014 deposition of Stephen B. Wicker;
- IPR Petition No. IPR2015-00067 and accompanying exhibits, including the declaration of Henry D. Pfister;
- IPR Petition No. IPR2015-00068 and accompanying exhibits, including the declaration of Henry D. Pfister;
- IPR Petition No. IPR2015-00060 and accompanying exhibits, including the declaration of Henry D. Pfister;
- IPR Petition No. IPR2015-00059 and accompanying exhibits, including the declaration of Henry D. Pfister;
- IPR Petition No. IPR2015-00061 and accompanying exhibits, including the declaration of Henry D. Pfister;
- IPR Petition No. IPR2015-00081 and accompanying exhibits, including the declaration of Henry D. Pfister;
- Transcript of the December 11, 2014 deposition of inventor Aamod Khandekar;
- Transcript of the January 7, 2015 deposition of inventor Hui Jin;
- Transcript of the Jan 15, 2015 deposition of Dariush Divsalar;
- Laboratory Notebook of Robert McEliece (CALTECH000004472-603);
- Caltech's Supplemental Responses to Defendants' First Set of Interrogatories, Nos. 3-5, Jan. 11, 2015;

- Caltech's Second Supplemental Responses to Interrogatories 1-5 and Caltech's First Supplemental Responses to Interrogatories 6-11;
- Email from Brendan Frey to Dariush Divsalar dated Dec. 8, 1999 (CALTECH000024021);
- Khandekar, Aamod ("Capacity Achieving Codes on the Binary Erasure Channel") (CALTECH000007321-7349).
- Khandekar, Aamod, "Graph-based Codes and Iterative Decoding," thesis dated June 10, 2002.
- McEliece Email dated March 7, 2000 (CALTECH000008667)
- Luby, M. et al., "Practical Loss-Resilient Codes," *STOC '97* (1997)
- Luby, M. et al., "Analysis of Low Density Codes and Improved Designs Using Irregular Graphs," *STOC '98*, p. 249-259 (1998)
- Richardson, T. et al. "Design of provably good low-density parity check codes," *IEEE Transactions on Information Theory* (1999) (preprint)

29. Level of Ordinary Skill in the Art

30. In my opinion, based on the materials and information I have reviewed, and on my extensive experience working with people in the technical areas relevant to the patents-in-suit (*i.e.* in the field of code design), a person of ordinary skill in the art is a person with a Ph.D. in electrical or computer engineering with emphasis in signal processing, communications, or coding, or a master's degree in the above area with at least three years of work experience this field at the time of the alleged invention.[1] I understand that Caltech has agreed with this definition of the level of ordinary skill in this case.[2][3]

[1] I was asked to use a similar qualification for a "person of ordinary skill in the art" for purposes of a declaration that I understand was filed in connection with petitions for *Inter Partes Review* of the asserted patents. *See* Declaration of Brendan Frey dated October 14, 2014, at ¶2.
[2] Reporter's Transcript of Claim Construction and Motion Hearing of July 9, 2014, Ex. 1026, at 98.
[3] This is also consistent with testimony given by, *e.g.*, Dr. Dariush Divsalar, an author of one of the prior art references discussed in this report (*see* Divsalar Dep. at 55-56).

D. Claim Constructions Used in This Report

31. I understand that the parties have agreed on the following claim constructions:

Claim Term	Agreed-Upon Construction
"irregularly" ('710 and '032 patents)	"a different number of times"
"interleaving" / "interleaver" / "scramble" ('710 patent)	"changing the order of data elements" / "module that changes the order of data elements"
"sums of bits in subsets of the information bits" / "summing of bits in a subset of the information bits" / "adding additional subsets of information bits" ('781 patent)	"the result(s) of adding together two or more information bits from a subset of information bits" / "adding together two or more information bits from a subset of information bits"
"wherein two or more memory locations of the first set of memory locations are read by the permutation module different times from one another" ('833 patent)	"where two or more memory locations of the first set of memory locations are read by the permutation module a different number of times from one another"
"permutation module" ('833 patent)	"a module that changes the order of data elements"

32. I further understand that the Court in this case has issued a claim construction order construing certain disputed claim terms as follows:

Claim Term	Court's Construction
"transmitting" / "transmission" ('032 patent)	"sending over a channel"

"codeword" ('781 patent)	"a discrete encoded sequence of data elements"
"repeat" ('710 and '032 patents)	plain meaning[4]
"combine" / "combining" ('833 patent)	"perform logical operations on"
Equation in claim 1 of the '032 patent ('032 patent)	"the parity bit x_j is the sum of (a) the parity bit x_{j-1} and (b) the sum of a number, 'a,' of randomly chosen irregular repeats of the message bits"
Tanner Graph term in claims 11 and 18 of '032 patent ('032 patent)	"a graph representing an IRA code as a set of parity checks where every message bit is repeated, at least two different subsets of message bits are repeated a different number of times, and check nodes, randomly connected to the repeated message bits, enforce constraints that determine the parity bits"

33. For the purposes of this report, I have used the constructions given in the two tables above. For all other claim terms, I have used the plain and ordinary meaning the term would have to one of ordinary skill in the art.

II. OVERVIEW OF THE TECHNOLOGY

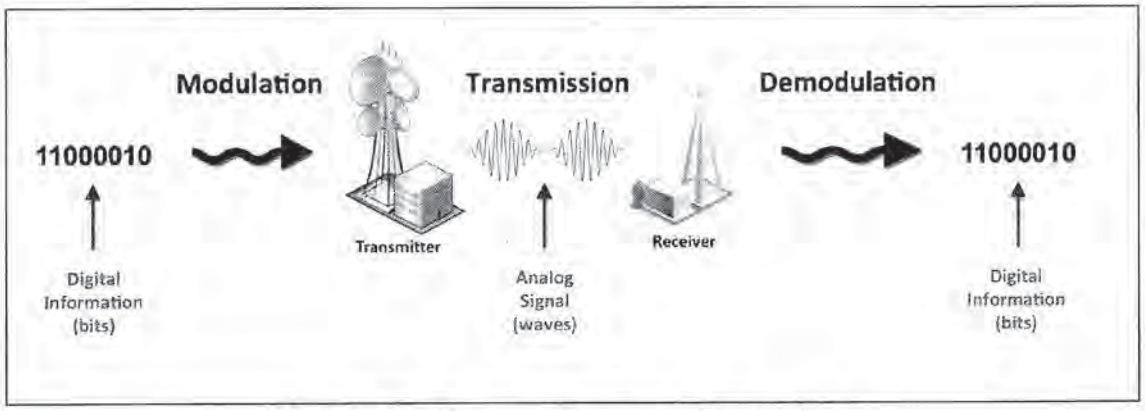
34. The four patents-in-suit, which share a common specification, relate to the field of error-correcting codes. Below I provide a brief introduction to channel coding and error-correcting codes, and highlight a few of the developments in the field that are relevant to the asserted patents. Also, attached as Appendix A is a mathematical description of some properties of error-correcting codes.

[4] The Claim Construction Order dated August 6, 2014 expounded on the plain meaning of "repeat." For example, the order said the "plain meaning of 'repeat' requires the creation of new bits corresponding to or reflecting the value of the original bits. In other words, repeating a bit with the value 0 will produce another bit with the value 0. The Court will refer to this concept as duplication" (Claim Construction Order dated August 6, 2014, p. 10).

A. Error-Correcting Codes in General

35. Most computing devices and other digital electronics use bits to represent information. A bit is a binary unit of information that may have one of two values: 1 or 0. Any type of information, including, *e.g.*, text, music, images and video information, can be represented digitally as a collection of bits.

36. When transmitting binary information over an analog communication channel, the data bits representing the information to be communicated (also called "information bits" or "source bits") are converted into an analog signal that can be transmitted over the channel. This process is called *modulation*. The transmitted signal is then received by a receiving device and converted back into binary form. This process, in which a received analog waveform is converted into bits, is called *demodulation*. The steps of modulation and demodulation are shown in the figure below:



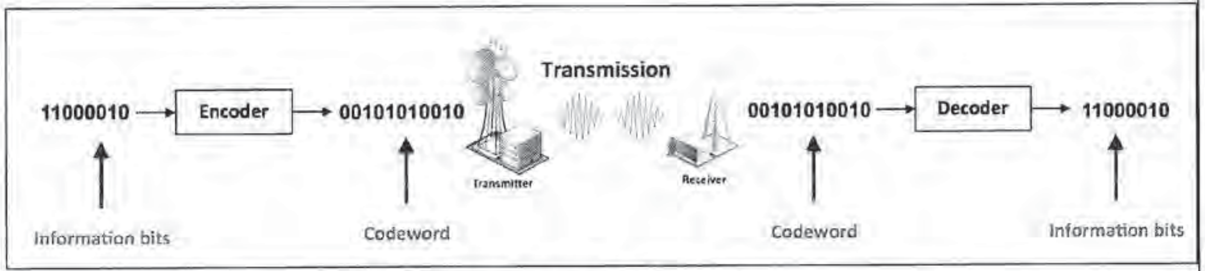
Modulation, Transmission, and Demodulation

37. Transmission over physical channels is never 100% reliable. The transmitted signal can be corrupted during transmission by "noise" caused by, *e.g.*, obstacles obstructing the signal path, interference from other signals, or electrical/magnetic disturbances. Noise can cause bits to "flip" during transmission: for example, because of noise, a bit that was transmitted as a 1 can be corrupted during transmission and demodulated as 0, and vice versa.

38. Error-correcting codes were developed to combat such transmission errors. Using the bits representing the information to be communicated (called "information bits", "data bits" or "source bits") an error-correcting code generates "parity bits" that allow the receiver to verify that the bits were transmitted correctly, and to correct transmission errors that may have occurred.

39. Bits are encoded by an *encoder*, which receives a sequence of information bits as input, generates parity bits based on the information bits according to a particular encoding algorithm, and outputs a sequence of encoded bits (or data elements) called a *codeword*. The codeword produced by the encoder is then modulated and transmitted as an analog signal.

40. At the receiver the signal is received, demodulated and passed to the *decoder*, which uses a decoding algorithm to recover the original codeword and the original information bits.



Encoding and Decoding

41. Error-correcting codes work by adding redundant information to the original message. Due to redundancy, the information represented by a given information bit is spread across multiple bits of the codeword. Thus, even if one of those bits is flipped during transmission, the original information bit can still be recovered from the others.

42. As a simple example, consider an encoding scheme, which I will call "repeat-three," that outputs three copies of each information bit. In this scheme, the information bits "1 0 1" would be encoded as "111 000 111." Upon receipt,

the decoder converts instances of "111" into "1" and instances of "000" into "0" to produce the decoded bits "1 0 1," which match the original information bits.

43. Suppose a bit is flipped during transmission, changing "000" to "0**1**0." The decoder will be able to detect that there was a transmission error, because "010" is not a valid "repeat-three" codeword. Using a "majority vote" rule, the decoder can infer that the original information bit was a 0, correcting the transmission error. Thus, due to the redundancy incorporated into the codeword, no information was lost due to the transmission error.

44. Error-correcting codes may be either *systematic* or *non-systematic*. In a systematic code, both the parity bits and the original information bits are included in the codeword. In a non-systematic code, the encoded data only includes the parity bits.

45. Systematic and non-systematic codes had been known in the art for decades prior to May 18, 2000, the claimed priority date of the patents-in-suit (*see*, *e.g.*, Wicker Dep. at 77:15-20; *see also, e.g.*, Divsalar Dep. at pp. 66-67).

B. Coding Rate

46. Many error-correcting codes encode information bits in groups, or *blocks* of fixed length *n*. An encoder receives an *k*-bit block of information bits as input, and produces a corresponding *n*-bit codeword. The ratio *k/n* is called the *rate* of the code. Because the codeword generally includes redundant information, *n* is generally greater than *k*, and the rate *k/n* of an error-correcting code is generally less than one.

C. Performance of Error-Correcting Codes

47. The effectiveness of an error-correcting code may be measured using a variety of metrics.

48. One tool used to assess the performance of a code is its *bit-error rate* (BER). The BER is defined as the number of corrupted information bits divided by the total number of information bits during a particular time interval. For example, if a decoder outputs 1000 bits in a given time period, and 10 of those bits are corrupted (*i.e.*, they differ from the information bits originally received by the encoder), then the BER of the code during that time period is (10 bit errors) / (1000 total bits) = 0.01 or 1%.[5]

49. The BER of a coded transmission depends on the amount of noise that is present in the communication channel, the strength of the transmitted signal (*i.e.*, the power that is used to transmit the modulated waveform), and the performance of the error-correcting code. An increase in noise tends to increase the error rate and an increase in signal strength tends to decrease the error rate. The ratio of the signal strength to the noise, called the "signal-to-noise ratio," is often used to characterize the channel over which the encoded signal is transmitted. The signal-to-noise ratio can be expressed mathematically as E_b/N_0, in which E_b is the amount of energy used to transmit each bit of the signal, and N_0 is the density of the noise on the channel.[6] The BER of an error-correcting code is often measured for multiple values of E_b/N_0 to determine how the code performs under various channel conditions.

50. Error-correcting codes may also be assessed based on their computational complexity. The complexity of a code is a rough estimate of how many calculations are required for the encoder to generate the encoded parity bits and how many calculations are required for the decoder to reconstruct the information

[5] Note that as used herein, BER refers to the *information* BER, which measures the percentage of bits that remain incorrect after decoding. This is not to be confused with the *transmission* BER, which measures the percentage of bits that are incorrect when they are received by the decoder.
[6] More precisely, E_b/N_0 is the *normalized* signal-to-noise ratio. It is a dimensionless quantity that does not depend on the particular units used to measure the strength of the signal and the quantity of noise on the channel.

bits from the parity bits. If a code is too complex, it may be impractical to build encoders/decoders that are fast enough to use it.

D. LDPC Codes, Convolutional Codes, Turbocodes, and Repeat-Accumulate codes

51. In 1963, Robert Gallager described a set of error correcting codes called Low Density Parity Check ("LDPC") codes. Gallager described how LDPC codes provide one method of generating parity bits from information bits using a matrix populated with mostly 0s and relatively few 1s, and he described how decoding could be performed using an iterative "message passing" decoding algorithm, as described below.[7]

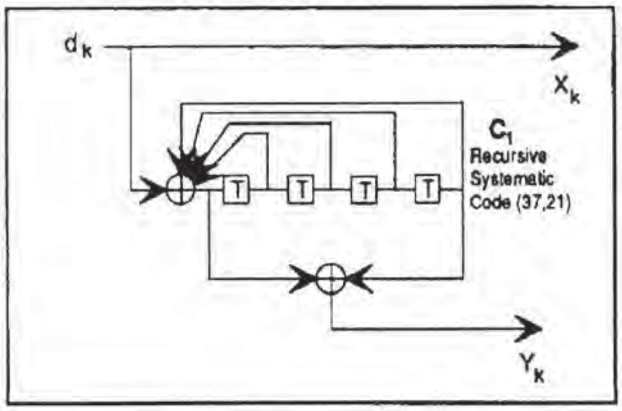
52. Gallager's work was largely ignored over the following decades, as researchers continued to discover other algorithms for calculating parity bits. These algorithms included, for example, convolutional encoding (see below) with Viterbi decoding and cyclic code encoding with bounded distance decoding. In many cases these new codes could be decoded using low-complexity decoding algorithms.

53. In 1993, researchers discovered "turbocodes," a class of error-correcting codes capable of transmitting information at a rate close to the Shannon Limit – the maximum rate at which information can be transmitted over a channel. Turbocodes make use of "convolutional codes", which were described in the 1960's and were widely used in telephone modems in the 1980's and 1990's. A convolutional code is a type of error-correcting code that generates parity bits by processing the information bits in order. The convolutional code contains a "memory bank" in the form of a short sequence of bits, e.g., 4 bits. When an information bit d_k is processed, the memory bits s_1, s_2, s_3, s_4 are combined with the information bit to produce a new memory bit and the remaining memory bits are

[7] Gallager, R., *Low-Density Parity-Check Codes* (Monograph, M.I.T. Press, 1963).

"shifted", so that the last memory bit is discarded. For example, the new memory bit $s_1^{'}$ could be computed by $s_1^{'} = d_k + s_1 + s_2 + s_3 + s_4$ *modulo* 2, and the other memory bits would be $s_2^{'} = s_1$, $s_3^{'} = s_2$, and $s_4^{'} = s_3$. What does "modulo 2" mean? If the sum of the bits is even, then the sum modulo 2 is zero, whereas if the sum of the bits is odd, then the sum modulo 2 is one. Note that s_4 has been discarded. When an information bit is being processed, a parity bit is also generated. The parity bit y_k is a combination of the new memory bit and the entire set of current memory bits, for example, $y_k = s_1^{'} + s_4$ *modulo* 2. The combinations used to determine the new memory bit and the parity bit need not include all of the bits, e.g., the above example uses all bits to compute the new memory bit, but only $s_1^{'}$ and s_4 when computing the parity bit. If a particular bit is used in a combination, we say there is a "tap" connected to that bit. In the example, the parity bit is connected by a tap to $s_1^{'}$ and another tap to s_4. The set of taps for the memory bit and the set of taps for the parity bit are fixed when processing information bits and they completely characterize the convolutional code. In a "systematic" convolutional code, the information bits are also transmitted across the channel, in addition to the parity bits. Some parity bits and/or some information bits may be punctured so as to adjust the rate of the convolutional code (the number of information bits processed divided by the number of bits transmitted). If the new memory bit doesn't have any taps to any memory bits, the code is called "non-recursive" and otherwise it is called "recursive", alluding to the fact that the new memory bit depends on the bits in the old memory. Using the above example, the figure below shows how a recursive convolutional code is depicted, where a circle with a plus inside indicates summation *modulo* 2 and a box with a T inside indicates a memory location (figure modified from [8]).

[8] Claude Berrou et al., *Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes*, 2 IEEE International Conference on Communications, ICC '93 Geneva. Technical



54. Convolutional codes are usually decoded using the "Viterbi algorithm" or the "BCJR algorithm". These algorithms can be viewed as iterative "message passing" decoding algorithms, if we represent the convolutional code using a "Tanner graph" or a "factor graph", as described below.

55. The main drawback of convolutional codes is that they only produce local redundancy in the output stream. They do not perform well when the channel introduces errors that are nearby. Turbocodes overcome this deficiency by encoding the input bits twice. The input bits are fed to a convolutional encoder in their normal order, and they are also reordered by an interleaver and the reordered bits are encoded by a second convolutional encoder. Using a turbocode, a small number of errors will not result in loss of information unless the errors happen to fall close together in both the original data stream and in the permuted data stream, which is unlikely.

56. A standard turbocoder encodes a sequence of information bits using two convolutional coders. The information bits are passed to the first convolutional coder in their original order. At the same time, a copy of the information bits

Program, Conference Record 1064 (1993); '032 patent, 1:29-56.

permuted by an interleaver is passed to the second convolutional coder. The figure below shows the structure of a typical turbocoder.[9]

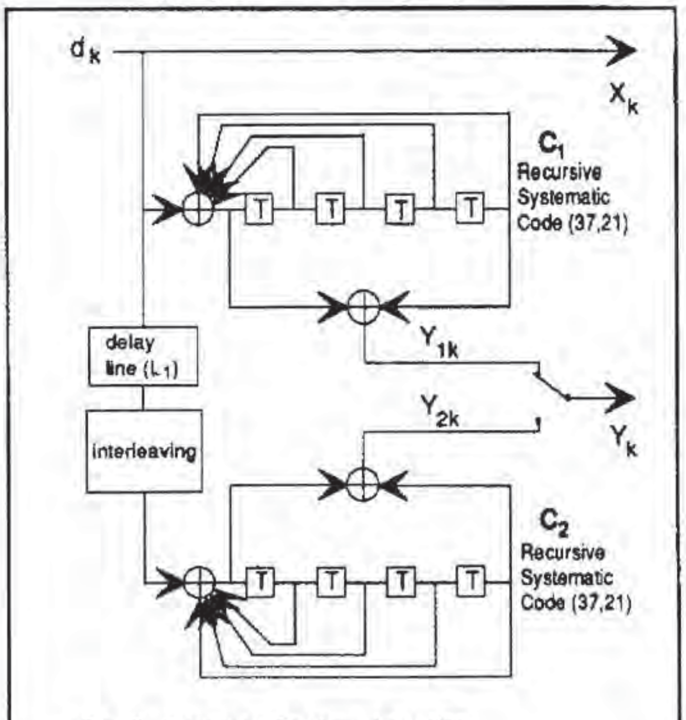


Fig. 2 Recursive Systematic codes with parallel concatenation.

57. In 1995, David J. C. MacKay rediscovered Gallager's work from 1963 relating to low-density parity-check (LDPC) codes and demonstrated that they have performance comparable to that of turbocodes.[10] Turbocodes and LDPC codes have some common characteristics: both codes use pseudo-random permutations to spread out redundancy, and both use iterative "message passing" decoding algorithms.

[9] Claude Berrou et al., *Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes*, 2 IEEE International Conference on Communications, ICC '93 Geneva. Technical Program, Conference Record 1064 (1993); '032 patent, 1:29-56.
[10] MacKay, D. J. C, and Neal, R. M. "Near Shannon Limit Performance of Low Density Parity Check Codes," *Electronics Letters*, vol. 32, pp. 1645-1646 (1996).

58. In 1995 and 1996, researchers began to explore "concatenated" convolutional codes.[11] While turbocodes use two convolutional coders connected in parallel, concatenated convolutional codes use two convolutional coders connected in series: the information bits are encoded by a first encoder, the output of the first encoder is interleaved, and the interleaved sequence is encoded by a second convolutional code. In such codes, the first and second encoders are often called the "outer coder" and the "inner coder," respectively.

59. In 1998, researchers developed "repeat-accumulate," or "RA codes" by simplifying the principles underlying turbocodes.[12] In RA codes, the information bits are first passed to a repeater that repeats (*i.e.*, duplicates) the information bits and outputs a stream of repeated bits (the encoder described above in the context of the "repeat three" coding scheme is one example of a repeater). The repeated bits are then passed through an interleaver, which scrambles their order, and then to an accumulator, where they are "accumulated" to form the parity bits, which are transmitted across the channel.

60. The accumulation operation is a running sum process whereby each input bit is added to the previous input bits to produce a sequence of running sums, each of which represents the sum of all input bits yet received. More formally, if an accumulator receives a sequence of input bits $i_1, i_2, i_3, \ldots i_n$, it will produce output bits $o_1, o_2, o_3, \ldots o_m$ such that:[13]

[11] Benedetto, S. et al., *Serial Concatenation of Block and Convolutional Codes*, 32.10 Electronics Letters 887-888 (1996).

[12] Divsalar, D. et al., "Coding Theorems for Turbo-like Codes," *Proc. 36th Allerton Conf. on Comm., Control and Computing*, 201 (Sept. 1998).

[13] Here I use the $\oplus$ symbol to denote modulo-2 addition.

$$\begin{aligned}
o_1 &= i_1 \\
o_2 &= i_1 \oplus i_2 \\
o_3 &= i_1 \oplus i_2 \oplus i_3 \\
&\vdots \\
o_n &= i_1 \oplus i_2 \oplus i_3 \oplus \cdots \oplus i_n
\end{aligned}$$

61. The accumulation operation can also be described as a recursive operation in which each output bit is the sum of the previous output bit and the current input bit:

$$\begin{aligned}
o_1 &= i_1 \\
o_2 &= o_1 \oplus i_2 \\
o_3 &= o_2 \oplus i_3 \\
&\vdots \\
o_n &= o_{n-1} \oplus i_n
\end{aligned}$$

62. As this recursive formulation shows, each accumulated bit can be calculated by performing a single modulo-2 addition operation. This relatively low computational complexity is one of the benefits of accumulate codes. In particular, it allows accumulate codes to be encoded quickly and cheaply.

63. Repetition and accumulation were well known in the art by May 18, 2000 and by March 7, 2000, the claimed priority date and the claimed conception date, respectively, of the patents-in-suit (*see*, *e.g.*, Wicker Dep. at 66:18-67:11, Jin Dep. at 67:8-23, 122:7-13).

E. Irregularity

64. A *regular code* is a systematic code that corresponds to a Tanner graph in which each information node is connected to the same number of check nodes, or a nonsystematic code that corresponds to a Tanner graph in which each parity node

is connected to the same number of check nodes.[14] By contrast, an *irregular code* is a systematic code that corresponds to a Tanner graph in which some information nodes are connected to more check nodes than others, or a nonsystematic code that corresponds to a Tanner graph in which some parity nodes are connected to more check nodes than others. The concepts of *regular* and *irregular* need not be expressed with reference to Tanner graphs, but it is convenient to do so.

65. Irregular LDPC codes were first introduced in a 1997 paper by Luby et al.[15] The paper showed that irregular codes perform better than regular ones on certain types of noisy channels. At the time, this paper was widely read by coding theorists, and gave rise to several lines of research into irregular error-correcting codes. For example, in my own paper titled "Irregular Turbocodes," presented at the 1999 Allerton Conference on Communications, Control, and Computing, I applied the concept of irregularity to turbocodes by explaining how to construct irregular turbocodes in which some information bits connect to more check nodes than others. My experimental results demonstrated that these irregular turbocodes perform better than the regular turbocodes that were known in the art.

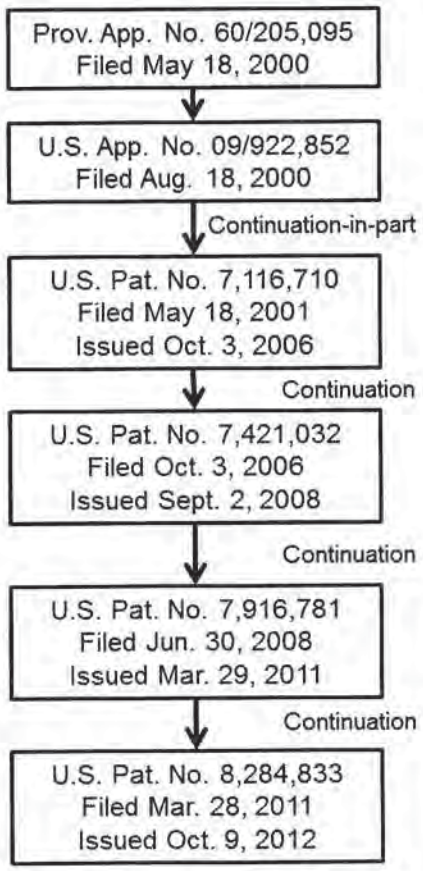
66. By May 18, 2000 and by March 7, 2000, the claimed priority date and the claimed conception date, respectively, of the patents-in-suit, it was known to those with ordinary skill in the art that the performance of any type of error-correcting code could be improved by adding irregularity (*see*, *e.g.*, Wicker Dep. at 232:6-233:8). For example, on Dec. 8, 1999, I wrote to Dr. Divsalar, the lead author on the paper "Coding Theorems for 'Turbo-Like' Codes" discussed in this report, suggesting that the RA codes that he and Dr. Robert McEliece had been working on should be made irregular (*see* CALTECH000024021).

[14] For a more complete discussion of Tanner graphs, *see generally* Appendix A.
[15] Luby, M. et al., "Practical Loss-Resilient Codes," *STOC '97* (1997).

III. THE PATENTS-IN-SUIT

A. Summary of the Specification.

67. I have been informed that the patents-in-suit share a common specification and that they were filed as a sequence of continuation applications as shown in the diagram below.



68. The specification, which is common to the four patents-in-suit, is generally directed to irregular RA codes (or "IRA" codes). Figure 2 of the specification, reproduced below, shows the structure of an IRA encoder:

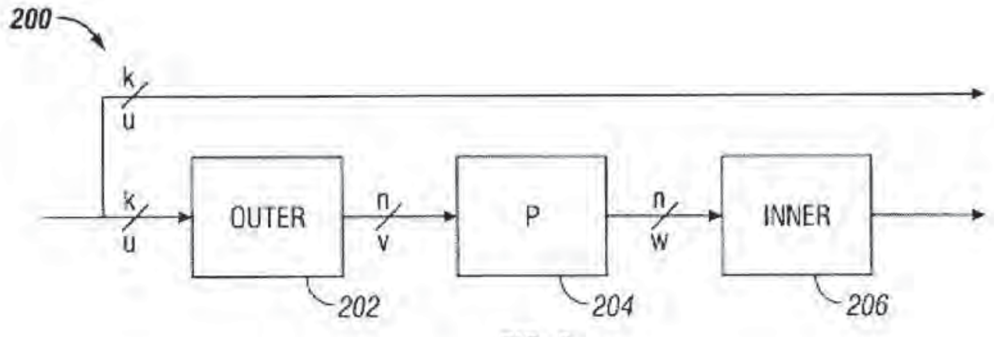


FIG. 2

69. Explaining this figure, the patents describe encoding data using an outer coder 202 connected to an inner coder 206 via an interleaver 204 (labeled "P") ('710 patent at 2:33-40).

70. Outer coder 202 receives a block of information bits and duplicates each of the bits in the block a given number of times, producing a sequence of repeated bits at its output (*id.* at 2:50-52). The outer coder repeats bits irregularly – *i.e.*, it outputs more duplicates of some information bits than others (*id.* at 2:48-50).

71. The repeated bits are passed to an interleaver 204, where they are scrambled (*id.* at 3:18-22). The scrambled bits are then passed to the inner coder 206, where they are accumulated to form parity bits (*id.* at 2:65-67; 2:33-38). According to the specification:

> Such an accumulator may be considered a block coder whose input block $[x_1, \ldots, x_n]$ and output block $[y_1, \ldots, y_n]$ are related by the formula
>
> $y_1=x_1$
>
> $y_2=x_1\oplus x_2$
>
> $y_3=x_1\oplus x_2\oplus x_3$
>
> $y_n=x_1\oplus x_2\oplus x_3\oplus\ldots\oplus x_n.$
>
> (*id.* at 3:2-10).

72. The patent specification teaches both systematic and non-systematic codes. In a systematic code, the encoder outputs a copy of the information bits in addition

to the parity bits output by inner coder 206 (the systematic output is represented in Fig. 2. as an arrow running toward the right along the top of the figure).

73. I discuss each of the patents individually below. However, I note here that Caltech has characterized all four of the asserted patents as being directed to IRA codes.[16]

B. '710 Patent

i) <u>Claims</u>

74. The '710 patent includes 33 claims, of which claims 1, 11, 15, and 25 are independent. Independent claims 1 and 11 are directed to methods of encoding a signal that include "first encoding" and "second encoding" steps. Independent claim 15 is directed to a "coder" for encoding bits that includes a "first coder" and a "second coder." Claim 25 is directed to a "coding system" that also encodes bits using a first and second coder, and further includes a decoder for decoding the encoded bits. I understand that Caltech asserts claims 1, 4, 6, 15, 20, and 22 in this case.

ii) <u>Prosecution History</u>

a) <u>*First Office Action: September 3, 2004*</u>

75. The patent office issued a first office action rejecting some of the claims under 35 U.S.C. § 102 as anticipated by U.S. Patent No. 6,014,411 (to Wang) and under 35 U.S.C. § 103 as obvious over Wang in view of Wiberg et al., "Codes and Iterative Decoding on General Graphs," *1995 Intl. Symposium on Information Theory*, Sep. 1995, p. 506.

[16] *See, e.g.*, Plaintiff's Technology Tutorial (Dkt. No. 85), p. 1 (which states that "[a]ll of the patents in suit relate to a novel error correction technique known as IRA codes").

b) *<u>Response: November 24, 2004</u>*

76. In response, the applicant argued that the rejected claims are not anticipated or obvious over the cited art because they all require that bits be repeated "irregularly" or "a different number of times" during the first encoding step, while Wang teaches repeating bits "the same number of times, i.e., regularly" (Response dated Nov. 24, 2004 at 11).

c) *<u>Second Office Action: March 4, 2005</u>*

77. The patent office issued a second office action allowing some claims and rejecting others. In particular, the examiner allowed claim 1 in response to the applicant's arguments. The examiner also rejected independent claims 15 and 24, under 35 U.S.C. § 102 as anticipated by U.S. Patent No. 6,396,423 (to Laumen et al.). The patent office also rejected several dependent claims under 35 U.S.C. § 103 as obvious over Laumen alone.

d) *<u>Response: May 5, 2005</u>*

78. In response, the applicant attempted to overcome the examiner's rejections by amending claims 15 and 24 to require that the second coder encode bits at a rate "within 50% of one" (previously, the claims had recited a rate "close to one") (Response dated May 5, 2005 at 7-8). In the same amendment, the applicant added new claims 32-35.

e) *<u>Third Office Action: July 21, 2005</u>*

79. The patent office issued a third office action maintaining its previous rejections over Laumen, noting that Laumen teaches a transmission rate of 1/2, and 1/2 is "within 50% of one" (Office Action dated Jul. 21, 2005 at 4).

f) *<u>Response: October 21, 2005</u>*

80. To overcome the examiner's rejection, the applicant canceled claims 32 and 34 and incorporated their subject matter into claims 15 and 24, respectively. As

amended, claims 15 and 24 require that the second coder encode bits at a rate "within 10% of one" (Response dated Oct. 21, 2005 at 9).

C. '032 Patent

i) Claims

81. The '032 patent includes 23 claims, of which claims 1, 11, and 18 are independent. Independent claim 1 is directed to a method that comprises generating a sequence of parity bits from a collection of message bits in accordance with particular mathematical formulae, and making the parity bits available for transmission. Independent claim 11 is directed to an encoder that generates a sequence of parity bits from a collection of message bits in accordance with a particular Tanner Graph. Independent claim 18 is directed to a device for decoding a data stream that has been encoded in accordance with the same Tanner Graph. I understand that Caltech asserts claims 1, 18, 19, and 22 in this case.

ii) Prosecution History

a) First Office Action: September 6, 2007

82. The patent examiner initially allowed pending claims 1-17 and rejected independent claim 18 and dependent claims 19-24 under 35 U.S.C. § 103 as obvious over U.S. Patent No. 5,530,707 (to Lin) in view of U.S. Patent No. 6,859,906 (to Hammons et al.).

b) Response: Feb 4, 2008

83. To overcome the examiner's rejection, the applicant canceled claim 20 and incorporated its subject matter into independent claim 18. The amendment further limited claim 18 to require that the message passing decoder of claim 18 be configured to decode a data stream that has been encoded in accordance with a particular Tanner graph.

D. ’781 Patent

i) <u>Claims</u>

84. The ’781 patent includes 22 claims, of which claims 1, 13, 19, 20, and 21 are independent. Independent claim 1 is directed to a two-step process for encoding a signal, where the first encoding step involves a linear transform operation and the second involves an accumulation operation. Independent claims 13 and 19 are directed to methods of encoding a signal that generate codewords by summing information bits and accumulating the resulting sums. Independent claims 20 and 21 are directed to methods that involve summing information bits and parity bits to generate a portion of an encoded signal. I understand that Caltech asserts claims 16 and 19 in this case.

ii) <u>Prosecution History</u>

a) *<u>First Office Action: October 28, 2010</u>*

85. The patent examiner issued a first office action allowing some claims but rejecting claims 13-17 and 20 as anticipated by U.S. Patent 5,181,207 (to Chapman, et. al.) and requiring applicants to clarify the term “irregular,” as it appeared in claims 9 and 23.

b) *<u>Response: January 27, 2011</u>*

86. To overcome the examiner’s rejection, the applicant canceled claim 21 and incorporated its subject matter into independent claim 13. As amended, claim 13 requires that “the information bits appear in a variable number of subsets” (Response dated Jan. 27, 2011 at 4).

87. In accompanying remarks, applicants disagreed with the examiner’s statement that the term “irregular” was unclear, stating that “[i]t is believed that <u>the meaning of the term "irregular" in the claims is clear and *is well known in the art of computer coding technology*</u>” (*id.* at 7) (emphasis added). However, to overcome

the examiner's rejection, the applicant amended claims 9 and 23 to remove the word "irregular," replacing it with the requirement that the information bits appear "in a variable number of subsets" (*id.* at 3, 6).

E. '833 Patent

i) Claims

88. The '833 patent includes 14 claims, of which claims 1and 8 are independent. Independent claims 1 and 8 are directed to an apparatus and a method, respectively, for encoding information bits that are stored in a first set of memory locations by combining information bits with parity bits that are stored in a second set of memory locations, and accumulating the bits in the second set of memory locations. Both claims require that at least two of the first set of memory locations be read "different times from one another."[17] I understand that Caltech asserts claims 1, 2, 4, and 8 in this case.

ii) Prosecution History

89. After the examiner had allowed all pending claims in the application, the applicant attempted to amend claims 1 and 8 as follows: "wherein a ~~total number of indices~~ <u>two or more memory locations</u> of the first set of memory locations <u>are read by the permutation module different times from one another</u> ~~represents a variable number~~" (Amendment dated May 7, 2012).

90. The examiner did not enter these amendments after allowance because they changed the scope of the claims that had already been allowed. The applicant subsequently filed a request for continued examination, after which the examiner allowed the claims as amended.

[17] As noted above, the parties have agreed that this claim term requires memory locations to be read a different ***number*** of times from one another.

IV. <u>SUMMARY OF THE PRIOR ART</u>

A. Divsalar

91. D. Divsalar, H. Jin, and R. J. McEliece, "Coding theorems for "turbo-like" codes," *Proc. 36th Allerton Conf. on Comm., Control and Computing*, Allerton, Illinois, pp. 201-210 ("Divsalar") was published in Sept. 1998, about 1.5 years before the filing of the provisional application to which the patents-in-suit claim priority, and I have been informed that Divsalar qualifies as prior art to all four of the patents-in-suit.

92. Divsalar teaches "repeat and accumulate" codes, which it describes as "a simple class of rate $1/q$ serially concatenated codes where the outer code is a q-fold repetition code and the inner code is a rate 1 convolutional code with transfer function $1/(1 + D)$" (Divsalar at 1). Fig. 3 of Divsalar, reproduced below, shows an encoder for a repeat-accumulate code with rate N/qN:

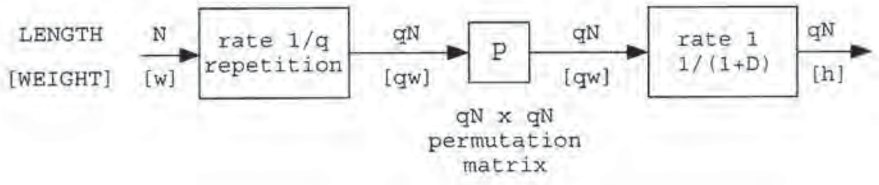


Figure 3. Encoder for a (qN, N) repeat and accumulate code. The numbers above the input-output lines indicate the length of the corresponding block, and those below the lines indicate the weight of the block.

93. A block of N information bits enters the coder at the left side of the figure and is provided to the repeater (labeled "rate 1/q repetition") (*see id.* at 5). The repeater duplicates each of the N information bits q times and outputs the resulting $N \times q$ repeated bits, which are then "scrambled by an interleaver of size qN" (*id.*, referring to the box labeled "P"). The scrambled bits are "then encoded by a rate 1 *accumulator*" (*id.*, emphasis in original; *see also* Divsalar Tr. at pp. 59-63, 68-69).

94. Divsalar describes the accumulator as follows:

> [W]e prefer to think of [the accumulator] as a block coder whose input block [$x_1, \ldots, x_n$] and output block [$y_1, \ldots, y_n$] are related by the formula
>
> $y_1 = x_1$
> $y_2 = x_1 + x_2$
> $y_3 = x_1 + x_2 + x_3$
> $y_n = x_1 + x_2 + x_3 + \ldots + x_n$.

(*id.* at 5). The plus signs ("+") in Divsalar's formula represent modulo-2, or exclusive-OR, addition (*see id.*; *see also* Divsalar Tr. 69:10-16).

95. Divsalar uses repeat-accumulate codes to prove a conjecture regarding the interleaver gain exponent (IGE), which is a numerical parameter that estimates the rate at which the word error rate decreases as the block length increases.

96. Divsalar further shows that RA codes have "very good" performance and that they can be efficiently decoded using a "message passing decoding algorithm" (*id.* at 9-10).

97. Divsalar teaches that turbocodes, serially concatenated convolutional codes and RA codes can all be viewed as "turbo-like" codes: "We call these systems "turbo-like" codes and they include as special cases both the classical turbo codes and the serial concatentation of interleaved convolutional codes" (Divsalar Abstract) and "In Section 5, we define a special class of turbo-like codes, the repeat-and-accumulate codes, and prove the IGE conjecture for them" (Divsalar at 1). More specifically, RA codes can be viewed as turbocodes, in which the information bits are punctured, or truncated, none of the parity bits are punctured, and the convolutional code is an accumulator. "The accumulator can be viewed as a truncated rate-1 recursive convolutional encoder with transfer function 1/(1 + D)" (Divslar at 5). Divsalar also makes use of the fact that RA codes can be viewed as turbocodes to explain the decoder: "But an important feature of turbo-like codes is the availability of a simple iterative, message passing decoding algorithm that approximates ML decoding. We wrote a computer program to implement this

'turbo-like' decoding for RA codes with q = 3 (rate 1/3) and q = 4 (rate 1/4), and the results are shown in Figure 5" (Divsalar at 9).

98. As explained further below, Divsalar teaches all but one aspect of an IRA code: irregularity (the "I" in ***I***rregular ***R***epeat-***A***ccumulate). That is, Divsalar teaches regular repeat-accumulate (RA) codes rather than irregular repeat-accumulate codes. A single modification to Divsalar – *i.e.*, changing the repeat to being irregular instead of regular – would result in the IRA codes that Caltech claims to have invented. I also explain below why it would have been obvious to one of ordinary skill before the Caltech patents were filed (and before Caltech's claimed conception date) to add irregularity to the repeat-accumulate codes of Divsalar, resulting in the irregular repeat-accumulate codes to which the patents-in-suit are directed.

B. Luby

99. U.S. Patent No. 6,081,909 to Luby et al. ("Luby"), titled "Irregularly graphed encoding technique," was filed Nov. 6, 1997, about 2.5 years before the filing of the provisional application to which the patents-in-suit claim priority, and I have been informed that Luby qualifies as prior art to all four of the patents-in-suit.

100. The Luby patent mirrors the teachings of Luby's seminal paper that I described above, in which the concept of irregular error-correcting codes was first introduced. Specifically, Luby teaches "a technique for creating loss resilient and error correcting codes having irregular graphing between the message data and the redundant data" (Luby at 1:5-10). "Irregular graphing" refers to codes with Tanner graphs in which some information nodes are connected to more check nodes than others (*see*, *e.g.*, *id.* at 3:27-29, stating that "different numbers of first edges are associated with the data items").

101. A Tanner graph corresponding to an irregular code is shown in Fig. 17 of Luby, reproduced below:

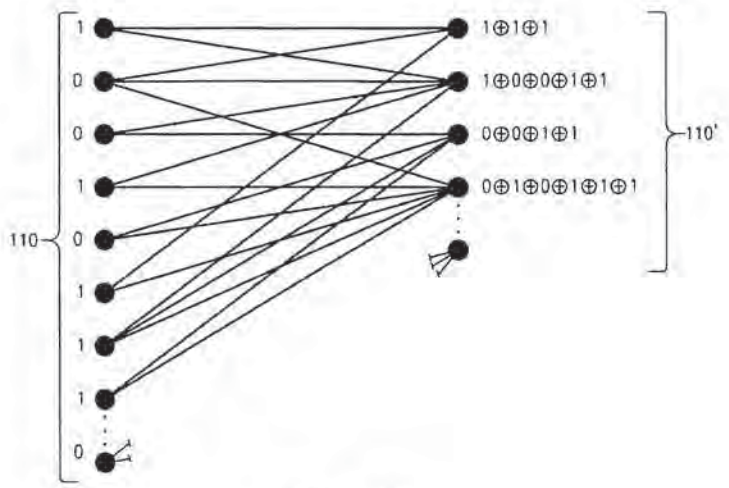


FIG. 17

102. In this figure, the circles on the left represent information bits to be encoded and the circles on the right represent parity checks computed for these information bits. Each parity check on the right is computed by summing together (modulo 2) all of the information bits connected to that parity check by an edge in the graph (*see id.* at 17:64-67).[18]

103. As the figure shows, some information nodes on the left contribute to three parity checks on the right, while others contribute to two (*i.e.*, all nodes on the left which are connected to two lines, such as the top node, contribute to two parity checks and all nodes on the left which are connected to three lines, such as the second node from the top, contribute to three parity checks). An encoding scheme with a Tanner graph in which some information nodes are connected to more check nodes than others is the defining characteristic of an irregular code.

[18] I explain what an "edge" is in this context in Appendix A, below.

C. MacKay

104. D. J. C. MacKay, S. T. Wilson, and M. C. Davey, "Comparison of constructions of irregular Gallager codes," *IEEE Trans. Commun.*, Vol. 47, No. 10, pp. 1449-1454 ("MacKay") was published in Oct. 1999, about six months before the filing of the provisional application to which the patents-in-suit claim priority, and I have been informed that MacKay qualifies as prior art to all four of the patents-in-suit.

105. MacKay is motivated by "[t]he excellent performance of irregular Gallager codes," and explores "ways of further enhancing these codes" (MacKay at 1459). In particular, MacKay investigates the constructions of both regular and irregular Gallager codes with encoding algorithms that have low computational complexity.

D. Ping

106. L. Ping, W. K. Leung, N. Phamdo, "Low Density Parity Check Codes with Semi-random Parity Check Matrix." *Electron. Letters*, Vol. 35, No. 1, pp. 38-39 ("Ping") was published in Jan. 1999, more than a year before the filing of the provisional application to which the patents-in-suit claim priority, and I have been informed that Ping qualifies as prior art to all four of the patents-in-suit.

107. Ping teaches constructing LDPC codes that can be encoded in two stages. In the first encoding stage, a generator matrix is applied to a sequence of information bits to produce sums of information bits. In the second stage, the sums of information bits are accumulated recursively to generate the parity bits (*see* Ping at 38).

108. Ping's code can be described as an LDPC code with two components: an outer coder that is an LDGM coder followed by an inner coder that is an

accumulator. Thus Ping teaches LDPC codes that are also accumulate codes.[19] I understand that the codes Caltech has accused of infringement, *i.e.*, the DVB-S2 codes, can also be encoded using LDPC + accumulate coders. One difference between Ping and the accused codes is that Ping's LDPC code is regular whereas in the accused DVB-S2 codes, the LDPC code is irregular. As explained below, it was obvious before Caltech's alleged invention to make codes irregular, e.g., because it was known that doing so would improve their performance. In particular, it was obvious before Caltech's alleged invention to make Ping's LDPC code irregular. Therefore, if Caltech establishes that its claims cover the accused DVB-S2 codes, then those claims would be invalid in view of Ping and the art that rendered it obvious to make Ping's LDPC code irregular, e.g. Luby, MacKay and Frey99.

E. Frey99

109. Frey, B. J. and MacKay, D. J. C., "Irregular Turbocodes," *Proc. 37th Allerton Conf. on Comm., Control and Computing*, Monticello, Illinois ("Frey99") was published on or before March 20, 2000, which is before the filing of the provisional application to which the patents-in-suit claim priority, and I have been informed that Frey99 qualifies as prior art to all four of the patents-in-suit.

110. Frey99 is a paper that I wrote in collaboration with David MacKay. In Frey99, David MacKay and I applied the concept of irregularity to turbocodes by explaining how to construct irregular turbocodes, *i.e.*, turbocodes with Tanner graphs in which some information nodes are connected to more check nodes than others. Our experimental results demonstrated that these irregular turbocodes perform better than the regular turbocodes that were known in the art.

[19] Below I refer to these codes as "LDPC + accumulate" codes.

111. As I explain in Frey99, "an *irregular turbocode* has the form shown in Fig. 2, which is a type 'trellis-constrained code' as described in [7]. We specify a *degree profile*, $f_d \in [0, 1]$, $d \in \{1, 2, \ldots, D\}$. f_d is the fraction of codeword bits that have degree d and D is the maximum degree. Each codeword bit with degree d[20] is repeated d times before being fed into the permuter. Several classes of permuter lead to linear-time encodable codes. In particular, if the bits in the convolutional code are partitioned into 'systematic bits' and 'parity bits', then by connecting each parity bit to a degree 1 codeword bit, we can encode in linear time" (Frey99 at 2).

112. As this passage explains, the irregular turbocodes I described in Frey99 operate by irregularly repeating the information bits, interleaving the repeated bits using a "permute" (*i.e.*, an interleaver), and encoding the permuted bits using a convolutional code. Figure 2 of Frey99, reproduced below, illustrates such an irregular turbocode:

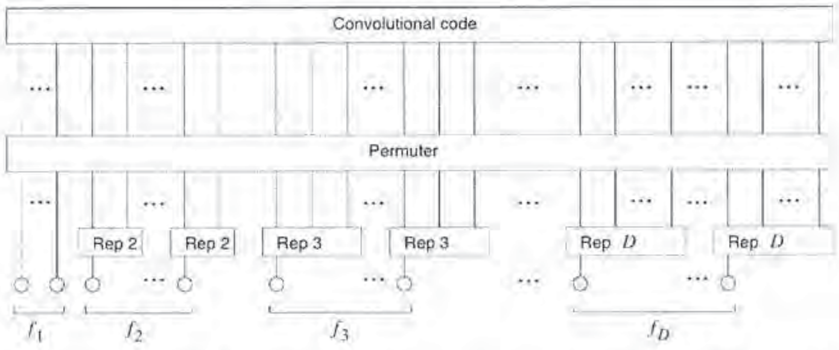


Figure 2: A general *irregular turbocode*. For $d = 1, \ldots, D$, fraction f_d of the codeword bits are repeated d times, permuted and connected to a convolutional code.

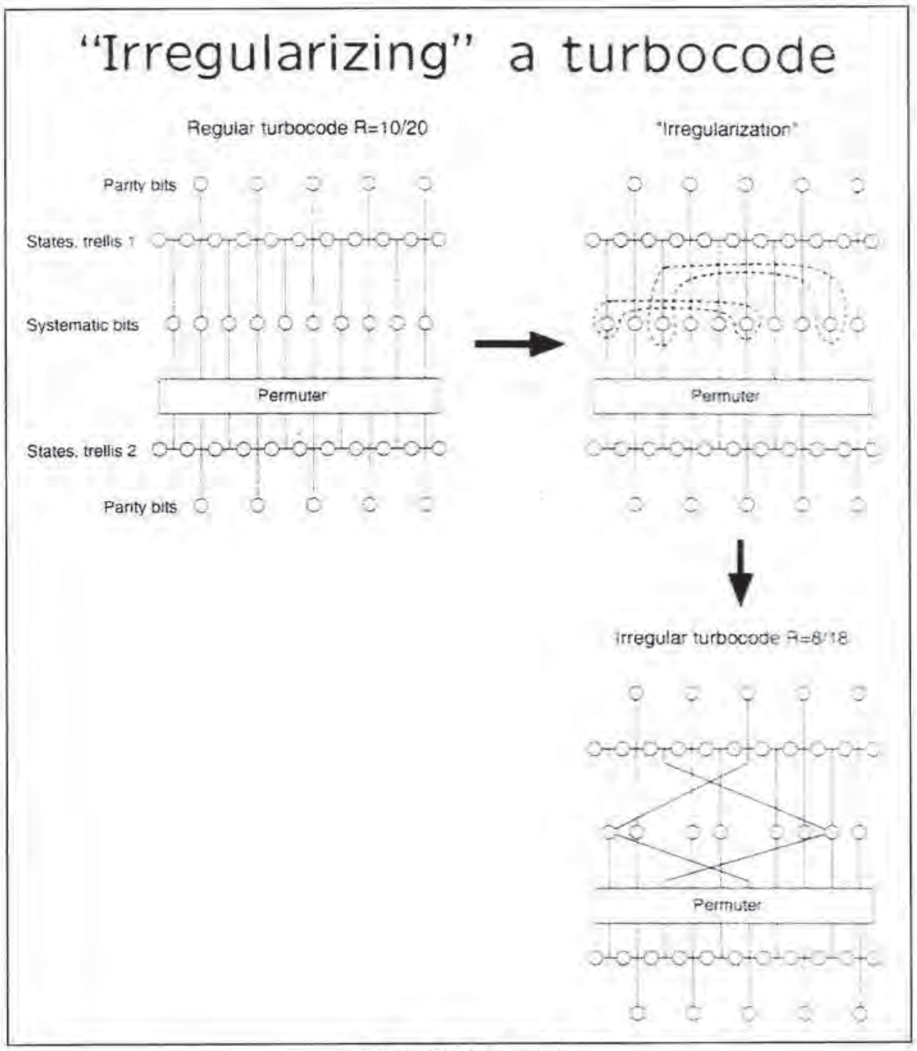
113. In this figure, bits in the subset f_1 are not repeated, bits in the subset f_2 are repeated twice, bits in the subset f_3 are repeated three times, and bits in the subset f_D are repeated D times.

[20] A bit with "degree d" is a bit that contributes to d parity check bits. In Frey99, bits of degree d are repeated d times prior to permutation.

F. Frey Slides

114. I prepared the Frey Slides (titled "Irregular Turbo-Like Codes") in collaboration with David MacKay and presented them at the Allerton Conference in September, 1999. The Frey Slides contain the material upon which the Frey99 paper, published in the Allerton 1999 conference proceedings, is based.

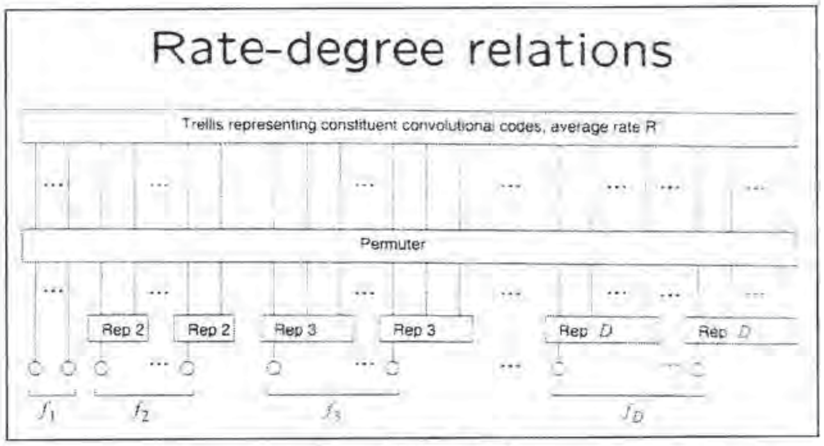
115. In particular, the Frey Slides describe how irregularity can improve code performance and introduce the concept of irregular turbocodes. Using the same procedure described in the Frey99 paper, the Frey Slides show how known, regular turbocodes can be "irregularized," step by step:


"Irregularizing" a turbocode

Frey Slides at 4

In the figure above, a regular turbocode (upper left) is "irregularized" by tying information nodes together (upper right), thereby raising their degree, resulting in an irregular turbocode (lower right).

116. Also, using a diagram identical to Figure 2 of Frey99 (described above) the Frey Slides show how irregular turbocodes can be implemented via irregular repetition:



Frey Slides at 5

117. The Frey Slides also describe selection of degree profiles (*see id.* at 6) and provide details regarding the rate of the resulting convolutional coder and the overall rate of the irregular turbocode (*id.* at 5-8, 13).

118. I understand that Caltech has alleged a date of invention of March 7, 2000. I further understand that Caltech may argue that the Frey99 paper was not published until after its alleged invention date. In the event that the Court finds that the patents-in-suit are entitled to a date of invention that predates the publication of Frey99, and the Frey99 paper is deemed not to be prior art to the patents-in-suit, then the Frey Slides may be substituted for the Frey99 paper in all of the positions explained below. For the purposes of the invalidity opinions set forth in this report, the teachings of Frey99 and the Frey Slides are interchangeable. To illustrate how the Frey Slides may be substituted for Frey99, wherever I cite to Frey99 in the

report below, I have also included citations to the corresponding teachings in the Frey Slides.

G. RA.c

119. Source code file "RA.c," dated September 28, 1998, was written by David MacKay at the University of California at San Francisco.

120. The RA.c source code implements a "[r]epeat-accumulate code simulator." The file includes a function called "RA_encode" that performs a repeat-accumulate encoding operation.

121. The operation of RA.c is described in a comment at the beginning of the source code file:

```
/*
   RA.c
             (c) DJCM 98 09 28

   Repeat-accumulate code simulator

  read in code definition
  loop {
    encode source string
    add noise
    decode
  }

  Code definition: (stored in "alist")
                              Use of alist allows arbitrary numbers of repetitions
                              of each bit.

   K                          source block length
   n_1 n_2 ... n_K            number of repetitions of each source bit
   N = sum n_k
   alist  defines             permutation of N encoded bits
                              note, an additional permutation of the N accumulated
                              bits may be a good idea. (for non-memoryless channels)

  transmitted bits are integral of encoded bits

  Future plans:
   clump source bits into clumps. Have multiple parallel accumulated streams.
   Have little sub-matrices (like GF(q) ) defining response of accumulator to
   clumps.

*/
```

(RA.c at 1) (emphasis added).

122. As shown by the highlighted passages above, the comment at the top of RA.c explicitly refers to repeat-accumulate codes in which different information bits are repeated different numbers of times. Therefore, this comment, written

more than 1.5 years before the alleged conception date of the patents-in-suit, explicitly teaches irregular repeat-accumulate codes.

H. '999 Patent

123. U.S. Patent No. 4,623,999 to Patterson et al. (hereinafter, the "'999 patent"), was filed on June 4, 1984, more than 15 years before the filing date of the provisional application to which the patents-in-suit claim priority, and I have been informed that the '999 patent qualifies as prior art to all four of the patents-in-suit.

124. The '999 patent teaches an encoder for encoding information bits using a linear error-correcting code. The encoder taught by the '999 patent uses a plurality of memories that store values used during the encoding process ('999 patent at Abstract, describing "[a]n efficient look-up table encoder for encoding *k* bit information words with linear error correcting block codes is provided comprising a plurality of read-only **memories** ...") (emphasis added). The teachings of the '999 patent illustrate that the use of memories to implement error-correcting coders was known in the art for decades prior to the claimed priority date of the patents-in-suit.

I. Accused Hughes Products

125. As I explain below, the earliest priority date to which the claims of the '833 patent could be entitled is March 28, 2011, the date those claims were first filed.

126. I have been informed that a number of the accused products in this case were sold by Defendants prior to March 28, 2011. If the claims of the '833 patent are entitled to a priority date of March 28, 2011, these accused products would qualify as prior art to the claims of the '833 patent.

V. <u>SUMMARY OF ANTICIPATION AND OBVIOUSNESS OPINIONS</u>

127. As I explain in detail below, the asserted claims are either anticipated by or obvious over the prior art references described above. Broadly speaking, the

claimed codes represent the combination of RA codes, which were generally known by those of ordinary skill in the art by March 7, 2000, with irregularity, which had been shown years before to improve the performance of codes like RA codes.

128. One of ordinary skill in the art would have been motivated to combine these two ideas. RA codes are described in detail in Divsalar, published more than a year before the alleged conception date of the patents-in-suit. The concept of irregularity had been introduced by Luby in 1997, and by March 7, 2000 had been thoroughly explored in a number of papers and publications, including Frey99, MacKay, and the Luby '909 patent, discussed below (in particular, Frey99 teaches irregular *repetition*, which is specifically required by some of the asserted claims). By March 7, 2000, both RA codes and irregularity would have been common knowledge to one of ordinary skill in the art.

129. Indeed, prior to March 7, I myself suggested incorporating irregularity into RA codes. In particular, as described below, I suggested in an email to Dariush Divsalar that he make his RA codes irregular (*See* Email from Brendan Frey to Dariush Divsalar dated Dec. 8, 1999 (CALTECH000024021)). Consistent with the email I sent to Dr. Divsalar, making RA codes irregular was merely an obvious application of my earlier work on irregular turbocodes, which I presented at the Allerton 1999 conference, 6 months before Caltech's alleged conception date of March 7, 2000, and which is described in Frey99 and the Frey Slides.

130. I explain these opinions in further detail below, with reference to each limitation of the various claims that have been asserted by Caltech.

VI. <u>THE ASSERTED CLAIMS OF THE '710 PATENT ARE INVALID</u>

131. As I explain below, asserted claims 1, 4, 6, 15, 20, and 22 of the '710 patent are invalid. I also explain why claims 3, 5, and 21, from which claims 4, 6, and 22

depend, respectively, are invalid. A summary of the opinions set forth in this section is given in the table below:

'710 Claim	Frey99 (or Frey slides)	Frey99 (or Frey slides) + Divsalar	Divsalar + Luby	Divsalar + MacKay
1	Anticipated	Anticipated by Frey or Obvious	Obvious	Obvious
3	Anticipated	Anticipated by Frey or Obvious	Obvious	Obvious
4		Obvious	Obvious	Obvious
5		Obvious	Obvious	Obvious
6		Obvious	Obvious	Obvious
15		Obvious	Obvious	Obvious
20		Obvious	Obvious	Obvious
21		Obvious	Obvious	Obvious
22		Obvious	Obvious	Obvious

A. Claim 1 of the '710 Patent is Invalid

132. Claim 1 of the '710 patent reads:

> 1. A method of encoding a signal, comprising:
>
> obtaining a block of data in the signal to be encoded;
>
> partitioning said data block into a plurality of sub-blocks, each sub-block including a plurality of data elements;
>
> first encoding the data block to from[21] a first encoded data block, said first encoding including repeating the data elements in different sub-blocks a different number of times;
>
> interleaving the repeated data elements in the first encoded data block; and
>
> second encoding said first encoded data block using an encoder that has a rate close to one.

i) <u>Claim 1 of the '710 Patent is Anticipated by Frey99</u>

133. I explain below, one limitation at a time, why claim 1 is anticipated by Frey99.

[21] I note that the word "from" here should be "form." That is, this limitation is about forming "a first encoded data block." Notwithstanding that typographical error, I have reproduced the claim as it is printed in the patent.

a) *"A method of encoding a signal"*

134. Even if the preamble limits the claim, it is taught by Frey99. As I explain above, Frey99 deals with the construction of irregular turbocodes. The purpose of the disclosed irregular turbocode is for the encoding and decoding of signals (*see also*, Frey Slides at 4). Frey99 explicitly discloses decoding signals that had been encoded using the disclosed irregular turbocode. *See*, *e.g.*, Frey99 at 4 ("***After receiving the channel output, the decoder computes*** the channel output log-likelihood ratios …") (emphasis added); 4 ("In our simulations, after each iteration, we check to see if the current decision gives a codeword. If it does, the iterations terminate and otherwise, ***the decoder iterates further*** …") (emphasis added); 6 ("Fig. 4 shows the simulated BER-E_b/N_0 curves for the original block length N-131,072 regular turbocode (dashed line) and its irregular cousin (solid line), using profile e = 10, $f_e = 0.05$"); *see also*, Frey Slides at 2 ("making decoding easier"); Frey Slides at 9, 11, 12 (showing BER-E_b/N_0 curves).

b) *"obtaining a block of data in the signal to be encoded"*

135. Frey99 deals exclusively with block codes. For example, Frey99 includes experimental results comparing a regular code and an irregular code, both having "block length $N = 131,082$" (Frey99 at 6; *see also* Frey Slides at 13, teaching "long block lengths," and "short block lengths"). Frey99's use and discussion of that block length means that Frey99 takes bits in blocks of 131,082 and encodes them, just as is required by this claim limitation. Similarly, Frey99 also includes other discussion of obtaining data in blocks for encoding. For example, Frey99 describes experimental results relating to, *e.g.*, the "block length" of irregular turbocodes. In selecting a coding profile, Frey99 teaches "making small changes to a <u>*block length $N = 10,000$*</u> version of the original rate $R = 1/2$ turbocode proposed by Berrou *et al.*" (Frey99 at 5) (emphasis added). Also, Frey99 uses the "BER" or "block error rate" to compare the performance of various codes (*see*, *e.g.*,

Frey99 at Figure 4). Frey99's reference to "block error rate" means that Frey99 obtains data in blocks for encoding.

c) *"partitioning said data block into a plurality of sub-blocks, each sub-block including a plurality of data elements"*

136. Frey99 teaches this limitation. Frey99 describes irregular turbocodes as follows: "an *irregular turbocode* has the form shown in Fig. 2, which is a type 'trellis-constrained code' as described in [7]. We specify a *degree profile*, $f_d \in [0, 1]$, $d \in \{1, 2, \dots, D\}$. ***f_d is the fraction of codeword bits that have degree d and D is the maximum degree. Each codeword bit with degree d is repeated d times before being fed into the permuter.*** Several classes of permuter lead to linear-time encodable codes. In particular, if the bits in the convolutional code are partitioned into 'systematic bits' and 'parity bits', then by connecting each parity bit to a degree 1 codeword bit, we can encode in linear time." Frey99 at 2 (emphasis added).

137. As described above, Frey99 partitions the information bits into groups, where the bits in each group all have the same degree (*i.e.*, they are all repeated the same number of times). Frey99 also illustrates this operation graphically in Figure 2, reproduced below:

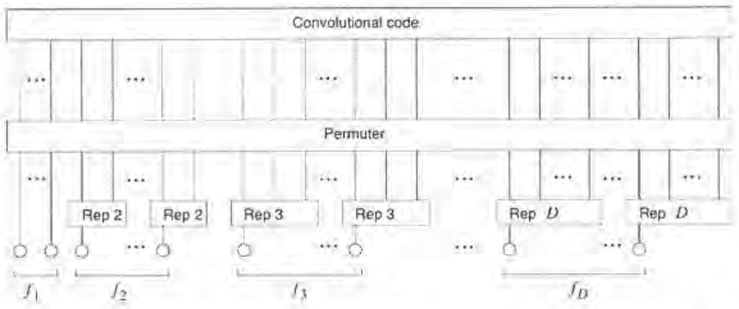


Figure 2: A general *irregular turbocode*. For $d = 1, \dots, D$, fraction f_d of the codeword bits are repeated d times, permuted and connected to a convolutional code.

138. In Figure 2 of Frey99, the circles at the bottom represent information bits. The groups of information bits labeled $f_2, f_3, \ldots, f_D$ represent sub-blocks into which the data block is partitioned (*see also* Frey Slides at 5).

139. Thus, the bits that are repeated twice (the bits labeled f_2) constitute one sub-block, the bits that are repeated three times (the bits labeled f_3) constitute a second sub-block, and so on. As shown in Figure 2 of Frey99, each of these sub-blocks contains a plurality of bits (or "data elements"), as required by claim 1 of the '710 patent.

d) *<u>"first encoding the data block to from [sic] a first encoded data block, said first encoding including repeating the data elements in different sub-blocks a different number of times"</u>*

140. Frey99 teaches repeating the data elements in different sub-blocks a different number of times (which is commonly known as "irregular repetition" to those of ordinary skill in in the art).

141. For example, Figure 2 of Frey99, reproduced above, shows that the data elements in each sub-block are repeated a different number of times. In Figure 2 of Frey99, the circles at the bottom represent information bits in the data block. The groups of information bits labeled $f_2, f_3, \ldots, f_D$ represent sub-blocks into which the data block is partitioned. The blocks labeled "Rep 2," "Rep 3," and "Rep D" represent the step of repetition. For example, an information bit that is connected to a box labeled "Rep 2" is repeated twice, a bit connected to a box labeled "Rep 3" is repeated three times, etc.. In the figure above, the repeated bits are represented by the vertical lines connecting the "Rep *n*" boxes to the box labeled "Permuter" (*see also* Frey Slides at 5).

e) *"interleaving the repeated data elements in the first encoded data block"*

142. Frey99 teaches this limitation. As I explain above, Frey99 teaches codes in which "Each codeword bit with degree *d* is repeated *d* times before being fed into the permuter" (Frey99 at 2; *see also* Frey99, Figures 1 and 2). Figure 1 of Frey99 illustrates how "a turbocode can be viewed as a code that copies the systematic bits, ***permutes both sets of these bits*** and then feeds them into a convolutional code" (Frey99 at 3) (emphasis added). *See also* Frey Slides at 4, 5 (showing copies of systematic bits fed into a "Permuter" block).

143. "Permuting" means "interleaving," and a "permuter" is an "interleaver," as both parties have agreed in their Joint Claim Construction Statement (construing both "interleaver" and "permutation module" to mean "module that changes the order of data elements"). Permuting/interleaving bits means changing the order of the bits. The permuter in Figure 2 of Frey99 receives the repeated bits (produced by the blocks labeled "Rep 1," "Rep 2," …, and "Rep *D*") and interleaves them (*see also* Frey Slides at 5; *see also* Divsalar Tr. at 278:2-23).

f) *"second encoding said first encoded data block using an encoder that has a rate close to one"*

144. Frey99 teaches this limitation. The "second encoding" taught by Frey99 is a convolutional encoder, which accepts irregularly repeated and permuted bits as input and encodes these bits to produce parity bits, as shown in Figure 2, reproduced below (*see also* Frey Slides at 5):

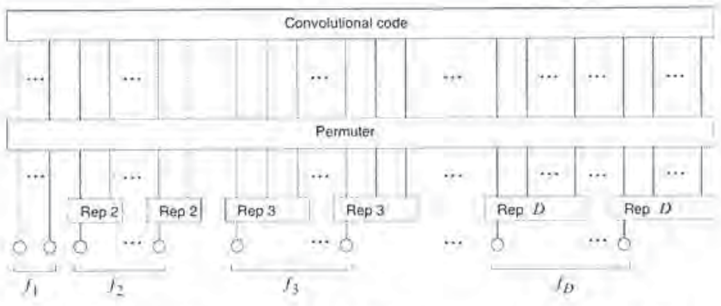


Figure 2: A general *irregular turbocode*. For $d = 1, \ldots, D$, fraction f_d of the codeword bits are repeated d times, permuted and connected to a convolutional code.

145. Frey99 teaches a convolutional coder "with the required convolutional code rate of $R' = 2/3$" (Frey99 at 5). A code with a rate of 2/3 has a rate "close to one," as required by this limitation. Indeed, during prosecution of the '710 patent, the Applicant attempted to overcome a prior art rejection by replacing "a rate close to one" with "a rate within 50% of one," in issued claim 15 (Response dated May 5, 2005 at 7-8). A code with rate 2/3 clearly has a rate "within 50% of one," and Applicant's amendment suggests that "a rate close to one" is even broader.

146. However, Frey99 also teaches second encoders with a rate even closer to one. Repetition increases the number of bits input to the convolutional encoder, but the number of bits sent across the channel can be decreased by "puncturing." Frey99 teaches "it is clear that when the average degree is increased, the rate of the convolutional code must also be increased to keep the overall rate at 1/2" (Frey99 at 5). Recall from above that the rate of an encoder is equal to the ratio between the number of bits input to the encoder and the total number of bits output by the encoder. A convolutional coder with rate 2/3 outputs three bits for every two bits of input. Puncturing the convolutional code lowers the number of output bits, reducing the denominator of the ratio and thus raising the rate of the code.

147. Specifically, Frey99 teaches puncturing the convolutional code to obtain a convolutional code with rate:

$$R' = 1 - \frac{1-R}{d} = 1 - \frac{1/2}{1/2 + 2(1/2 - f_e) + ef_e}$$

(Frey99 at 5).

148. This equation includes two variables, *e*, and f_e. Frey99 presents results that "show that for $e = 10, f_e = 0.05$ is a good fraction, and that for $f_e = 0.05$, $e = 10$ is a good degree" (Frey99 at 5). Plugging the values $e = 10$ and $f_e = 0.05$ into the equation above, we obtain:

$$R' = 1 - \frac{\frac{1}{2}}{\frac{1}{2} + 2\left(\frac{1}{2} - f_e\right) + e \cdot f_e} = \frac{1.4}{1.9} \approx \mathbf{0.74}$$

149. One of ordinary skill in the art would recognize that a rate of 0.74 is a rate "close to one." *See also* Frey Slides at 6 (showing equation for convolutional code rate); Frey Slides at 7 (showing "$d_e = 10$"); Frey Slides at 8 (showing "$f_e = .05$"), leading to the same rate R' = 0.74.

g) <u>*Summary*</u>

150. As explained above, Frey99 teaches every limitation of claim 1 and therefore anticipates claim 1.

ii) <u>Claim 1 of the '710 Patent is Obvious Over Frey99 In View of Divsalar</u>

151. As explained above, in my opinion, Frey99 teaches every limitation of, and therefore anticipates, claim 1 of the '710 patent. However, in the event Frey99 is found not to teach the "rate close to one" limitation of claim 1, then claim 1 is obvious over the combination of Frey99 and Divsalar.

152. Specifically, I explain later in this section that Divsalar teaches a second encoding step using an encoder with a rate "close to one." Also, as I explain below, one of ordinary skill in the art would have been motivated to combine Divsalar and

Frey99 in general, and would specifically have been motivated to use the accumulator of Divsalar in the irregular turbo codes of Frey99. Finally, I explain why such a combination would represent a minor modification to the teachings of Frey99, and would not fundamentally change its principle of operation.

a) *<u>The accumulator of Divsalar has "a rate close to one"</u>*

153. If the rate of the second encoder taught by Frey99 were found to not be "***<u>close to one</u>***," as required by the final limitation of claim 1, it would have been obvious to substitute Divsalar's rate-1 accumulator for Frey's convolutional code. In such a combination, the code rate of the second encoder would be exactly one, which would satisfy the "close to one" requirement.

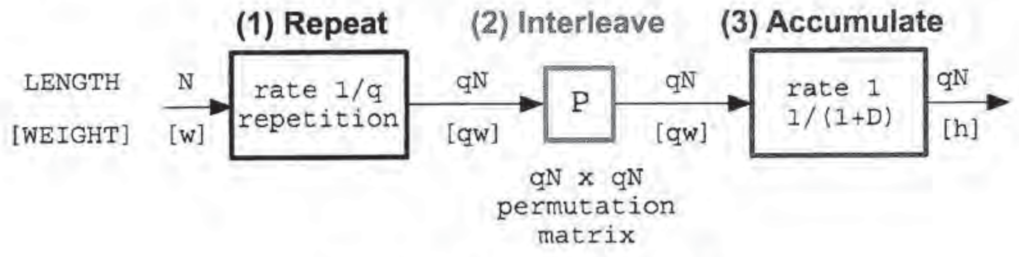
154. As explained above, Divsalar teaches an RA code that uses three steps:

(1) **repeat** bits q times;

(2) **interleave** the repeated bits (with the block labeled "P" in Figure 3); and

(3) **accumulate** the repeated-interleaved bits with the ***rate 1*** accumulator.

Each of these steps is represented by a block in Figure 3, reproduced below.



Divsalar, Figure 3 (annotated)

155. Divsalar explains the accumulate step as follows:

> [W]e prefer to think of [the accumulator] as a block coder whose input block $[x_1, \ldots, x_n]$ and output block $[y_1, \ldots, y_n]$ are related by the formula
>
> $y_1=x_1$
> $y_2=x_1+x_2$
> $y_3=x_1+x_2+x_3$
> $y_n=x_1+x_2+x_3+\ldots+x_n.$

(Divsalar at 5)

156. An encoder that outputs *n* bits (*i.e.*, "output block [$y_1, \ldots, y_n$]") for every *n* bits of input (*i.e.*, "input block [$x_1, \ldots, x_n$]) has a rate of $n/n = 1$. Thus, the "second encoder" taught by Divsalar has a rate of exactly 1 (and it is described in Fig. 3 of Divsalar as a "rate 1" encoder). Divsalar's accumulator therefore teaches exactly the second encoding step of claim 1 of the '710 patent.[22]

b) Motivations to combine the teachings of Frey99 with those of Divsalar, generally

157. Frey99 and Divsalar are both directed to the same field, namely, the field of error-correcting codes. Further, Frey99 and Divsalar are both related to variations on turbo codes. Frey99 is directed to irregular turbo codes (*see*, *e.g.*, Frey99 at 2, "[i]n this paper, we show that by tweaking a turbocode so that it is irregular, we obtain a coding gain …"; *see also* Frey Slides at 4, titled "Irregularizing" a turbocode). Divsalar is related to "turbo-like codes" (*see*, *e.g.*, Divsalar at 2, "In Section 3 we define the class of '*turbo-like*' codes …. In Section 4 we state a conjecture … about the ML decoder performance of *turbo-like* codes. In Section 5, we define a special class of *turbo-like* codes …") (emphasis added).[23] Also, as explained above, Divsalar teaches that the accumulator is a "truncated rate-1 recursive convolutional encoder with transfer function $1/(1 + D)$" (Divsalar at 5). Therefore, one of ordinary skill would have been aware of both references and would have considered them to disclose components that could be substituted for one another.

[22] As confirmed by the testimony of Hui Jin, one of the inventors listed on the patents-in-suit (*see* Jin Tr. at 122).

[23] The "turbo-like" codes described by Divsalar include both classical turbo codes and concatenated codes (*see* Divsalar at Abstract). Thus, every turbo code is a "turbo-like code," as that term is used in Divsalar.

c) Specific motivations to use Divsalar's accumulator in Frey99

158. Frey99's second encoder is implemented using a convolutional code. Divsalar's second encoder is implemented using an accumulator. Accumulation is a particular type of convolutional code that is simpler than the convolutional code used in Frey99 (*see*, *e.g.*, Divsalar Tr. at 279-280). Therefore, one of ordinary skill would have been motivated to substitute Divsalar's accumulator for Frey99's convolutional code at least for the following reasons.

159. First, using Divsalar's accumulator in place of Frey99's convolutional code would result in an encoder that was easier to implement in hardware, used fewer transistors and required fewer computations to produce the encoded codewords. As explained above, accumulation allows calculating each successive parity bit using a single modulo-2 addition operation. One of ordinary skill would have thus been motivated to simplify Frey99's code by replacing the convolutional coder with Divsalar's accumulator – an even simpler convolutional coder.

160. Second, converting Frey99's convolutional code into Divsalar's accumulator would result in a simpler code that would have been easier to analyze analytically. Divsalar's original motivation for producing the RA code was to produce a code that would be easy to analyze analytically. For example, the section of Divsalar that introduces RA codes begins: "[i]n this section we will introduce a class of turbo-like codes which are ***simple enough*** so that we can prove the IGE conjecture. We call these codes repeat and accumulate (RA) codes" (Divsalar at 5) (emphasis added). Indeed, Divsalar attempted to prove the IGE conjecture for more complicated coding schemes (*see id.* at 1, "[u]nfortunately, the difficulty of the first step … has kept us from full success, except for **some *very simple* coding systems, *which we call repeat and accumulate codes***") (emphasis added). One of ordinary skill would have been similarly motivated to simplify other codes in order to make them easier to study; one such simplification that would have been

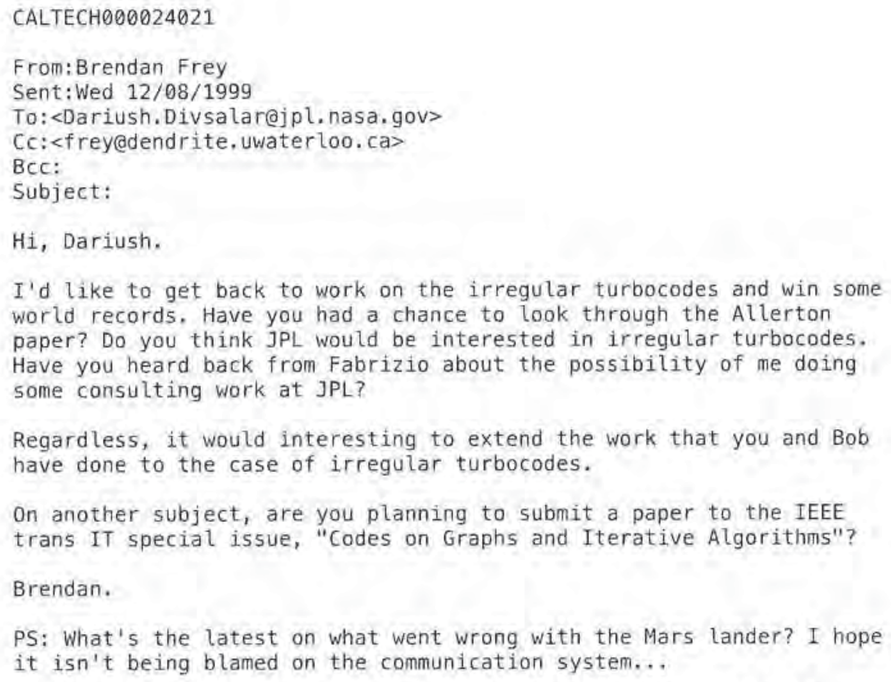
obvious to one of ordinary skill in the art would be replacing Frey99's convolutional coder with Divsalar's relatively less complex accumulator.

161. Also, convolutional coders and accumulators are related. That is, accumulation is a simple form of convolutional coding.[24] One of ordinary skill would recognize an accumulator as a simple form of convolutional coder. Divsalar teaches: "The accumulator can be viewed as a truncated rate-1 recursive convolutional encoder with transfer function 1/(1 + D)" (Divsalar at 5). Thus, if one of ordinary skill wanted to simplify the convolutional coder taught in Frey99, e.g., for the reasons given above, an accumulator would have been a logical choice because it would be a simple form of the convolutional coder explicitly disclosed in Frey99.

162. Further, using Divsalar's accumulator in place of the convolutional encoder explicitly taught in Frey99 would have been a routine substitution of one component for another and the resulting combination would have performed as expected.

163. Finally, my own presentation at the Allerton Conference in September 1999 taught that making a turbocode irregular would improve its performance. Below, I provide additional evidence that it would be obvious to a person of ordinary skill in the art that the RA code of Divsalar is a simple convolutional code and that it could be made irregular. The email below was sent to Dariush Divsalar by myself on December 8, 1999, nearly three months before the claimed date of conception of the patents-in-suit. The email mentions my paper on irregular turbocodes (Frey99) and Dariush Divsalar and Robert McEliece's work on RA codes (Divsalar), and further goes on to mention combining the two pieces of work.

[24] Divsalar described his accumulator as a convolutional code (Divsalar at 1 ("...and the inner code is a rate 1 convolutional code...")).



CALTECH000024021

From:Brendan Frey
Sent:Wed 12/08/1999
To:<Dariush.Divsalar@jpl.nasa.gov>
Cc:<frey@dendrite.uwaterloo.ca>
Bcc:
Subject:

Hi, Dariush.

I'd like to get back to work on the irregular turbocodes and win some world records. Have you had a chance to look through the Allerton paper? Do you think JPL would be interested in irregular turbocodes. Have you heard back from Fabrizio about the possibility of me doing some consulting work at JPL?

Regardless, it would interesting to extend the work that you and Bob have done to the case of irregular turbocodes.

On another subject, are you planning to submit a paper to the IEEE trans IT special issue, "Codes on Graphs and Iterative Algorithms"?

Brendan.

PS: What's the latest on what went wrong with the Mars lander? I hope it isn't being blamed on the communication system...

Email from Dr. Frey to Dr. Divslar (CALTECH000024021)

164. Other similarities between Divsalar and Frey99 further motivate the combination

165. As I explain in this section, Divsalar teaches not only the "rate close to one" limitation, but also most of the remaining limitations of claim 1 of the '710 patent. The similarity and combinability of Frey99 and Divsalar is evidenced by the number of claim limitations they both teach.

i. <u>"A method of encoding a signal"</u>

166. To the extent that the preamble is determined to be a limitation of the claim, it is taught by Divsalar. Divsalar describes a "turbo-like" code called a repeat-accumulate code. The purpose of the disclosed repeat-accumulate code is for the encoding and decoding of signals. Divsalar explicitly discloses decoding signals that had been encoded using the disclosed repeat-accumulate code. *See*, *e.g.*, Divsalar at 2 ("Finally, in Section 6 we present performance curves for some RA codes, using an iterative, turbo-like, decoding algorithm. This performance is seen

to be remarkably good, despite the simplicity of the codes and the suboptimality of the decoding algorithm"); 9 ("Figure 4. Comparing the RA code 'cutoff threshold' to the cutoff rate of random codes using both the classical union bound and the Viterbi-Viterbi improved union bound.").

ii. "obtaining a block of data in the signal to be encoded"

167. Divsalar deals exclusively with block codes. The repeat-accumulate codes introduced by Divsalar are encoded by receiving an "input block" or "information block of length N" and passing the block to the repeater (Divsalar at 5). *See also*, for example, Figure 3, reproduced above.

iii. "first encoding the data block to from a first encoded data block, said first encoding including repeating the data elements in different sub-blocks"

168. Divsalar teaches a first encoding step that includes repeating information bits, as shown in Figure 3, reproduced above.

169. A block of N information bits enters the coder at the left side of the figure and is provided to the repeater (labeled "rate 1/q repetition") (Divsalar at 5). The repeater duplicates each of the N information bits q times and outputs the resulting $N \times q$ repeated bits (*id.*).

170. While Divsalar does not teach partitioning the data block into a plurality of sub-blocks and repeating information bits in in different sub-blocks "a different number of times" (*i.e.*, irregular repetition), these limitations are taught by Frey99, as explained above.[25]

[25] Note that the "partitioning" and the "different number of times" limitations of claim 1 are related. Any coding scheme that repeats different information bits different numbers of times (such as that taught in Frey99) will *de facto* partition information bits into sub-blocks (*i.e.*, with bits in one sub-block being repeated one number of times and with bits in another sub-block being repeated a different number of times).

iv. "interleaving the repeated data elements in the first encoded data block"

171. Divsalar teaches this limitation. Figure 3 of Divsalar, reproduced above, shows a "permutation matrix" (the box labeled "P"). After the repeater duplicates each of the N information bits q times and outputs $N \times q$ repeated bits, the repeated bits are "scrambled by an interleaver of size qN" (Divsalar at 5).

v. Summary

172. As explained above, claim 1 of the '710 patent is obvious in view of the combination of Frey99 and Divsalar. In particular, it would have been obvious to use Divsalar's accumulator in place of the convolutional encoder disclosed in Frey99.

iii) Claim 1 of the '710 Patent is Obvious Over Divsalar in View of One of MacKay or Luby

173. I explain below, limitation by limitation, why claim 1 is rendered obvious by a combination of Divsalar and either MacKay or Luby. As noted above, Divsalar teaches all but one feature of the IRA codes that Caltech claims to have invented. That is, Divsalar teaches *regular* repeat-accumulate codes instead of *irregular* repeat-accumulate codes. Adding one feature, irregularity, to Divsalar results in the claimed IRA codes. As explained below, it would have been obvious to combine the teachings of Divsalar with the irregularity taught in either of Luby or MacKay.

174. In this section I describe how Divsalar teaches the remaining limitations of claim 1 of the '710 patent (*i.e.*, the limitations unrelated to irregularity). I also explain that any limitation not taught by Divsalar is taught by both Luby and MacKay. Also, as I explain below, one of ordinary skill in the art would have been motivated to combine Divsalar and Luby or MacKay in general, and would specifically have been motivated to incorporate the one necessary feature from

Luby or MacKay – irregularity – forming a combination that meets every limitation of claim 1 of the '710 patent. Finally, I explain why such a combination would only represent a minor modification to the teachings of Divsalar, and would not fundamentally change its principle of operation or purpose.

a) *<u>Divsalar teaches every limitation of Claim 1 except irregularity</u>*

175. As I explain above (with respect to the combination of Frey99 and Divsalar), Divsalar teaches:

- "A method of encoding a signal;"
- "obtaining a block of data in the signal to be encoded;"
- "first encoding the data block to from a first encoded data block, said first encoding including repeating;"
- "interleaving the repeated data elements in the first encoded data block;" and
- "second encoding said first encoded data block using an encoder that has a rate close to one."

176. The only portions of the claim that Divsalar fails to teach are: "partitioning said data block into a plurality of sub-blocks, each sub-block including a plurality of data elements" (I will call this the "partitioning" limitation); and "said first encoding including repeating the data elements in different sub-blocks a different number of times" (which I will call the "irregularity" limitation).

b) *<u>Both Luby and MacKay Teach the Partitioning and Irregularity Limitations</u>*

177. One of ordinary skill would have needed to incorporate only one feature from Luby or MacKay into Divsalar – irregularity – to form a combination that meets every limitation of claim 1 of the '710 patent. Below, I explain why one of ordinary skill would have been motivated to combine the teachings of Divsalar with the irregularity taught in both Luby and MacKay.

178. As I explain above, the partitioning limitation and the irregularity limitation are related: by repeating different information bits different numbers of times, a coding scheme *de facto* partitions information bits into sub-blocks. However, for the sake of clarity, I will discuss both limitations individually in the remainder of this subsection.

i. <u>"partitioning said data block into a plurality of sub-blocks, each sub-block including a plurality of data elements"</u>

179. Because Divsalar's repetition is regular instead of irregular, Divsalar does not partition the blocks into sub-blocks for purposes of repetition. However, as part of their general teaching of irregularity, both Luby and MacKay teach partitioning the blocks into sub-blocks.

180. Luby teaches that sparse graph codes can be improved by using "irregular graphing" (*see*, *e.g.*, Luby at 11:23-49). The "irregular graphing" encoder used by Luby "partition[s] said data block into a plurality of sub-blocks, each sub-block including a plurality of elements."

181. This process is represented graphically in Figure 17 of Luby, reproduced below:

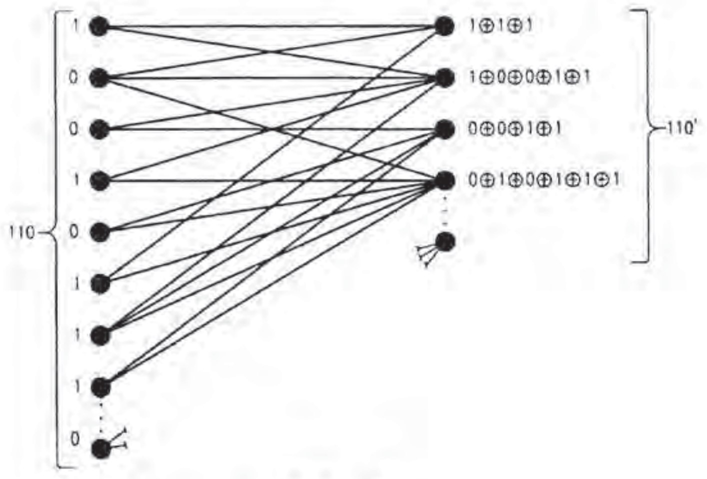


FIG. 17

182. In this figure, the filled circles on the left represent information bits to be encoded, and the filled circles on the right represent parity checks computed for these information bits. In the above figure, each parity check on the right is computed by summing together (modulo 2) all of the information bits connected to that parity check by an edge in the graph (*see*, *e.g.*, Luby at 17:64-67, "[t]he redundant data items associated with nodes at the layer 110' are computed by an exclusive-or operation of the message bits to which they are connected").

183. As shown above, some information bits are connected to two parity checks (*i.e.*, have a degree of two) and other information bits are connected to three parity checks (*i.e.*, have a degree of three). Luby "partition[s] said data block into a plurality of sub-blocks, each sub-block including a plurality of elements" by assigning a first group of two or more input bits a first degree (*e.g.*, two) and a second group of input bits a second degree (*e.g.*, three). In this scheme, the input bits with a degree of two constitute one sub-block (shown below in green) and the input bits with a degree of three constitute a second sub-block (shown below in red):

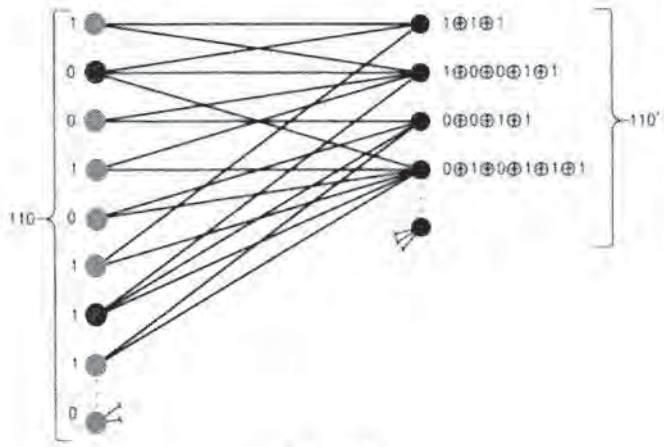


FIG. 17

184. Each of the sub-blocks includes at least two input bits, as shown in the colored figure above.

185. MacKay also teaches this limitation. MacKay builds on the earlier work of Luby to examine the properties of certain irregular Gallager codes (*see*, *e.g.*, MacKay at 1449). Like Luby, MacKay describes assigning different degrees to different bits: "We can define an irregular Gallager code in two steps. First, we select a *profile* that describes the desired number of columns of each weight and the desired number of rows of each weight. The parity check matrix of a code can be viewed as defining a bipartite graph with 'bit' vertices corresponding to the columns and 'check' vertices corresponding to the rows. Each nonzero entry in the matrix corresponds to an edge connecting a bit to a check. The profile specifies the degrees of the vertices in this graph." (MacKay at 1449-1450).

186. As the passage above explains, in the parity-check matrices taught by MacKay, each information bit corresponds to a particular column, where the weight of that column (*i.e.*, the number of 1s contained in that column of the parity-check matrix) represents the degree of the information bit.[26] MacKay also teaches systematic codes that use constructions of parity check matrices where some columns correspond to information bits and other columns correspond to parity bits.

187. As shown in Table 1 of MacKay, reproduced below, the irregular code with "Profile 93" partitions the information blocks into two sub-blocks: one sub-block having degree 3 and the other having degree 9:

---

[26] In general, depending on how the matrix is represented, a particular information bit can correspond to either a row or a column of the generator matrix. That is, if the vector of information bits is multiplied by the generator matrix on the right (denoted **v***G*), then each information bit will correspond to a row of the generator matrix. Conversely, if the vector of information bits is multiplied by the generator matrix on the left (denoted *G***v**), then each information bit will correspond to a column of the generator matrix.

TABLE I
THE TWO PROFILES STUDIED IN THIS PAPER

Profile 3	Column weight	Fraction of columns	Row weight	Fraction
	3	1	6	1
Profile 93	Column weight	Fraction of columns	Row weight	Fraction
	3	11/12	7	1
	9	1/12		

(MacKay at 1451)

As the table above indicates, 1/12th of the information bits in "Profile 93" have degree 9, and the remaining 11/12ths of information bits have degree 3. Because the parity check matrices taught by MacKay have many more than 12 columns (*see id.*, showing "blocklength about $N = 10\ 000$"), each of these sub-blocks contains a plurality of bits, or data elements.

ii. <u>"… repeating the data elements in different sub-blocks a different number of times"</u>

188. As I explain above, Divsalar teaches repeating information bits. As shown in Figure 3, a block of N information bits enters the coder at the left side of the figure and is provided to the repeater (labeled "rate 1/q repetition") (Divsalar at 5). The repeater duplicates each of the N information bits q times and outputs the resulting $N \times q$ repeated bits (*id.*).

189. While Divsalar does not teach repeating data elements in different sub-blocks a different number of times (*i.e.*, "irregular" repetition), one of ordinary skill in the art would have known to combine the repetition of Divsalar with the irregularity of Luby or MacKay.

190. As I explain above, Luby teaches that sparse graph codes can be improved by using "irregular graphing" (*see*, *e.g.*, Luby at 11:23-49). "Irregular graphing" refers to codes with Tanner graphs in which some information nodes are connected to more check nodes than others (*see*, *e.g.*, *id.* at 3:27-29, stating that "different numbers of first edges are associated with the data items"). Thus, combining the

"irregular" encoder taught by Luby with the repetition taught by Divsalar would result in an encoder that "repeat[s] the data elements in different sub-blocks a different number of times," as required by claim 1.

191. The irregularity taught by Luby is represented graphically in Figure 17. In particular, the version of Luby's Fig. 17 reproduced above with green and red highlighting shows that some information bits (colored red) contribute to three parity checks whereas other information bits (colored green) contribute to only two parity checks.

192. While Luby does not explicitly teach repetition, Fig. 17 shows that information bits are *used* a different number of times. Reuse is not, in general, repetition; it is possible to reuse bits without repeating them.[27] However, the irregular reuse taught by Luby can be implemented using the repetition of Divsalar, as I explain in more detail below. In other words, one way to incorporate Luby's irregularity into Divsalar was to make Divsalar's repetition irregular.

193. MacKay also teaches this limitation. MacKay builds on the earlier work of Luby to examine the properties of certain irregular Gallager codes (*see*, *e.g.*, MacKay at 1449). Like Luby, MacKay describes assigning different degrees to different information bits: "[w]e can define an irregular Gallager code in two steps. First, we select a profile that describes the desired number of columns of each weight and the desired number of rows of each weight. The parity check matrix of a code can be viewed as defining a bipartite graph with 'bit' vertices corresponding to the columns and 'check' vertices corresponding to the rows. Each nonzero entry in the matrix corresponds to an edge connecting a bit to a check. The profile specifies the degrees of the vertices in this graph" (MacKay at 1449-1450).

[27] I understand that the Plaintiff attempted to argue that the two terms are synonymous, but the Court was correctly not persuaded by this argument. *See* Claim Construction Order (Dkt. No. 105) at 11 ("Caltech argues that "repeat" can also refer to the re-use of a bit, but the patent's claims and specification support the Court's construction").

194. As the passage above explains, in the parity-check matrices taught by MacKay, each information bit corresponds to a particular column, where the weight of that column (*i.e.*, the number of 1s contained in that column of the parity-check matrix) represents the degree of the information bit.

195. As I explain above with reference to Table 1 of MacKay, reproduced above, the irregular code with "Profile 93" taught by MacKay effectively partitions the information blocks into two sub-blocks: one sub-block having degree three and the other having degree nine. The information bits in the first sub-block contribute to three parity checks, while the information bits in the second sub-block contribute to nine.

196. Like the scheme taught by Luby, the codes taught by MacKay involve irregular *reuse* of information bits. As I explain above, reuse is not, in general, repetition. However, as with Luby, the irregularity taught by MacKay can be implemented using the repetition of Divsalar (and one way to incorporate MacKay's irregularity into Divsalar was to make the repetition irregular), as I explain in more detail below.

c) *Motivations to combine the teachings of Divsalar with those of Luby or MacKay, generally*

197. Divsalar, Luby and MacKay are directed to the same field, namely, the field of error-correcting codes. Further, all three references are related to variations and improvements on linear error-correcting codes, and in particular to error-correcting codes that can be encoded quickly. *See*, *e.g.*, Divsalar at 1 (referring to "*practical encoding* and decoding algorithms") (emphasis added); *see also* Luby at 2:51-55 ("it is an objective of the present invention to provide a technique for creating loss resilient and error correcting codes which substantially *reduce the time* required to *encode* and decode messages") (emphasis added); *see also* MacKay at 1449 ("whereas Gallager codes normally take N^2 time to encode, we investigate

constructions of regular and irregular Gallager codes that allow *more rapid encoding* and have smaller memory requirements in the encoder") (emphasis added). Accordingly, one of ordinary skill would have been aware of all the references and further would have understood that the teaching of one reference would inform that of the others. That is, one of ordinary skill would have expected to apply the teachings of the references to each other.

d) Motivations to Incorporate the irregularity of Luby or MacKay into the RA codes of Divsalar

198. Both Luby and MacKay are related to modifying known regular codes by introducing irregularity. MacKay notes that "[t]he best known binary Gallager codes are irregular codes" (MacKay at 1449), explaining that "[t]he excellent performance of irregular Gallager codes is the motivation for this paper, in which we explore ways of further enhancing these codes" (*id.*). Similarly, Luby shows that incorporating irregularity into known regular codes can improve performance (*see, e.g.*, Luby at 21:52-55, stating that "the failure rate using the [irregular] techniques described above provide a much lower failure rate than those obtainable with regular graphing of the left and right nodes utilized in conventional error correction encoding"). In view of the fact that both Luby and MacKay teach how regular codes can be improved by the introduction of irregularity, one of ordinary skill in the art would have been motivated to incorporate irregularity into the regular repeat-accumulate codes of Divsalar.

199. Luby's work on irregularity is fundamental to the field of coding, representing a major advance in coding theory with broad applicability across various types of codes. By the time the patents-in-suit were filed, a person of ordinary skill in the art would know that regular codes could be improved by the addition of irregularity.

200. Consistent with this view, Aamod Khandekar, one of the inventors named on the patents-in-suit, wrote in his Ph.D. thesis that "Luby et al. also introduced the concept of irregularity, which seems to provide hope of operating arbitrarily close to channel capacity in a practical manner, on a wide class of channel models" (Khandekar Thesis at 2).[28] Khandekar hails "the introduction of irregular LDPC codes by Luby et al." as a "major breakthrough" (*id.* at 46) and states that IRA codes were merely an application of Luby's "concept of irregularity to the ensemble of RA codes" as described in Divsalar (*id.*at 47; *see also id.* at 51).

201. For at least these reasons, it would have been obvious to one of ordinary skill in the art to incorporate irregularity into the RA codes of Divsalar.

e) *<u>Incorporating the irregularity of Luby or MacKay into the RA codes of Divsalar would not have been difficult</u>*

202. Incorporating irregularity into the RA codes of Divsalar would have been simple for one of ordinary skill; *i.e.*, one of ordinary skill would have converted the regular repeater shown in Figure 3 of Divsalar (reproduced above) into an *irregular* repeater. This modification would allow the other two components of the encoder – the interleaver and the accumulator – to remain unchanged.[29]

203. Divsalar teaches repeating each information bit q times (*see* Divsalar at 5). Using an irregular repeater that repeats some information bits more than others, and then interleaving and accumulating the irregularly repeated bits, would naturally result in an irregular code.

[28] When Khandekar refers to "Luby et al.," he is referring to Michael G. Luby and several of his colleagues. Michael G. Luby is the first-named inventor on the Luby reference. Khandekar cites four separate academic articles by Luby in his graduate thesis, which deals with the subject of IRA codes (*see* Khandekar Thesis at 103).

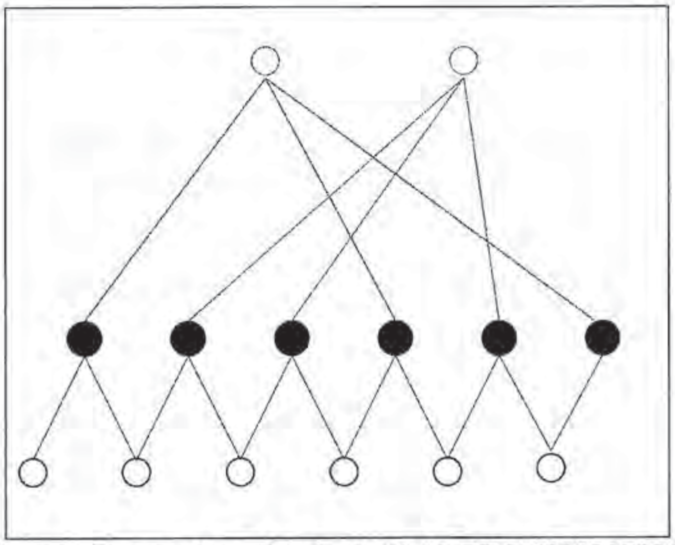
[29] Minor modifications could be made to the interleaver to account for interleaving a different number of bits. However, such modifications would not strictly be necessary. For example, if Divsalar's "repeat every bit q times" strategy were changed such that one bit was repeated q+1 times and another bit were repeated q-1 times, the repeat would be irregular and the interleaver would still deal with the same number of bits per block.

204. Nor would this modification have been challenging from a technological standpoint. Repeaters – whether regular or irregular – are conventional components that have been used for decades in a wide variety of digital electronics.[30] Modifying an existing encoder to replace a regular repeater with an irregular one would be a simple matter for one of ordinary skill in the art. Also, modifying the message passing decoder would be a simple matter for one of ordinary skill in the art, since the rules of deriving the decoder from the Tanner graph were broadly understood at the time.

205. Further, such a modification would preserve the simplicity of the RA codes taught by Divsalar. As I explain above, Divsalar introduced RA codes specifically because they are simple enough to analyze mathematically. IRA codes do not significantly add to the complexity of RA codes in this respect.

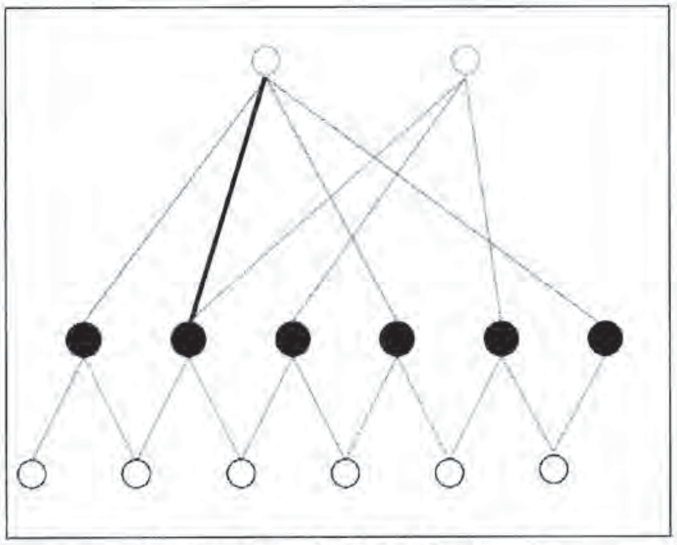
206. Indeed, IRA codes are so similar to RA codes that the Tanner graph representing any RA code can be modified to represent an IRA code by the addition of a single edge. For example, the figure below, taken from a presentation delivered by Aamod Khandekar, one of the inventors of the patents-in-suit, corresponds to an RA code in which each information node (the hollow circles at the top) contributes to three parity checks (represented by filled circles):

[30] As confirmed by, *e.g.*, the testimony of Stephen Wicker (*see* Wicker Tr. at 67).



Tanner Graph of an RA Code (CALTECH000007326)

207. This RA code can be turned into an IRA code by adding a single edge (shown in red below). Addition of that single edge (shown in red) makes the bit be repeated four times instead of three.[31]



Tanner Graph of an IRA Code

208. It would have been obvious to one of ordinary skill in the art to incorporate irregularity into Divsalar's RA code by making the repeater irregular. Divsalar's RA code repeater is a simple, obvious, and straightforward component to which to

[31] As confirmed by, *e.g.*, the testimony of Dariush Divsalar (*see* Divsalar Tr. at 248-249).

apply irregularity. As shown by the two Tanner graphs above, making the repeat irregular is exceedingly simple and does not overly complicate the code analytically. Choosing to make the repeater irregular is one of a finite number of identified, predictable ways to improve the performance of the code (which is the purpose of making a code irregular as taught by Frey99, MacKay, and Luby)..[32]

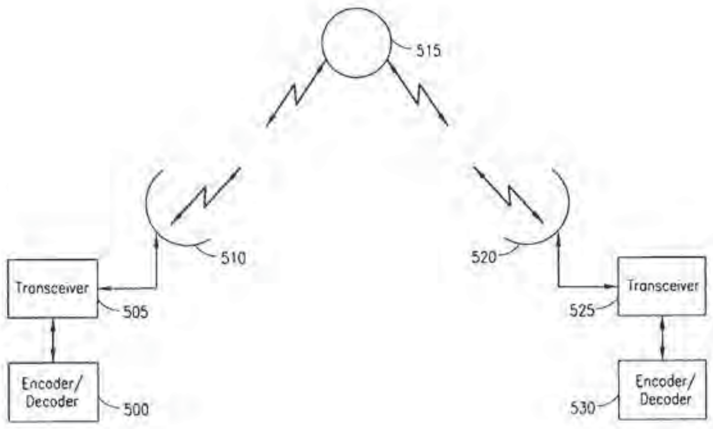
209. Finally, as I explain in the remainder of this section, Luby and MacKay teach not only the partitioning and irregular repetition limitations, but several of the other limitations of claim 1 of the '710 patent as well. The similarity and combinability of Divsalar and Luby or MacKay is evidenced by the number of claim limitations they all teach.

i. "A method of encoding a signal"

210. The preamble is taught by MacKay and Luby. As I explain above, MacKay describes both regular and irregular Gallager codes. The purpose of the disclosed Gallager codes is for the encoding and decoding of signals. MacKay explicitly discloses decoding signals that had been encoded using the disclosed Gallager codes. *See*, *e.g.*, MacKay at 1451 ("In the experiments presented here, we study binary codes with rate 1/2 and blocklength about $N = 10\ 000$. We simulate an additive white Gaussian noise channel in the usual way [2] and examine the block error probability as a function of the signal-to-noise ratio. The error bars we show are one standard deviation error bars on the estimate of the logarithm of the block error probability p defined"); *id.* ("Fig. 3 (a) Comparison of one representative of each of the constructions … (b) Representatives of all six constructions in Fig. 2").

[32] There are of course many options for making the repeat irregular (*e.g.*, repeat one bit one more times than the others, or use degree profiles suggested by Luby or MacKay) and a person of ordinary skill would have been motivated to design a particular code that had good performance. However, the decision to incorporate irregularity itself into the repeater was an easy one.

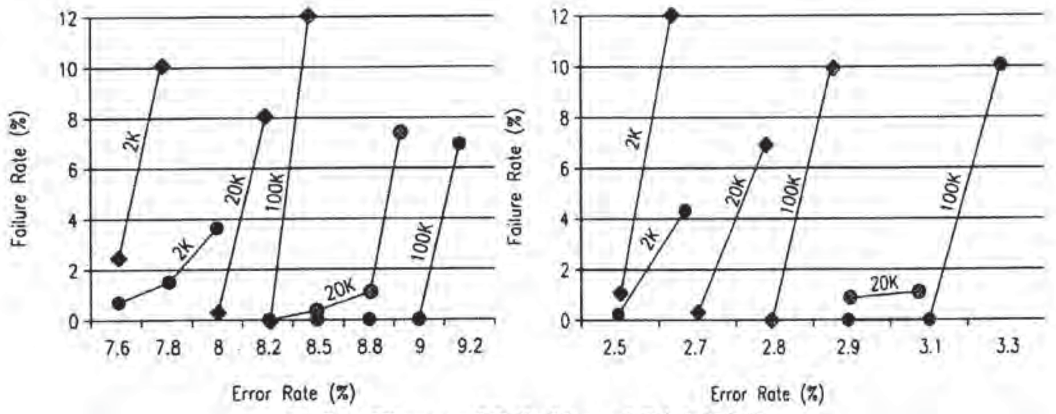
211. The preamble is also taught by Luby. As I explain above, Luby introduces irregularity to codes. The purpose of the disclosed irregular codes is for the encoding and decoding of signals. Luby explicitly discloses decoding signals that had been encoded using the disclosed codes. *See, e.g.*, Luby at Figs. 23, 24 (showing percent failure rate vs. percent error rate for various codes); *see also id.* at Fig. 25 (reproduced below, showing a signal being encoded, modulated, transmitted, received, and decoded):



Luby, Fig. 25

ii. <u>"obtaining a block of data in the signal to be encoded"</u>

212. Luby deals exclusively with block codes. For example, Figures 23 and 24 of Luby show experimental results comparing codes of various block lengths (*i.e.*, block lengths of 2K, 20K, or 100K bits):



Luby, Figures 23 (left) and 24 (right)

213. Mackay also deals with block codes. For example, Figure 1 of MacKay, reproduced below, shows experimental results relating to codes of various block lengths (*see*, *e.g.*, the block lengths of 24,000 and 65,536 bits identified in the caption of Figure 1):

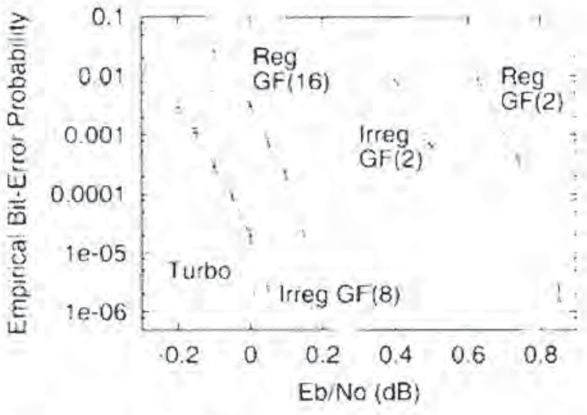


Fig. 1. Empirical results for Gaussian channel, rate 1/4 left–right: irregular LDPC, $GF(8)$ blocklength 24 000 bits; JPL Turbo, blocklength 65 536 bits; regular LDPC, $GF(16)$, blocklength 24 448 bits; irregular LDPC, $GF(2)$, blocklength 64 000 bits; regular LDPC, $GF(2)$, blocklength 40 000 bits. (Reproduced from [1].)

MacKay, Figure 1

214. As the figures above indicate, each of the codes taught by Luby and MacKay are associated with a "block length." The "block length" of a code is the number of bits, or "data elements," contained in the group of information bits that is encoded as a unit.

B. Claim 3 of the '710 Patent is Invalid

215. Claim 3 of the '710 patent reads:

> 3. The method of claim 1, wherein said first encoding is carried out by a first coder with a variable rate less than one, and said second encoding is carried out by a second coder with a rate substantially close to one.

i) <u>Claim 3 of the '710 Patent is Anticipated by Frey99</u>

216. As I explain above, Frey99 teaches every limitation of claim 1. Frey also teaches the limitations added by claim 3, namely that the "first encoding is carried out by a first coder with a variable rate less than one" and that the "second encoding is carried out by a second coder with a rate substantially close to one."

217. Frey99's first coder is the collection of blocks labeled "Rep2," "Rep 3," to "Rep D" in Figure 2. The rate of that encoder is a "variable rate less than one." Because the number of times bits are repeated varies from 1 to *D* (*see*, *e.g.*, Frey99 at Figure 2; *see also* Frey Slides at 5), the rate of the first encoder varies within a block between 1 and 1/*D*, where *D* may be set as high as desired. Also, because in an irregular turbocode some bits are duplicated at least once, the rate of the first encoder is always less than or equal to 1, and so the first encoder always has a rate less than one.

218. In the paragraph immediately above, I interpreted "variable rate" to refer to an encoder with a rate that varies within a block. As I explain above, under this interpretation of "variable rate," the encoder taught by Frey99 is a "variable rate" encoder. However, if "variable rate" were construed to mean that the rate of the encoder varies from block to block, then this claim would have been obvious in view of Frey99 because changing the rate of a code over time would have been easy for one of ordinary skill. "Variable rate," however, should not be construed to mean that the rate of the encoder varies from block to block because such an interpretation is not supported by the specification of the patents.

219. Frey99 also teaches a second coder with a rate "substantially close to one." As described above, Frey99 teaches a convolutional coder with a rate $R' \approx 0.74$. This is a rate "substantially close to one."

220. In summary, Frey99 teaches each and every limitation of claim 3 and therefore anticipates it.

ii) Claim 3 of the '710 Patent is Obvious Over Frey99 in View of Divsalar and Over the Frey Slides in View of Divsalar

221. As I explain above, the combination of Frey99 and Divsalar teaches every limitation of claim 1.

222. Even if Frey99 is found not to teach a second encoder with a rate "substantially close to one," this limitation is taught by Divsalar. As explained above, the "second coder" of Divsalar is an accumulator with a rate exactly equal to 1.[33] It would have been obvious to one of ordinary skill in the art to combine the teachings of Frey99 with those of Divsalar, also for the reasons given above.

223. Therefore, if Frey99 is found to not teach a second coder with a rate substantially close to one, then claim 3 is obvious over the combination of Frey99 and Divsalar.

iii) Claim 3 of the '710 Patent is Obvious Over Divsalar in View of One of Luby or MacKay

224. As I explain above, Divsalar combined with either Luby or MacKay renders claim 1 of the '710 patent obvious.

225. Divsalar in combination with either of Luby or MacKay also teaches a "first coder with a variable rate less than one." The "first encoding" step taught by these combinations of references is "irregular repetition," in which different information

[33] As confirmed by the testimony of Hui Jin, one of the inventors listed on the patents-in-suit (*see* Jin Tr. at 122).

bits are repeated different numbers of times (*i.e.*, Divsalar's repeater modified by the irregular teaching of Luby or MacKay).

226. The rate of the first encoder taught by these combinations of references is less than one. Because the first encoder is based on the principle of repetition, it always outputs more bits than it accepts as input, because it outputs multiple duplicates of each information bit. As explained above, the "rate" of an encoder is the ratio between the number of input bits and the number of output bits, so the rate of a repetition-based encoder is always less than one.

227. The rate of the first encoder taught by these combinations of references is also "variable." By combining the repetition of Divsalar with the irregularity of Luby or MacKay, we obtain an encoder that repeats different information bits different numbers of times. Therefore, depending on the particular information bit being encoded, the ratio of input bits to output bits – *i.e.*, the rate of the first encoder – varies.

228. Further, as explained above, Divsalar also teaches a second coder with a rate equal to one, and thus, a rate "substantially close to one" as required by claim 3 of the '710 patent.[34]

229. As I explain in detail above, it would have been obvious to incorporate the irregularity of Luby or MacKay (*i.e.*, a "variable rate encoder") with the repeat-accumulate codes taught by Divsalar. Thus, the combination of Divsalar with either Luby or MacKay renders claim 3 of the '710 patent obvious.

230. Finally, as explained above, although "variable rate" should not be construed to mean that the rate of the encoder varies from block to block, the claim would still be obvious over Divsalar in view of either MacKay or Luby because changing the rate of a code over time would have been easy for one of ordinary skill.

[34] As confirmed by, *e.g.*, the testimony of Hui Jin (*see* Jin Tr. at 122).

C. Claim 4 of the '710 Patent is Invalid

231. Claim 4 of the '710 patent reads:

> 4. The method of claim 3, wherein the second coder comprises an accumulator.

i) Claim 4 of the '710 Patent is Obvious Over Frey99 in View of Divsalar

232. As I explain above, the combination of Frey99 and Divsalar teaches every limitation of claim 3. Claim 4 adds to claim 3 that "the second coder comprises an accumulator." As explained above, Divsalar teaches that the second coder is an accumulator and it would have been obvious to use Divsalar's accumulator in Frey99. Claim 4 is therefore obvious over the combination of Frey99 and Divsalar.

ii) Claim 4 of the '710 Patent is Obvious Over Divsalar in View of One of Luby or MacKay

233. As I explain above, the combination of Divsalar with one of Luby or MacKay teaches every limitation of claim 3. I have also explained that the "second coder" of Divsalar is an "accumulator," as required by claim 4.

D. Claim 5 of the '710 Patent is Invalid

234. Claim 5 of the '710 patent reads:

> 5. The method of claim 4, wherein the data elements comprises bits.

i) Claim 5 of the '710 Patent is Obvious Over Frey99 in View of Divsalar

235. As I explain above, the combination of Frey99 and Divsalar teaches every limitation of claim 4. Claim 5 adds to claim 4 that "the data elements comprise bits." Both Frey99 and Divsalar teach methods of encoding signals in which the "data elements" comprise bits.

236. For example, Frey99 teaches codes in which "Each codeword _bit_ with degree *d* is repeated *d* times before being fed into the permuter" (Frey99 at 2) (emphasis added) (*see also* Frey Slides at 4, showing "parity bits" and "systematic bits", and at 5 in which open circles also represent bits). Divsalar teaches a "*binary* linear (*n*, *k*) block code" (Divsalar at 2) (emphasis added). One of ordinary skill in the art would understand Divsalar's "binary" block code is a code in which the input data elements are "**bi**nary digi**ts**" or "bits."

ii) Claim 5 of the '710 Patent is Obvious Over Divsalar in View of One of Luby or MacKay

237. As I explain above, the combination of Divsalar with one of Luby or MacKay teaches every limitation of claim 4. I have also explained that Divsalar teaches methods of encoding signals in which the "data elements" comprise bits.

238. Further, both Luby and MacKay teach encoding systems and methods that operate on bits. *See*, *e.g.*, Luby at 3:17-20 ("a method is provided for encoding a message having a plurality of data items, e.g. message packets or data bits"); *see also, e.g.*, MacKay at Figure 1.

E. Claim 6 of the '710 Patent is Invalid

239. Claim 6 of the '710 patent reads:

> 6. The method of claim 5, wherein the first coder comprises a repeater operable to repeat different sub-blocks a different number of times in response to a selected degree profile.

i) Claim 6 of the '710 Patent is Obvious Over Frey99 in View of Divsalar

240. As I explain above, the combination of Frey99 and Divsalar teaches every limitation of claim 5. Further, Frey99 teaches the limitation added by claim 6, *i.e.*, repeating "different sub-blocks a different number of times in response to a selected degree profile." As I explain in Frey99, "an *irregular turbocode* has the form shown in Fig. 2, which is a type 'trellis-constrained code' as described in [7].

We specify a *degree profile*, $f_d \in [0, 1]$, $d \in \{1, 2, \ldots, D\}$. f_d is the fraction of codeword bits that have degree d and D is the maximum degree. Each codeword bit with degree d is repeated d times before being fed into the permuter" (Frey99 at 2) (emphasis in original) (*see also* Frey Slides at 5, titled "Rate-degree relations) and 6, titled "Simplified degree profiles").

241. The "degree profile" described in the above passage from Frey99 determines what fraction of information bits are repeated d times, for all relevant values of d (*see also*, *e.g.*, Frey Slides at 6, "Degree d_e: Fraction f_e 'elite' bits have degree d_e").

ii) Claim 6 of the '710 Patent is Obvious Over Divsalar in View of One of Luby or MacKay

242. As I explain above, the combination of Divsalar with one of Luby or MacKay teaches every limitation of claim 5. Further, these combinations also teach the limitation added by claim 6.

243. MacKay teaches constructing an irregular Gallager code by selecting a degree profile: "We can define an irregular Gallager code in two steps. First, we select a *profile* that describes the desired number of columns of each weight and the desired number of rows of each weight. The parity check matrix of a code can be viewed as defining a bipartite graph with 'bit' vertices corresponding to the columns and 'check' vertices corresponding to the rows. Each nonzero entry in the matrix corresponds to an edge connecting a bit to a check. The profile specifies the degrees of the vertices in this graph" (MacKay at 1449-1450) (emphasis in original).

244. Luby also teaches constructing an irregular code by selecting a degree profile. This process is represented graphically in Figure 17 of Luby, reproduced below:

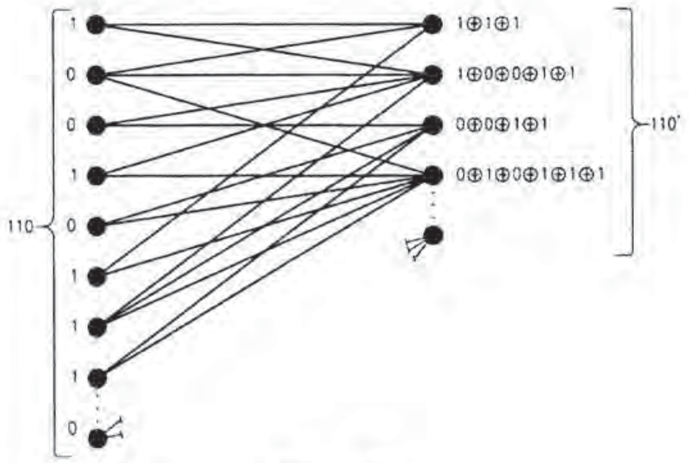


FIG. 17

245. This figure represents a degree profile for an irregular error-correcting code. As shown above, some information bits are connected to two parity checks (*i.e.*, have a degree of two) and other information bits are connected to three parity checks (*i.e.*, have a degree of three). A person of ordinary skill in the art would understand that assigning a first group of two or more input bits a first degree (*e.g.*, two) and a second group of input bits a second degree (*e.g.*, three) is selection of a "degree profile" for the code.[35]

246. By combining the repetition of Divsalar with the degree profiles taught by either Luby or MacKay, one of ordinary skill in the art would obtain "a repeater operable to repeat different sub-blocks a different number of times in response to a selected degree profile," as required by claim 6 of the '710 patent.

247. Further, as I explain in detail above, it would have been obvious to incorporate the irregular degree profiles of Luby or MacKay with the repeat-accumulate codes taught by Divsalar. Thus, the combination of Divsalar with either Luby or MacKay renders claim 6 of the '710 patent obvious.

[35] This is consistent with the testimony of Dariush Divsalar (*see, e.g.*, Divsalar Tr. at 143-146).

F. Claim 15 of the '710 Patent is Invalid

248. Claim 15 of the '710 patent reads as follows:

> 15. A coder comprising:
>
> a first coder having an input configured to receive a stream of bits, said first coder operative to repeat said stream of bits irregularly and scramble the repeated bits; and
>
> a second coder operative to further encode bits output from the first coder at a rate within 10% of one.

i) <u>Claim 15 of the '710 Patent is Obvious Over Frey99 in View of Divsalar</u>

249. I explain below that Frey99 teaches every limitation of claim 15 of the '710 patent except the requirement that the second coder encode bits "at a rate within 10% of one." Also, as explained above with respect to claim 1, Divsalar teaches a second coder, *i.e.*, an accumulator, that has a rate of exactly one, and it would have been obvious to use Divsalar's accumulator in Frey99. Therefore, claim 1 is obvious in view of the combination of Frey99 and Divsalar.

a) *<u>Frey99 teaches every limitation of Claim 1 except "a rate within 10% of one"</u>*

i. <u>"A coder comprising …"</u>

250. Even if the preamble limits the claim, it is taught by Frey99. As I explain above, Frey99 deals with the construction of irregular turbocodes. A person of ordinary skill in the art would recognize that these turbocodes encode information bits using "a coder." Further, in the experimental results disclosed in Frey99 (and the Frey Slides) (*e.g.*, as identified above with respect to the preamble of claim 1), the encoded bits were produced by a coder.

ii. "a first coder having an input configured to receive a stream of bits, said first coder operative to repeat said stream of bits irregularly and scramble the repeated bits"

251. Frey99 teaches this limitation. As explained above in the context of claim 1 of the '710 patent, Frey99 teaches a first coder that irregularly repeats bits. Frey99 further teaches that the irregularly repeated bits are passed as input to a permuter, which scrambles the repeated bits.

252. A "stream" of bits, as that term is used by those of ordinary skill in the art, is merely a sequence of bits. Block encoders like the encoders taught by Frey99, and the ones described in the specification of the patents-in-suit, receive a "stream" of bits and partition that stream into blocks of bits. Each block of bits is then encoded by a first encoder, the encoded bits are then interleaved, and the interleaved bits are encoded by a second encoder, producing a codeword. One of ordinary skill in the art would thus understand that the methods and systems taught in Frey99 operate on a stream of bits.[36]

iii. "a second coder operative to further encode bits output from the first coder"

253. Frey99 teaches "a second coder operative to further encode bits output from the first encoder." The "second coder" taught by Frey99 is a convolutional encoder, which accepts irregularly repeated and permuted bits as input and encodes

[36] I understand that Caltech has accused DVB-S2 LDPC encoders of infringement. The DVB-S2 LDPC encoder is a block encoder that operates on fixed size blocks. If by "stream," Caltech meant an un-partitioned continuous set of bits, then the "stream" limitation could not be infringed. I therefore understand "stream" in the asserted claims to mean a sequence of bits. Even in the absence of considerations of DVB-S2, "sequence of bits" is the meaning one of ordinary skill would assign to "stream" in the asserted patents. I note that in the *Inter Partes Review*, Prof. Pfister considered the alternate interpretation of "stream," i.e., an un-partitioned continuous set of bits. Even under that interpretation, the claims using the "stream" limitation would be obvious in view of the references addressed herein. However, for the reasons explained herein, Caltech must interpret "stream" to cover block codes to preserve its infringement case and therefore under Caltech's application of the claims, references that describe block codes meet the "stream" limitation.

these bits to produce parity bits, as shown in Figure 2, reproduced below (*see also* Frey Slides at 5):

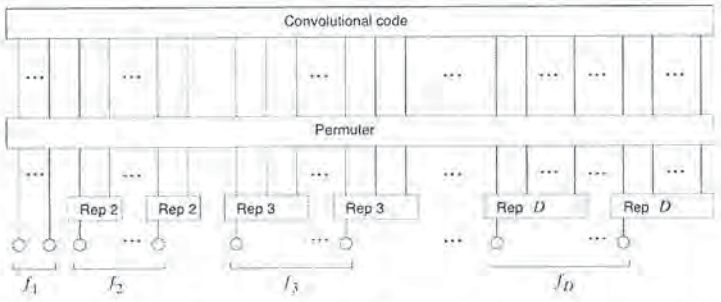


Figure 2: A general *irregular turbocode*. For $d = 1, \ldots, D$, fraction f_d of the codeword bits are repeated d times, permuted and connected to a convolutional code.

b) *<u>"at a rate within 10% of one"</u>*

254. As I explain above, the accumulator of Divsalar is a "second encoder" with a rate that is exactly equal to 1.

c) *<u>One of ordinary skill in the art would have been motivated to combine Divsalar's accumulator with the irregular turbocodes of Frey99</u>*

255. As I explain above, one of ordinary skill in the art would have been motivated to combine Divsalar and Frey99 in general, and would specifically have been motivated to use the accumulator of Divsalar in Frey99.

d) *<u>The combinability of Frey99 and Divsalar is further demonstrated by Divsalar's teaching of other limitations of claim 15</u>*

256. As I explain in this section, Divsalar teaches not only a "rate within 10% of one," but also most of the remaining limitations of claim 15 of the '710 patent. The similarity and combinability of Frey99 and Divsalar is evidenced by the number of claim limitations they both teach.

i. "A coder comprising …"

257. Divsalar teaches the preamble. As I explain above, Divsalar describes a "turbo-like" code called a repeat-accumulate code. A "coder" capable of encoding information bits using a repeat-accumulate code is shown in Figure 3 of Divsalar, reproduced above.

ii. "a first coder having an input configured to receive a stream of bits, said first coder operative to repeat said stream of bits"

258. As explained above with reference to claim 1 of the '710 patent, Divsalar teaches a first coder that repeats bits. While Divsalar does not teach repeating the bits "irregularly," it would have been obvious to one of ordinary skill in the art to combine the repetition of Divsalar with the irregular repetition of Frey99, as I explained above with reference to claim 1 of the '710 patent.

iii. "and scramble the repeated bits"

259. Divsalar teaches this limitation. Figure 3 of Divsalar, reproduced above, shows a "permutation matrix" (the box labeled "P"). As I explain in detail above, after the repeater duplicates each of the N information bits q times and outputs $N \times q$ repeated bits, the repeated bits are "scrambled by an interleaver of size qN" (Divsalar at 5).

ii) Claim 15 of the '710 Patent is Obvious Over Divsalar in View of One of Luby or MacKay

260. Claim 15 is rendered obvious by a combination of Divsalar and either MacKay or Luby. As noted above, Divsalar teaches all but one feature of the IRA codes that Caltech claims to have invented. That is, Divsalar teaches *regular* repeat-accumulate codes instead of *irregular* repeat-accumulate codes. Adding one feature, irregularity, to Divsalar results in the claimed IRA codes. As explained in detail above, with reference to claim 1 of the '710 patent, it would

have been obvious to combine the teachings of Divsalar with the irregularity taught in either of Luby or MacKay.

a) *<u>Divsalar teaches every limitation of Claim 15 except irregularity</u>*

261. As I explain above, Divsalar teaches:

- "A coder comprising: …"
- "a first coder having an input configured to receive a stream of bits, said first coder operative to repeat said stream of bits …"
- "… and scramble the repeated bits;" and
- "a second coder operative to further encode bits output from the first coder at a rate within 10% of one"

262. The only portion of the claim that Divsalar fails to teach is: "… repeat[ing] said stream of bits *<u>irregularly</u>* …." (the "irregularity" limitation).[37]

b) *<u>Both Luby and MacKay teach the irregularity limitation</u>*

263. As explained above with reference to claim 1, Luby and MacKay each teach irregularity and one of ordinary skill would have been motivated to incorporate that irregularity into Divsalar. Doing so results in a combination that teaches all limitations of claim 15.

G. Claim 20 of the '710 Patent is Invalid

264. Claim 20 of the '710 patent reads as follows:

> 20. The coder of claim 15, wherein the first coder comprises a low-density generator matrix coder.

i) <u>Claim 20 of the '710 Patent is Obvious Over Frey99 in View of Divsalar</u>

265. As I explained above, the combination of Frey99 with Divsalar teaches every limitation of claim 15. Both Frey99 and Divsalar also teach the limitation

[37] To be clear, Divsalar does teach "repeating said stream of bits," but does not teach "repeating said stream of bits *<u>irregularly</u>*."

added by claim 20, *i.e.*, that the "first coder comprises a low-density generator matrix coder."

266. As explained in Appendix A, a generator matrix is a mathematical representation of an encoder that represents how information bits are transformed into encoded bits. A generator matrix is a two-dimensional array of 1s and 0s. A "low-density" generator matrix is a matrix with a relatively small number of 1s compared to the number of 0s.[38]

267. The generator matrix associated with a "repeat" encoder (whether regular, as taught by Divsalar, or irregular, as taught by Frey99) is a low-density generator matrix. For example, the following is a generator matrix that can be used to repeat each information bit three times:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

268. In this matrix, the rows correspond to bits input to the LDGM encoder: the first row corresponds to a first bit input to the encoder, the second row corresponds to a second input bit, the third to the third, and so on. Because each column of this matrix contains only a single "1," each parity bit produced by this matrix will be a duplicate (or "repeat") of one of the input (or "information") bits. If a column contained more than a single "1," then the corresponding parity bit would be a

[38] Caltech agrees with this interpretation of "low-density." As Caltech explains in its *Markman* tutorial, "[m]atrices with contain mostly zeroes and very few ones are called sparse matrices or low-density matrices" (Dkt. No. 85 at 14:13-14; *see also* Wicker Tr. at 60; *see also, e.g.*, Jin Tr. at 174).

combination of (or sum of) information bits, but this matrix contains no such columns.

269. The number of repeated bits generated by this encoder is defined by the number of "1s" appearing in each input bit's row. Using the generator matrix above, encoding a stream of input bits beginning with "101…" would result in an encoded sequence of bits that begins "111000111…"

270. As the example above shows, a generator matrix corresponding to a repeat encoder has exactly one "1" per column. Thus, a $k \times n$ repeater matrix, with k rows and n columns, contains a total of n 1s, for a total density of $n/(kn) = 1/k$. One of ordinary skill in the art would understand that a matrix having only a single 1 per column is a "low-density" matrix.[39]

271. Summarizing, the repeaters taught in both Frey99 and Divsalar correspond to an LDGM coder that uses a generator matrix of the form illustrated above. Because that matrix is "low-density," both Frey99 and Divsalar teach the limitation added by claim 20.

ii) <u>Claim 20 of the '710 Patent is Obvious Over Divsalar in View of One of Luby or MacKay</u>

272. As I explained above, the combination of Divsalar with one of Luby or MacKay teaches every limitation of claim 15. Also, as I explain above, Divsalar teaches a first coder, *i.e.*, a repeater, that is a low-density generator matrix coder. Even when Divsalar's repeater is made into an irregular repeater by incorporating Luby's or MacKay's teaching of irregularity, the repeater remains a low-density

[39] A generator matrix for repeating a very small block size may not be low density. For example, of the block size is two and each bit is repeated twice, the matrix would have four elements, two ones and two zeroes. With half of the elements being non-zero, the matrix would not be low density. However, such degenerate cases do not detract from the point that in general generator matrices for repeat codes are low density. Once the block size is increased sufficiently, the matrix will become low density. For example, a generator matrix for the block sizes explicitly contemplated in Frey99, Divsalar, Luby or MacKay would all be low density.

generator matrix coder. Therefore, claim 20 is obvious over Divsalar in view of one of Luby or MacKay.

H. Claim 21 of the '710 Patent is Invalid

273. Claim 21 of the '710 patent reads:

> 21. The coder of claim 15, wherein the second coder comprises a rate 1 linear encoder.

i) Claim 21 of the '710 Patent is Obvious Over Frey99 in View of Divsalar

274. As I explained above, the combination of Frey99 and Divsalar teaches every limitation of claim 15.

275. Also, as explained above with reference to claim 1 of the '710 patent, Divsalar teaches a second coder that is an accumulator having a rate equal to 1. Also, an accumulator is a linear encoder. The generator matrix for an accumulator has the form:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \cdots \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \cdots \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & \cdots \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & \cdots \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

276. As explained in Appendix A, a generator matrix represents a linear transformation, and any code (such as this one) that can be represented using a generator matrix is a linear code.

277. Claim 22 of the '710 patent underscores the fact that an accumulator is a linear encoder. It recites "[t]he coder of claim 21, wherein the second coder comprises an accumulator." It logically follows that Divsalar's accumulator is a particular example of a "rate 1 linear encoder," as required by claim 21 (*see also*, *e.g.*, Jin Dep. at 122:7-13).

ii) Claim 21 of the '710 Patent is Obvious Over Divsalar in View of One of Luby or MacKay

278. As I explained above, the combination of Divsalar and either Luby or MacKay renders claim 15 obvious. I also explained above how Divsalar teaches a second encoder comprising a rate 1 linear encoder. Therefore, claim 21 is also obvious over Divsalar in view of one of Luby or MacKay.

I. Claim 22 of the '710 Patent is Invalid

279. Claim 22 of the '710 patent reads:

> 22. The coder of claim 21, wherein the second coder comprises an accumulator.

i) Claim 22 of the '710 Patent is Obvious Over Frey99 in View of Divsalar

280. Above I explain how Divsalar and Frey99 render obvious claim 21, and further how Divsalar teaches a "second encoder" that comprises an "accumulator." Therefore, claim 22 is also obvious over the combination of Divsalar and Frey99.

ii) Claim 22 of the '710 Patent is Obvious Over Divsalar in View of One of Luby or MacKay

281. Above I explain how Divsalar combined with either Luby or MacKay render obvious claim 21, and further how Divsalar teaches a "second encoder" that comprises an "accumulator." Therefore, claim 22 is also obvious over Divsalar combined with either Luby or MacKay.

VII. THE ASSERTED CLAIMS OF THE '032 PATENT ARE INVALID

282. As I explain below, asserted claims 1, 18, 19, and 22 of the '032 patent are invalid. A summary of the opinions set forth in this section is given in the table below:

'032 Claim	Ping + Frey99 (or Frey slides)	Ping + MacKay or Luby	Divsalar + Frey99 (or Frey slides), Luby, or MacKay	Divsalar + Frey99 (or Frey slides), Luby, or MacKay + Ping
1	Obvious	Obvious (under Caltech's construction of "repeat")	Obvious	
18			Obvious	Obvious (Ping not necessary)
19			Obvious	Obvious
22			Obvious	Obvious (Ping not necessary)

A. Claim 1 of the '032 Patent is Invalid

283. Claim 1 of the '032 patent reads:

> 1. A method comprising:
>
> receiving a collection of message bits having a first sequence in a source data stream;
>
> generating a sequence of parity bits, wherein each parity bit "x_j" in the sequence is in accordance with the formula
>
> $$x_j = x_{j-1} + \sum_{i=1}^{a} v_{(j-1)a+i}$$
>
> where "x_{j-1}" is the value of a parity bit "j-1," and
>
> $$\sum_{i=1}^{a} v_{(j-1)a+i}$$
>
> is the value of a sum of "a" randomly chosen irregular repeats of the message bits; and
>
> making the sequence of parity bits available for transmission in a transmission data stream.

i) <u>Claim 1 of the '032 patent is Obvious over Ping In View of Frey99 (or Frey Slides)</u>

284. I explain below, one limitation at a time, why claim 1 is rendered obvious by Ping in view of Frey99 (or Frey Slides).

a) <u>*"receiving a collection of message bits having a first sequence in a source data stream"*</u>

285. Ping teaches "receiving a collection of message bits having a first sequence in a source data stream."

286. Ping refers to the collection of information bits to be encoded using the vector variable name **d**. Ping states: "[d]ecompose the codeword **c** as **c** = [**p**, **d**], where **p** and **d** contain the parity and information bits, respectively" (Ping at 38). Ping goes on to provide equations from which "**p** = $\{p_i\}$ can easily be calculated from a given **d** = $\{d_i\}$" (*id.*).

287. The term "message bits" is synonymous with "information bits." One of ordinary skill in the art would understand that the information bits **d**, as taught by Ping, is a "collection of message bits having a first sequence."

288. Further, as I explain above, under Caltech's application of the claims, a "data stream," is merely a sequence of bits. Block encoders like the ones taught by Ping, and the ones described in the specification of the patents-in-suit, receive a "collection of message bits having a first sequence in a source data stream."

b) *<u>"generating a sequence of parity bits, wherein each parity bit "x_j" in the sequence is in accordance with the formula</u>*

$$x_j = x_{j-1} + \sum_{i=1}^{a} v_{(j-1)a+i}"$$

289. This limitation means that each parity bit x_j in the claimed sequence of parity bits is equal to the sum of the previous parity bit x_{j-1} and the sum of "*a*" information bits, $\sum_{i=1}^{a} v_{(j-1)a+i}$.

290. This is precisely the coding method taught by Ping. Specifically, Ping teaches an encoding operation that calculates the parity bits $\{p_i\}$ using the information bits $\{d_i\}$ as an input as follows:

$$p_1 = \sum_j h^d_{1j} d_j$$

$$p_i = p_{i-1} + \sum_j h^d_{ij} d_j$$

(Ping at 38) (Eq. 4)

291. In Ping, the parity bits, referenced in the claim as x_j, are denoted using the letter *p* (*e.g.*, in Ping, the *i*th parity bit is denoted p_i).

292. As required by claim 1 of the '032 patent, the first parity bit of Ping, p_1, is calculated as the sum of a subset of information bits and, as shown below, each subsequent parity bit p_i is calculated by adding together the previous parity bit p_{i-1} (the green box) and a sum of bits in a subset of information bits (the red box):[40]

---

[40] As was well understood by those of ordinary skill, the "Σ" symbol denotes a summation. For example, $\sum_j h^d_{ij} d_j$ means $h^d_{i1} d_1 + h^d_{i2} d_2 + \ldots + h^d_{iJ} d_J$, where J is an integer. If the "*h*" values are all either zero or one, as they are in Ping, and the "*d*" values are information bits, then this equation produces a sum of information bits.

$$p_i = \boxed{p_{i-1}} + \boxed{\sum_j h^d_{ij} d_j}$$

Ping, Eq. 4

c) <u>*"where "x_{j-1}" is the value of a parity bit "j-1," and*</u>

$$\sum_{i=1}^{a} v_{(j-1)a+i}$$

<u>*is the value of a sum of "a" randomly chosen irregular repeats of the message bits"*</u>

293. Ping teaches an encoding method in which each parity bit is the sum of the previous parity bit plus the sum of a number of randomly chosen collections of the message bits. The expression $\sum_{i=1}^{a} v_{(j-1)a+i}$, as it appears in the claim, is given in Ping as $\sum_j h^d_{ij} d_j$ (Ping at 38).

294. The variable h^d_{ij} represents the value at the ith row and the jth column of the parity check matrix $\mathbf{H^d}$ (*see id.*). The variable d_j represents the value of the jth information bit. Thus,

$$\sum_j h^d_{ij} d_j$$

represents the sum of the bits in a subset of information bits (specifically, the subset of information bits d_j where $h^d_{ij} = 1$). As Ping explains, the matrix $\mathbf{H^d}$ is randomized. $\mathbf{H^d}$ is comprised of t sub-blocks $\mathbf{H^{d1}}, \ldots, \mathbf{H^{dt}}$ as follows:

$$\mathbf{H^d} = \begin{pmatrix} \mathbf{H^{d1}} \\ \vdots \\ \mathbf{H^{dt}} \end{pmatrix}$$

(Ping at 38)

295. Ping further states: "[i]n each sub-block $\mathbf{H}^{\mathbf{d}i}$, $I = 1, 2 \ldots t$, we ***randomly*** create exactly one element 1 per column and $kt/(n-k)$ 1s per row" (*id.*) (emphasis added). Thus, the particular information bits summed by the expression $\sum_j h_{ij}^{d} d_j$ are "randomly chosen," as required by claim 1 of the '032 patent.[41]

296. Ping therefore teaches everything in this limitation except the "irregular repeats" limitation. While Ping does not teach "irregular repeats" of the message bits, Frey99 (and the Frey Slides) teaches irregular repetition, as I explained above.

297. For the reasons given below, it would have been obvious to one of ordinary skill in the art to combine the accumulation-based encoding of Ping with the irregular repetition of Frey99 (or the Frey Slides).

d) <u>*"making the sequence of parity bits available for transmission in a transmission data stream"*</u>

298. As explained above, the codeword taught by Ping comprises parity bits, (denoted with the boldface letter **p**). Specifically, Ping teaches "[d]ecompos[ing] the codeword **c** as **c** = [**p**, **d**], where **p** and **d** contain the parity and information bits, respectively" (Ping at 38).

299. Ping also analyzes the performance of the codes it describes, graphing the BER of various LDPC-accumulate coders against various values of E_b/N_0:

[41] I understand that Caltech has argued that the asserted claims cover the DVB-S2 algorithm. A DVB-S2 encoder merely implements previously defined deterministic (non-random) operations. Therefore, under Caltech's application of the claims, "randomly chosen" must refer to random choices made while defining the coding algorithm itself (as opposed to making random choices during the encoding itself).

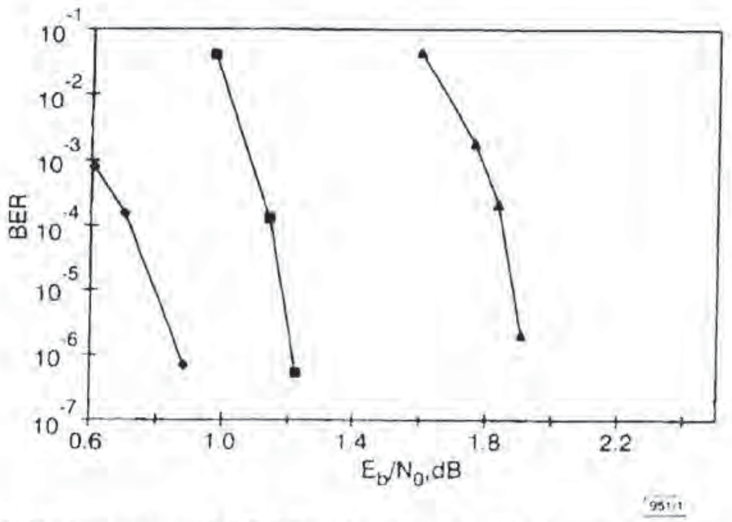


Fig. 1 *Performances of LDPC codes generated by semi-random parity check matrixes with $k = 30000$*

◆ $R = 1/3$
■ $R = 1/2$
▲ $R = 2/3$

Ping, Fig. 1

300. The concepts of "BER" and E_b/N_0 only make sense in the context of generating codewords (including the parity bits **p**), transmitting them over a noisy channel, and decoding them at the other end. Thus, Ping teaches "making the sequence of parity bits available for transmission in a transmission data stream," as required by claim 1 of the '032 patent.

e) *Summary*

301. As explained above, the combination of Ping and Frey99 (or the Frey Slides) teaches every limitation of claim 1 of the '032 patent.

f) *Motivations to Combine the teachings of Ping with those of Frey99 (or the Frey Slides)*

302. Ping and Frey99 (or the Frey Slides) are both directed to the same field, namely the field of error-correcting codes. In particular, both Ping and Frey99 relate to linear error-correcting codes and enhancements thereto: Frey99, titled "Irregular Turbocodes," teaches modifying known coding techniques to include

irregularity; Ping, titled "Low Density Parity Check Codes with Semirandom Parity Check Matrix," teaches constructing LDPC codes that can be encoded efficiently and have good BER vs. E_b/N_0 performance (*see* Ping at 39). Given that both references relate to improvements to error-correcting codes, one of ordinary skill in the art would have been motivated to combine their teachings.

303. Further, as explained above, Luby and MacKay taught that performance of a code could be improved by making the code irregular and that teaching was well known in the art prior to Caltech's claimed conception date or its filing date. That well-known teaching would have further motivated one of ordinary skill to incorporate Frey's irregular repetition into Ping's coding algorithm.

304. Further, combining irregular repetition as taught by Frey99 (and the Frey Slides) with accumulation as taught by Ping would have been a simple matter for one of ordinary skill in the art, as described above with reference to the asserted claims of the '710 patent. Such a combination would involve a routine substitution of one component for another and the resulting combination would have performed as expected.

ii) <u>Claim 1 of the '032 patent is Obvious over Ping In View of MacKay or Luby</u>

305. I understand that the Plaintiff attempted to argue that "repeat" and "reuse" are synonymous, but the Court was correctly not persuaded by this argument. *See* Claim Construction Order (Dkt. No. 105) at 11 ("Caltech argues that 'repeat' can also refer to the re-use of a bit, but the patent's claims and specification support the Court's construction"). Unless stated otherwise, my invalidity opinions in this report are based on the Court's construction that "repeat" should be given its plain meaning, which is "duplication." Claim Construction Order dated August 6, 2014, p. 10.

306. Plaintiff's infringement arguments, however, still appear to be based on an interpretation of "repeat" that does not require duplication, but merely reuse.[42] Under this interpretation, claim 1 of the '032 patent would be rendered obvious by Ping in view of either MacKay or Luby. Further, repeating bits in order to reuse them would not have been inventive and instead would have been nothing more than an implementation detail. Accordingly, even under the proper construction in which repeat means duplicate, the claims are still obvious over Ping in view of either MacKay or Luby.

307. As I explain above, Ping teaches every limitation of claim 1 except "irregular repeats." Neither MacKay nor Luby teach "repeats" under the Court's construction of the term "repeat – that is, they do not teach duplicating bits. However, MacKay and Luby do teach irregular *reuse* of bits, as I explain above with reference to the claims of the '710 patent.

308. It would have been obvious to one of ordinary skill in the art to combine the LDPC-accumulate coders of Ping with the irregularity of MacKay or Luby. As described above, Luby and MacKay are directed to the same field, namely the field of error correcting codes, and specifically, variations and improvements on linear error-correcting codes that allow them to be encoded more quickly. Ping is related to the same field; Ping, titled "Low Density Parity Check Codes with Semirandom Parity Check Matrix," teaches constructing LDPC codes that can be encoded efficiently and have good BER vs. E_b/N_0 performance (*see* Ping at 39). Given that Ping, MacKay, and Luby relate to improvements to error-correcting codes, one of ordinary skill in the art would have been motivated to combine the teachings of Ping with those of at least one of Luby or MacKay.

[42] That is, in DVB-S2, the parity bits are not repeats of the information bits. Rather, in DVB-S2, each parity bit is the sum of a collection of information bits. Thus, although information bits may be reused in DVB-S2's LDPC code, they are not repeated.

309. Further, because Luby and MacKay both taught that irregular codes perform better than regular ones, one of ordinary skill would have been motivated to incorporate irregularity into Ping. Ping's code is regular because each column in Ping's H^d matrix contains the same number of ones, i.e., each of Ping's columns contains exactly "t" ones (Ping at 38). However, changing Ping's H^d matrix such that not all columns had the same weight would have made Ping's code irregular and would have been an easy way for one of ordinary skill to incorporate irregularity into Ping. As explained above, MacKay teaches parity-check matrices in which each information bit corresponds to a column, where the weight of that column (*i.e.*, the number of 1s contained in that column of the parity-check matrix) represents the degree of the information bit. MacKay also notes that "[t]he best known binary Gallager codes are *irregular* codes whose parity check matrices have *nonuniform* weight per column" (Mackay at 1449) (emphasis in original). Given these teachings of MacKay, it would have been obvious to one of ordinary skill in the art to incorporate irregularity into the LDPC-accumulate coders of Ping by making the column weights of the parity check matrix $\mathbf{H}^d$ nonuniform.

310. Summarizing, Ping teaches a code that can be described as a regular LDPC followed by an accumulate (or a serial concatenated code in which the outer coder is a regular LDPC coder and the inner coder is an accumulator).[43] Thus, in Ping's code, every parity bit is the sum of (a) the previous parity bit and (b) a sum of randomly chosen regular "reuses" of the message bits. One of ordinary skill in the art would have been motivated by the teachings of Luby and MacKay to replace Ping's regular LDPC coder with an irregular LDPC coder.

[43] Ping's equation (4) for p_i, is of the form $p_i = p_{i-1} + X$, which is an accumulate operation and shows that Ping's outer coder is an accumulator. Further, the summation term in equation (4) (denoted by "X" in the prior sentence) provides an LDPC encoding, thus showing that Ping's inner coder is an LDPC coder.

311. In Ping's code as modified to include irregularity per the teachings of Luby or MacKay, each parity bit would be the sum of (a) the previous parity bit and (b) a sum of randomly chosen irregular "reuses" of the message bits.

312. Thus, under Caltech's theory that "repeat" means "reuse," claim 1 of the '032 patent would be rendered obvious by Ping in view of MacKay or Luby. Also, as noted above, repeating bits in order to reuse them would not have been inventive and instead would have been nothing more than an obvious implementation detail. Accordingly, even under the proper construction in which repeat means duplicate, the claims are still obvious over Ping in view of either MacKay or Luby.

iii) <u>Claim 1 of the '032 Patent is Obvious over Divsalar in view of Luby or MacKay</u>

313. I explain below, one limitation at a time, why claim 1 is rendered obvious by Divsalar in view of Luby or MacKay.

a) <u>*"receiving a collection of message bits having a first sequence in a source data stream"*</u>

314. As explained above with reference to the claims of the '710 patent, Divsalar teaches "receiving a collection of message bits having a first sequence." Also for the reasons explained above, while Divsalar does not explicitly make reference to an input configured to receive a "data stream," as required by this limitation, one of ordinary skill in the art would understand that the methods and systems taught in Divsalar operate on a data stream.

b) <u>*"generating a sequence of parity bits, wherein each parity bit "x_j" in the sequence is in accordance with the formula*</u>

$$x_j = x_{j-1} + \sum_{i=1}^{a} v_{(j-1)a+i}"$$

315. This limitation means that each parity bit x_j in the claimed sequence of parity bits is equal to the sum of the previous parity bit x_{j-1} and the sum of "a" information bits, $\sum_{i=1}^{a} v_{(j-1)a+i}$.

315. This limitation means that each parity bit x_j in the claimed sequence of parity bits is equal to the sum of the previous parity bit x_{j-1} and the sum of "a" information bits, $\sum_{i=1}^{a} v_{(j-1)a+i}$.

316. As explained above, the accumulator of Divsalar performs an accumulation operation as follows:

> [W]e prefer to think of [the accumulator] as a block coder whose input block [$x_1, \ldots, x_n$] and output block [$y_1, \ldots, y_n$] are related by the formula
>
> $y_1 = x_1$
> $y_2 = x_1 + x_2$
> $y_3 = x_1 + x_2 + x_3$
> $y_n = x_1 + x_2 + x_3 + \ldots + x_n$
>
> (Divsalar at 5)

317. This operation can be represented recursively using the equation $y_i = y_{i-1} + x_i$. Using the recursive formulation, one can see that each parity bit y_i is the sum of the previous parity bit y_{i-1} and a single information bit x_i. Therefore, for the case in which $a = 1$, Divsalar meets this limitation.

318. For cases in which "a" is greater than one, this limitation would be met by modifying the teachings of Divsalar so that each parity bit y_i is the sum of the previous parity bit y_{i-1} and *multiple* information bits x_{i1}, x_{i2}, and x_{i3}. That is, modifying the teachings of Divsalar so that $y_i = y_{i-1} + (x_{i1} + x_{i2} + x_{i3})$.

319. It would have been obvious to implement such a code by inserting a step between the interleaver and the accumulator that sums consecutive groups of a repeated bits. For example, for $a = 3$, this step would receive repeated information bits $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, \ldots$ and would output the sums of consecutive groups of three repeated information bits $(x_1 + x_2 + x_3)$, $(x_4 + x_5 + x_6)$, $(x_7 + x_8 +$

x_9), …. These sums would then be passed to the accumulator, resulting in a code where $y_i = y_{i-1} + (x_{i1} + x_{i2} + x_{i3})$, which satisfies this limitation of claim 1 of the '032 patent.

320. As explained below with reference to the claims of the '781 patent, this effect can also be achieved by "puncturing" some of the parity bits y_i output by Divsalar. If two out of every three parity bits were punctured, leaving only y_1, y_4, y_7, etc., then each parity bit (*e.g.*, y_4) would be the sum of the previous parity bit (*i.e.*, y_1) and a group of three information bits (*i.e.*, $x_1 + x_2 + x_3$). Thus, this would also result in a code where $y_i = y_{i-1} + (x_{i1} + x_{i2} + x_{i3})$, which satisfies this limitation of claim 1 of the '032 patent. As explained below, "puncturing" parity bits would have been well known to one of ordinary skill in the art (*see, e.g.*, Frey99 at 3).

321. Modifying the teachings of Divsalar in this way would have been obvious in view of the teachings of Luby or MacKay. As explained above Luby and MacKay teach LDPC codes (*see*, *e.g.*, Luby at 17:58-60, "[f]or example, a low-density parity check code defined by a graph similar to that used between the other layers is particularly suitable for this purpose …"; *see also*, *e.g.*, MacKay at Fig. 1). It was well known in the art that LDPC codes (*a.k.a.* "Gallager codes") are a class of high-performance error-correcting codes with desirable properties. As explained above, Gallager codes had been known in the art for decades by the time the patents-in-suit were filed, and had been the subject of intensive research since 1995, when they were rediscovered by David J. C. MacKay. Combining the repeat-accumulate codes taught by Divsalar with the LDPC codes taught by Luby and MacKay to create an LDPC-accumulate coder would have been obvious to one of ordinary skill in the art.

c) *"where "x_{j-1}" is the value of a parity bit "j-1," and*

$$\sum_{i=1}^{a} v_{(j-1)a+i}$$

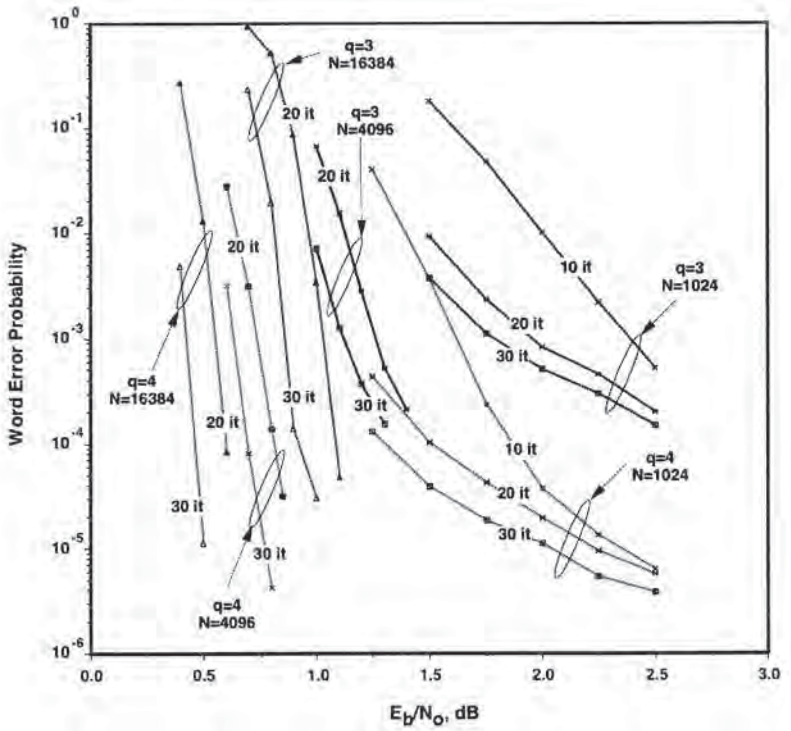
is the value of a sum of "a" randomly chosen irregular repeats of the message bits"

322. As I explain above, it would have been obvious to modify the teachings of Divsalar with the LDPC codes taught by Luby or MacKay by inserting a summation step between the interleaver and the accumulator of Divsalar. The resulting code would be a code in which $x_i = x_{i-1} + (v_{i1} + v_{i2} + v_{i3})$ (where $a = 3$). The quantity $(v_{i1} + v_{i2} + v_{i3})$ is the value of a sum of "*a*" repeats of the message bits. Because the bits are permuted by the interleaver prior to this step of summation, the repeated information bits v_{i1}, v_{i2}, v_{i3} are "randomly chosen," at least according to Caltech's interpretation, repeats of the message bits, as required by the claim.

323. Divsalar does not teach "irregular" repeats," as required by claim 1. However, as explained above with reference to the claims of the '710 patent, it would have been obvious to one of ordinary skill in the art to combine the regular repetition of Divsalar with the irregularity of MacKay and Luby, resulting in irregular repetition.

d) *"making the sequence of parity bits available for transmission in a transmission data stream"*

324. Divsalar teaches this limitation. Divsalar analyzes the performance of the codes it describes, graphing the word error probability of various RA codes against various values of E_b/N_0:



Divsalar, Fig. 5

325. The concepts of "BER" and E_b/N_0 only make sense in the context of generating codewords (including parity bits), transmitting them over a noisy channel, and decoding them at the other end. Thus, Divsalar teaches "making the sequence of parity bits available for transmission in a transmission data stream," as required by claim 1 of the '032 patent.[44]

e) *Summary*

326. As explained above, the combination of Divsalar and either Luby or MacKay teaches every limitation of claim 1 of the '032 patent.

f) *Motivations to Combine*

327. As explained above with reference to the claims of the '710 patent, one of ordinary skill in the art would have been motivated to combine the repeat accumulate codes of Divsalar with the irregular LDPC codes taught by MacKay

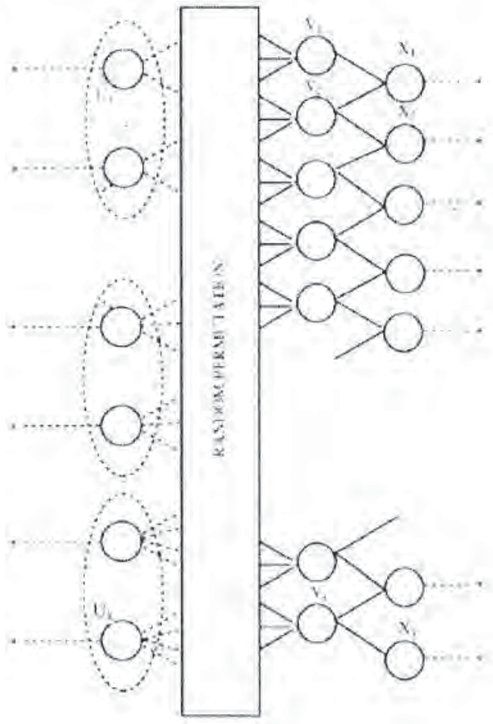
[44] This is consistent with the testimony of Dariush Divsalar (*see, e.g.*, Divsalar Tr. at 76-77).

and Luby, resulting in an LDPC-accumulate coder that satisfies the limitations of claim 1 of the '032 patent.

B. Claim 18 of the '032 Patent is Invalid

328. Claim 18 of the '032 patent reads:

> 18. A device comprising:
>
> a message passing decoder configured to decode a received data stream that includes a collection of parity bits,
>
> the message passing decoder comprising two or more check/variable nodes operating in parallel to receive messages from neighboring checking/variable nodes and send updated messages to the neighboring variable/check nodes,
>
> wherein the message passing decoder is configured to decode the received data stream that has been encoded in accordance with the following Tanner graph:



i) Claim 18 of the '032 Patent is Obvious over Divsalar in View of Frey99, Luby, or MacKay

329. I explain below, one limitation at a time, why Claim 18 of the '032 patent is rendered obvious by Divsalar in view of Frey99, Luby, or MacKay.

a) <u>*"a device comprising ..."*</u>

330. The encoding methods taught by Divsalar are performed by "a device." A schematic diagram of such a device (*i.e.*, an "encoder") is shown by Divsalar, Fig. 3, reproduced below:

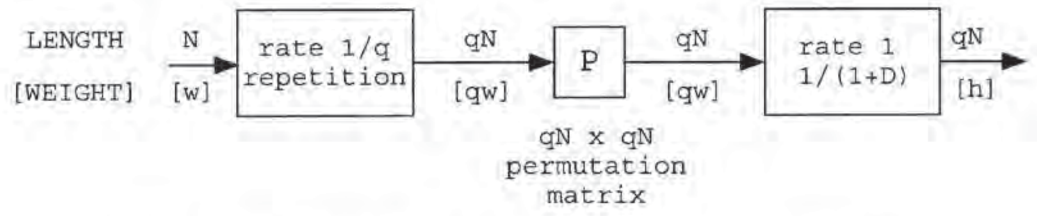


Figure 3. Encoder for a (qN, N) repeat and accumulate code. The numbers above the input-output lines indicate the length of the corresponding block, and those below the lines indicate the weight of the block.

Divsalar, Fig. 3

b) <u>*"a message passing decoder configured to decode a received data stream that includes a collection of parity bits"*</u>

331. Divsalar teaches "a message passing decoder configured to decode a received data stream that includes a collection of parity bits." Divsalar teaches that "an important feature of turbo-like codes is the availability of a simple iterative, **<u>message passing decoding</u>** algorithm that approximates ML decoding. We wrote a computer program to implement this "turbo-like" decoding for RA codes with q = 3 (rate 1/3) and q = 4 (rate 1/4), and the results are shown in Figure 5" (Divsalar at 9) (emphasis added).

332. Message passing decoders are conventional elements that were well known in the prior art.[45] A message passing decoder repeatedly calculates several related mathematical functions, where the functions are called "messages." A message passing decoder includes "variable nodes," which represent information bits, and

[45] *See generally*, *e.g.*, Judea Pearl, *Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach*, Proceedings of the Second National Conference on Artificial Intelligence Pittsburgh, PA 133-136 (1982) ("Pearl").

"check nodes," which represent mathematical constraints that the information bits must follow. Using an algorithm for decoding called "message passing decoding," messages are passed among the variable and check nodes in order to determine the original values of the information bits. One of ordinary skill would understand that this is what is referenced by Divsalar's use of the term "message passing decoding."[46]

333. Further, as explained above with reference to the asserted claims of the '710 patent, the decoding methods taught by Divsalar are intended to be applied to a "data stream."

c) <u>*"the message passing decoder comprising two or more check/variable nodes operating in parallel to receive messages from neighboring checking/variable nodes and send updated messages to the neighboring variable/check nodes"*</u>

334. This limitation is directed to features that are obvious elements in any message-passing decoder, including the message-passing decoder taught by Divsalar. A person of ordinary skill in the art would recognize that conventional implementations of decoding by "message passing", decoding by the "sum-product algorithm", decoding by "belief propagation", decoding by "probability propagation", decoding a code defined by a "Tanner graph" and decoding a code defined by a "factor graph" would in particular comprise two or more check/variable nodes operating in parallel to receive messages from neighboring check/variable nodes and send updated messages to the neighboring variable/check nodes. These operations are described in several publications prior to Divsalar, including in the teachings of R. G. Gallager ("Low Density Parity Check Codes", monograph, M.I.T. Press, 1963), of B. J. Frey and F. R. Kschischang (Presented at the Allerton Conference in September 1995, proceedings published in May 1996),

[46] As confirmed by the testimony of Dariush Divsalar (*see, e.g.*, Divsalar Tr. at 152-153).

of B. J. Frey, F. R. Kschischang, H.-A. Loeliger and N. Wiberg (Presented at the Allerton Conference in September 1996, proceedings published in May 1997) and of R. J. McEliece , D. J. C. Mackay and J.-F. Cheng (published in the IEEE Journal on Selected Areas in Communications, February 1998). I discuss message passing decoding further below with respect to Frey99, but that description of message passing decoding applies here as well.

335. Divsalar teaches that "an important feature of turbo-like codes is the availability of a simple iterative, ***message passing decoding*** algorithm that approximates ML decoding" (Divsalar at 9) (emphasis added). The iterative message-passing algorithm taught by Divsalar operates according to the principles common to conventional message-passing decoders (as taught by, *e.g.* Pearl) and therefore meets this limitation.[47]

d) *Tanner Graph*

336. The Court has construed this term to require "a graph representing an IRA code as a set of parity checks where every message bit is repeated, at least two different subsets of message bits are repeated a different number of times, and check nodes, randomly connected to the repeated message bits, enforce constraints that determine the parity bits."

337. As explained above, an IRA code is an "irregular repeat-accumulate" code, in which information bits are irregularly repeated and optionally interleaved, with the interleaved bits being passed to an accumulator, which generates the parity bits.

i. Divsalar teaches every requirement of the Tanner Graph limitation except irregularity

338. Divsalar teaches repeating message bits, as required by the Court's construction. This is shown in Figure 3 of Divsalar, reproduced below:

[47] This is consistent with the testimony of Dariush Divsalar (*see* Divsalar Tr. at 152-153).

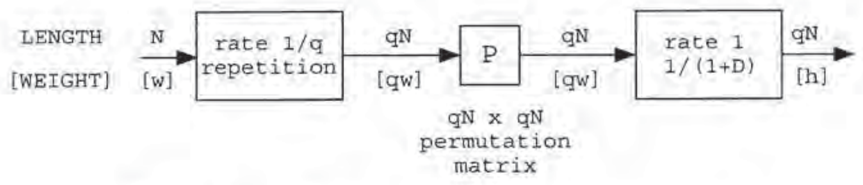


Figure 3. Encoder for a (qN, N) repeat and accumulate code. The numbers above the input-output lines indicate the length of the corresponding block, and those below the lines indicate the weight of the block.

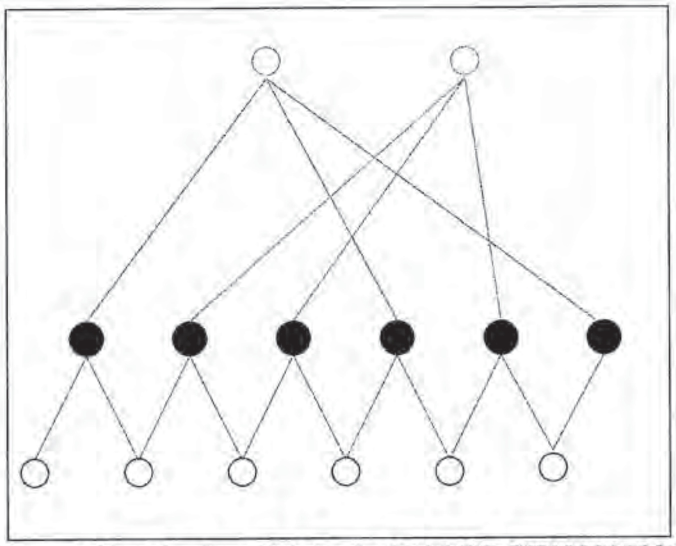
339. As explained above, a block of *N* information bits enters the coder at the left side of the figure and is provided to the repeater (labeled "rate 1/q repetition") (Divsalar at 5). The repeater duplicates each of the *N* information bits *q* times and outputs the resulting $N \times q$ repeated bits (*id.*).

340. Divsalar also teaches "an [I]RA code as a set of parity checks … check nodes, randomly connected to the repeated message bits, enforce constraints that determine the parity bits" required by the Court's construction of this term. In particular, Divsalar's interleaver disposed between the repeater and accumulator satisfies the "randomly connected" portion of the Court's construction.[48] Further, the accumulator taught by Divsalar satisfies the "[I]RA code … enforce constraints that determine the parity bits." The '032 patent itself teaches that an accumulator satisfies those constraints and therefore Divsalar's accumulator satisfies them as well.

341. As explained above with reference to claim 1 of the '710 patent, the drawing below represents a Tanner graph for a simple version of the RA code taught in Divsalar. That Tanner graph is "a graph representing an [I]RA code as a set of parity checks where every message bit is repeated, [at least two different subsets of message bits are repeated a different number of times,] and check nodes, randomly

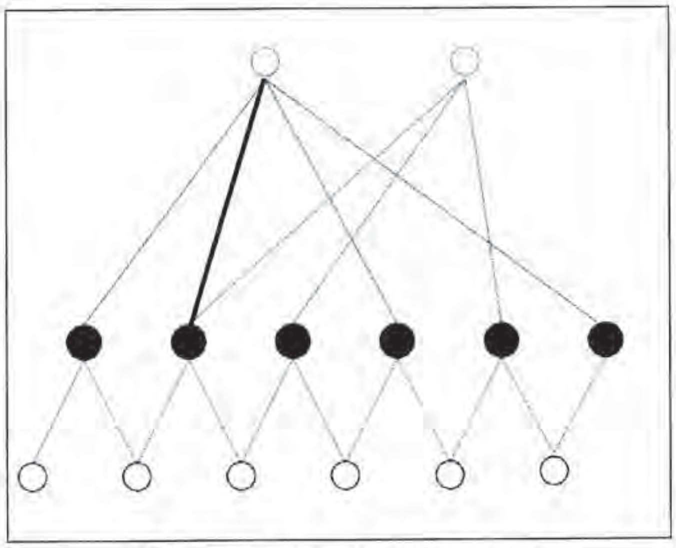
[48] Again, I have interpreted "randomly connected" consistent with Caltech's apparent application of the claim to DVB-S2 to mean implementation of an algorithm that may have been defined in advance using random operations.

connected to the repeated message bits, enforce constraints that determine the parity bits." That is, the Tanner graph of Divsalar's RA code meets all requirements imposed by the Court's construction of the Tanner graph term in claim 18, except that it is regular instead of irregular. In the Tanner graph below, the message bits are the two open circles at the top. They are both repeated twice. The check nodes are the black circles in the middle. They are randomly connected to the message bits. The open circles at the bottom represent parity bits. The connections between the check nodes and the parity bits enforce constraints that determine the parity bits. In particular, those constraints require the parity bits to be the accumulation of the repeated, interleaved message bits.



Tanner Graph of an RA Code (CALTECH000007326)

342. Only a single change is required to make the Tanner graph above meet all requirements imposed by the Court's construction of the Tanner graph limitation of claim 18. That is, if the repeat of the message bits is made irregular, e.g., by inserting one extra edge between one of the message bits and one of the check nodes (e.g., as shown by the extra red edge in the Tanner graph below) such that one message bit is repeated four times instead of three, then the Tanner graph meets all aspects of the Court's construction.



Tanner Graph of an IRA Code

ii. <u>The irregularity required by the Tanner graph limitation is taught by Frey99, Luby, and MacKay</u>

343. While Divsalar does not teach a scheme in which "at least two different subsets of message bits are repeated a different number of times" as required by the Court's construction, this limitation is taught by each of Frey99, Luby, and MacKay.

344. As I have explained above, Frey99 teaches the claimed irregular repetition. Also, as explained above, Luby and MacKay also teach the benefits of irregular codes.

e) *<u>Summary</u>*

345. As explained above, every limitation of claim 18 is taught by Divsalar except the irregularity required by the Court's construction of the Tanner graph limitation, which is taught by each of Luby, MacKay, and Frey99.

ii) <u>One of ordinary skill in the art would have been motivated to incorporate the irregularity of Frey99, Luby, or MacKay into the RA codes of Divsalar</u>

a) <u>*Combining the RA codes of Divsalar with the irregularity of Frey99, Luby, or MacKay, generally*</u>

346. For the reasons explained above with reference to the asserted claims of the '710 patent, it would have been obvious to incorporate irregularity (as motivated by Luby, MacKay or Frey99) into the repeat-accumulate codes taught by Divsalar.

b) <u>*Other similarities between Divsalar and each of Frey99, Luby, and MacKay further motivate the combination*</u>

347. The motivations to combine Divsalar with any one of Frey99, Luby, or MacKay references are strengthened by the fact that all four of these references teach "message passing" decoders, as required by claim 18.

348. As described above, Divsalar teaches a message passing decoder.

349. Frey99 teaches a "message passing" decoder as well. The irregular turbocodes of Frey99 are decoded using an interactive application of the sum-product algorithm, which is a type of message-passing decoder. Frey99 states: "[w]e construct irregular turbocodes with systematic bits that participate in varying number of trellis sections. These codes can be decoded by the iterative application of the sum-product algorithm (a low-complexity, more general form of the turbodecoding algorithm)" (Frey99 at 1). A person of ordinary skill in the art would recognize that an "iterative application of the sum-product algorithm," as described by Frey99, describes a "belief propagation" decoder, which is a type of "message passing" decoder (as shown by, *e.g.*, claim 22 of the '032 patent). *See*

also Frey Slides at 3 (showing non-elite and elite bits being "pinned down SLOWLY" and "pinned down QUICKLY," respectively) (emphasis in original).[49]

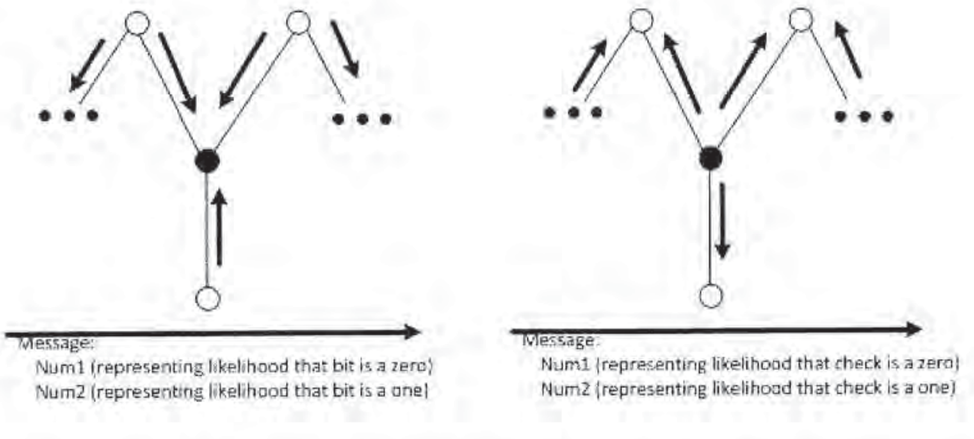
350. Conventional implementations of this kind of decoder consist of check/variable nodes operating in parallel to receive messages from neighboring check/variable nodes and send updated messages to the neighboring variable/check nodes, as described in the teaching of B. J. Frey, F. R. Kschischang, H.-A. Loeliger and N. Wiberg (presented at the Allerton Conference in September 1996, proceedings published in May 1997). The message sent from a variable node to a check node is comprised of one number for every possible value[50] of the variable, which is computed by taking the product of the corresponding messages received by the variable from other check nodes. The message sent from a check node to a variable node is comprised of one number for every possible value of the variable, which is computed by adding together terms that correspond to configurations of all other variables connected to the check node such that the parity check is satisfied, where each term is given by the product of the corresponding messages received by the check node from those variable nodes. An information bit is decoded by examining its corresponding variable node and for every possible value of the variable computing a number by taking the product of the corresponding messages received by the variable from all check nodes that it is connected to. The bit is decoded by setting it to the value with the largest number. There are variations on these operations that involve different ways of scaling the messages, different ways of scheduling the order in which messages are updated, different

[49] One of ordinary skill in the art would have recognized that "pinning down" bits refers to the operation of a message passing decoder, which "pins down" an information bit by iteratively applying the sum-product algorithm to the corresponding variable node.

[50] In binary coding systems, such as all of the references discussed herein, each variable can have only one of two possible values, i.e., 0 or 1. Thus, for any given information or parity bit, one number is computed representing the likelihood that the bit has value 0 and another number is computed representing the likelihood that the bit has value 1.

arithmetic operations that may be used, and different ways of more efficiently storing the numbers comprising the messages, all of which were known to those of ordinary skill before Caltech's alleged invention.

351. The drawing below graphically illustrates the above described operation of the message passing decoding algorithm using a small portion of a Tanner graph. In the drawings below, variable nodes corresponding to information and parity bits are the open circles at the top and bottom of the diagram, respectively, and the filled circle in the middle is a check node. In one cycle, as shown on the left, each variable node computes a message and sends its message to the check nodes to which it is connected. In the next cycle, as shown in the right, each check node computes a message and sends its message to the variable nodes to which it is connected. Several such iterations are performed, *e.g.*, until a solution stabilizes or until a maximum number of iterations has been reached. Then, each information or parity bit can combine multiple messages from its neighboring check nodes and use those messages to determine whether its value should be zero or one.


Message:
Num1 (representing likelihood that bit is a zero)
Num2 (representing likelihood that bit is a one)
Message:
Num1 (representing likelihood that check is a zero)
Num2 (representing likelihood that check is a one)

352. Luby also teaches message passing decoders, stating that "[t]o properly decode corrupted bits conventional *belief propagation* is utilized. *Belief propagation* is described in detail in "The Forward-Backward Algorithm" by G. David Forney, Jr. in Proceedings of the 34th Allerton Conference on

Communication, Control, and Computing (October, 1996), pp. 432-446" (Luby 18:29-38) (emphasis added). As explained above (and as confirmed by, *e.g.*, claim 22 of the '032 patent), a "belief propagation" decoder is a particular type of "message passing" decoder.

353. MacKay also teaches message passing decoders, teaching codes that "can be practically decoded with Gallager's sum-product algorithm giving near Shannon limit performance" (MacKay at 1449). As explained above, the "sum-product algorithm" refers to a particular type of message-passing decoder.

C. Claim 19 of the '032 Patent is Invalid

354. Claim 19 of the '032 patent reads:

> 19. The device of claim 18, wherein the message passing decoder is configured to decode the received data stream that includes the message bits.

355. Claim 19 is rendered obvious by Divsalar in combination with one of Frey99, Luby, or MacKay, alone or further in combination with Ping ((Divsalar + (Frey99, Luby or MacKay)) alone or (Divsalar + (Frey99, Luby or MacKay) + Ping)).

356. I explain above that Divsalar in combination with one of Frey99, Luby, or MacKay renders obvious every limitation of Claim 18.

357. Frey99 explicitly teaches transmitting the message bits as well as the parity bits. "In particular, if the bits in the convolutional code are partitioned into "systematic bits" and "parity bits", then by connecting each parity bit to a degree 1 codeword bit, we can encode in linear time" (Frey99 at 2). It was widely accepted at the time that "systematic bits" refers to message bits that are transmitted. MacKay also teaches transmitting message bits (*see* MacKay at Fig. 5, showing "[b]its $t_1 \ldots t_k$ defined to be source bits"). Further, both systematic and non-systematic codes were known long before Caltech's alleged invention. As an

example, Berrou's original disclosure of turbocodes was of a systematic code.[51] In Berrou's Figure 2, copied below, the arrow at the top pointing to the right represents transmission of the information bits, making the code a systematic code.

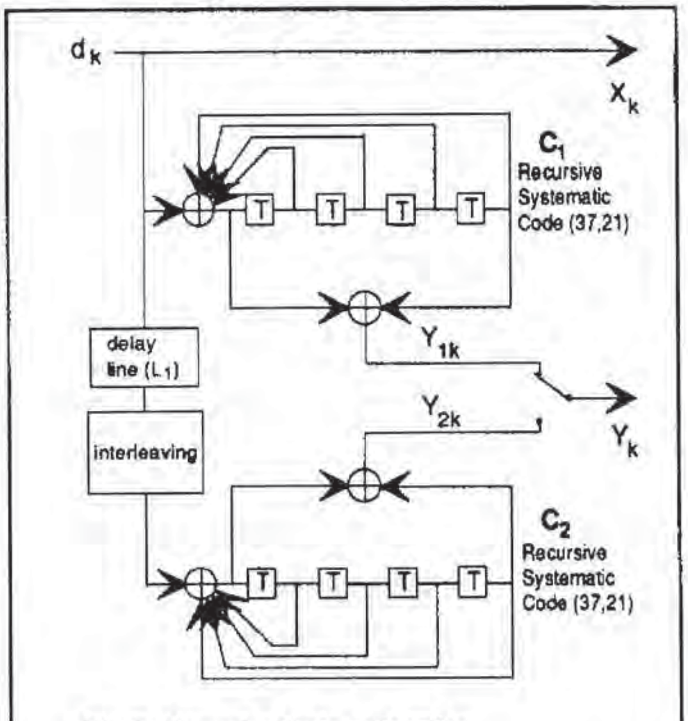
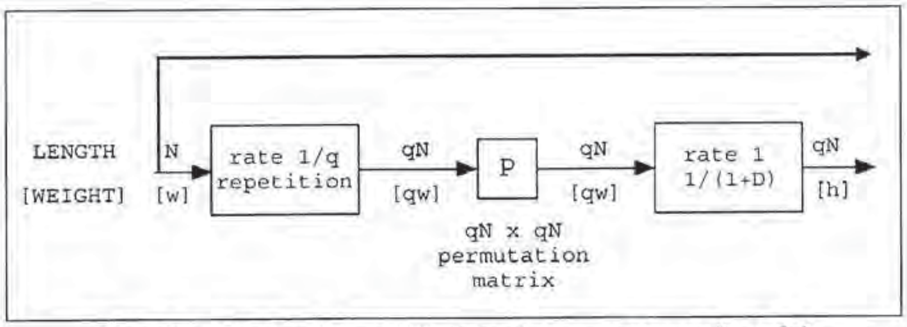


Fig. 2 Recursive Systematic codes with parallel concatenation.

Berrou, Figure 2

358. One of ordinary skill reading Divsalar or Luby would have understood that making the disclosed codes be systematic instead of non-systematic would have simple and obvious. Figure 3 of Divsalar is copied again below with an added red line to indicate the change to Divsalar that would make its code be systematic instead of non-systematic.[52]

[51] Claude Berrou et al., *Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes*, 2 IEEE International Conference on Communications, ICC '93 Geneva. Technical Program, Conference Record 1064 (1993); '032 patent, 1:29-56.
[52] This is consistent with the testimony of Dariush Divsalar (*see* Divsalar Tr. at 67-68).



Divsalar, Figure 3 (modified to show a systematic code)

359. While Divsalar and Luby do not explicitly teach decoding a received data stream that "includes the message bits" (*i.e.*, decoding a *systematic* code) this limitation would have been obvious to one of ordinary skill in view of those references alone. As explained above, Frey99 and MacKay both explicitly teach systematic codes.

360. Moreover, systematic codes are taught by Ping. Ping defines a "codeword **c** as **c** = [**p**, **d**], where **p** and **d** contain the parity and information bits, respectively" (Ping at 38). Ping goes on to provide equations from which "**p** = $\{p_i\}$ can easily be calculated from a given **d** = $\{d_i\}$" (*id.*). Thus, the codewords of Ping, which collectively comprise the "data stream" received by the decoder, include the information bits **d**.

361. As explained above, Frey99 and MacKay teach systematic codes, but additionally, it would have been obvious to one of ordinary skill in the art to modify any of Divsalar or Luby to make the code be systematic. Moreover, it would have further been obvious to incorporate Ping's teaching of systematic codes into Divsalar. Systematic codes had been well known in the art for decades prior to the claimed priority date of the patents-in-suit (*see*, *e.g.*, Wicker Dep. at 77:15-20). It would have been obvious to one of ordinary skill in the art that the techniques taught by Divsalar (as well as those taught by Frey99, Luby, and MacKay) can be applied equally to both systematic and non-systematic codes.

362. As described above, Divsalar, Frey99, Luby, and MacKay are directed to the same field, namely the field of error correcting codes, and specifically, variations and improvements on linear error-correcting codes that allow them to be encoded more quickly. Ping is related to the same field; Ping, titled "Low Density Parity Check Codes with Semirandom Parity Check Matrix," teaches constructing LDPC codes that can be encoded efficiently and have good BER vs. E_b/N_0 performance (*see* Ping at 39). Given that all four of these references relate to improvements to error-correcting codes, one of ordinary skill in the art would have been motivated to combine the teachings of Ping with those of Divsalar and at least one of Frey99, Luby, or MacKay.

363. Therefore, claim 19 is obvious over Divsalar in combination with one of Frey99, Luby, or MacKay, alone or further in combination with Ping.

D. Claim 22 of the '032 Patent is Invalid

364. Claim 22 of the '032 patent reads:

> 22. The device of claim 18, wherein the message passing decoder comprises a belief propagation decoder.

365. Claim 22 is rendered obvious by Divsalar in combination with one of Frey99, Luby, or MacKay.

366. I explain above that Divsalar in combination with any one of Frey99, Luby or MacKay teaches every limitation of claim 18.

367. The additional limitation imposed by claim 22 (*i.e.*, that the decoder comprise "a belief propagation decoder") is taught by Divsalar. Divsalar teaches that "an important feature of turbo-like codes is the availability of a simple ***iterative, message passing*** decoding algorithm that approximates ML decoding." (Divsalar at 9) (emphasis added). A person of ordinary skill in the art would recognize that the "iterative message passing" algorithm described in the above passage refers to a "belief propagation decoder."

368. As explained above with reference to claim 18, Frey99, Luby, and MacKay also teach belief propagation decoders, as required by claim 22. For example, Luby states that "[t]o properly decode corrupted bits conventional ***belief propagation*** is utilized. ***Belief propagation*** is described in detail in "The Forward-Backward Algorithm" by G. David Forney, Jr. in Proceedings of the 34th Allerton Conference on Communication, Control, and Computing (October, 1996), pp. 432-446" (Luby 18:29-38) (emphasis added).

369. Therefore, claim 22 is obvious over Divsalar in view of any one of Frey99, Luby, or MacKay.

VIII. THE ASSERTED CLAIMS OF THE '781 PATENT ARE INVALID

370. As I explain below, asserted claims 16 and 19 of the '781 patent are invalid. I also explain why claims 13, 14, and 15, from which claim 16 depends, are invalid. A summary of the opinions set forth in this section is given in the table below:

'781 Claim	Divsalar	Ping	Ping + MacKay	Divsalar + Frey99 (or Frey slides), or MacKay	Divsalar + Luby + (Ping, Frey99, MacKay or '999 Patent)
13	Anticipated / Obvious		Obvious	Obvious	Obvious (over Divsalar + Frey99)
14	Anticipated / Obvious		Obvious	Obvious	Obvious (over Divsalar + Frey99)
15	Obvious		Obvious	Obvious	Obvious
16	Obvious		Obvious	Obvious	Obvious
19	Anticipated / Obvious	Anticipated	Anticipated (by Ping)	Anticipated (by Divsalar)	Anticipated / Obvious (by Divsalar and Ping)

A. Claim 13 of the '781 Patent is Invalid

371. Claim 13 of the '781 patent reads:

> 13. A method of encoding a signal, comprising:
>
> receiving a block of data in the signal to be encoded, the block of data including information bits; and
>
> performing an encoding operation using the information bits as an input, the encoding operation including an accumulation of mod-2

or exclusive-OR sums of bits in subsets of the information bits, the encoding operation generating at least a portion of a codeword, wherein the information bits appear in a variable number of subsets.

i) <u>Claim 13 of the '781 Patent is Anticipated by Divsalar</u>

372. I explain below, one limitation at a time, why claim 13 is anticipated by Divsalar.

a) <u>*"A method of encoding a signal"*</u>

373. Even if the preamble, "[a] method of encoding a signal," limits the claim, it is taught by Divsalar, as explained above with reference to the asserted claims of the '710 patent.

b) <u>*"receiving a block of data in the signal to be encoded, the block of data including information bits"*</u>

374. Divsalar teaches this limitation. As explained above with reference to the '710 patent, Divsalar deals exclusively with block codes. The repeat-accumulate codes introduced by Divsalar are encoded by receiving an "information block of length *N*" and passing the block to the repeater. *See* Divsalar at 5 ("[a]n information block of length *N* is repeated *q* times…") and, for example, Figure 3, reproduced above. Divsalar refers to the input block as an "information block" because it includes information bits.

c) <u>*"performing an encoding operation using the information bits as an input, the encoding operation including an accumulation of mod-2 or exclusive-OR sums of bits in subsets of the information bits, the encoding operation generating at least a portion of a codeword"*</u>

375. Divsalar teaches this limitation. As explained above, Figure 3 of Divsalar depicts an encoder that is operable to perform an encoding operation in which an "information block of length *N*" is fed into a repeater, which repeats each of the *N*

information bits q times, producing a total of $N \times q$ repeated bits. This process is illustrated below for $N = 5$, and $q = 3$:[53]

$$\underbrace{i_1, i_2, i_3, i_4, i_5}_{N \text{ information bits}} \longrightarrow \boxed{\textbf{Repeater}} \longrightarrow \underbrace{i_1, i_1, i_1, i_2, i_2, i_2, i_3, i_3, i_3, i_4, i_4, i_4, i_5, i_5, i_5}_{N \times q \text{ repeated information bits}}$$

Example of Repetition as Described in Divsalar, with N=5 and q=3

376. Figure 3 of Divsalar, reproduced above, shows a "permutation matrix" (the box labeled "P"). After the repeater duplicates each of the N information bits q times and outputs $N \times q$ repeated bits, the repeated bits are "scrambled by an interleaver of size qN" (Divsalar at 5). Continuing the example above, with $N = 5$ and $q = 3$, the interleaving process may be illustrated as follows:

$$\underbrace{i_1, i_1, i_1, i_2, i_2, i_2, i_3, i_3, i_3, i_4, i_4, i_4, i_5, i_5, i_5}_{N \times q \text{ repeated information bits}} \longrightarrow \boxed{\textbf{Interleaver}} \longrightarrow \underbrace{i_4, i_5, i_1, i_3, i_1, i_2, i_4, i_2, i_1, i_3, i_2, i_5, i_5, i_3, i_4}_{N \times q \text{ repeated, scrambled information bits}}$$

Example of Interleaving as Described in Divsalar, with N=5 and q=3

The example interleaving shown in the figure above illustrates just one of many permutations that may be used to scramble the repeated information bits. One of ordinary skill in the art would understand that the interleaver shown in Figure 3 of Divsalar would be prefabricated or preprogrammed to implement one of the possible permutations.

377. As explained above, the repeated, scrambled information bits are then *accumulated*. Divsalar explains the accumulate step as follows:

> [W]e prefer to think of [the accumulator] as a block coder whose input block $[x_1, \ldots, x_n]$ and output block $[y_1, \ldots, y_n]$ are related by the formula
>
> $y_1 = x_1$
>
> $y_2 = x_1 + x_2$

[53] Divsalar explicitly discloses, *e.g.*, in Figure 5, several values for N and q. Divsalar's values of N are much larger than 5, but I have used $N = 5$ in this example for ease of exposition. RA codes with $q = 3$ are explicitly disclosed by Divsalar.

$$y_3 = x_1 + x_2 + x_3$$
$$y_n = x_1 + x_2 + x_3 + \ldots + x_n.$$

(Divsalar at 5)

378. This accumulation operation operates on "sums of bits in subsets of the information bits." To explain why this is the case, I will continue the example above (with $N = 5$ and $q = 3$).

379. In our example, $N = 5$ and $q = 3$, so the accumulator will accept 15 x bits as input, and produce 15 y bits as output. The excerpt from Divsalar above provides explicit equations for y_1, y_2, and y_3, and a general equation for each y_n thereafter. Writing these equations out explicitly for each of the 15 y bits yields the following 15 equations:

$$y_1 = x_1$$
$$y_2 = x_1 + x_2$$
$$y_3 = x_1 + x_2 + x_3$$
$$y_4 = x_1 + x_2 + x_3 + x_4$$
$$y_5 = x_1 + x_2 + x_3 + x_4 + x_5$$
$$y_6 = x_1 + x_2 + x_3 + x_4 + x_5 + x_6$$
$$y_7 = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7$$
$$y_8 = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8$$
$$y_9 = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9$$
$$y_{10} = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10}$$
$$y_{11} = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11}$$
$$y_{12} = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12}$$
$$y_{13} = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13}$$
$$y_{14} = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14}$$
$$y_{15} = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15}$$

380. As explained above, the x bits taught by Divsalar are repeated, interleaved information bits. Each x bit (such as, *e.g.*, x_3) is a duplicate of one of the information bits. For example, continuing our interleaving example above, the correspondence between x bits and i bits is as follows:

$$\boxed{\begin{array}{ccccccccccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} & x_{13} & x_{14} & x_{15} \\ \Downarrow & \Downarrow & \Downarrow & \Downarrow & \Downarrow & \Downarrow & \Downarrow & \Downarrow & \Downarrow & \Downarrow & \Downarrow & \Downarrow & \Downarrow & \Downarrow & \Downarrow \\ i_4 & i_5 & i_1 & i_3 & i_1 & i_2 & i_4 & i_2 & i_1 & i_3 & i_2 & i_5 & i_5 & i_3 & i_4 \end{array}}$$

Correspondence between repeated, scrambled information bits *x* and information bits *i*

As the figure above shows, each *i* bit corresponds to exactly 3 *x* bits (because every information bit is duplicated q=3 times before interleaving). For example, the three duplicates of the 4th information bit i_4 are x_1, x_7, and x_{15}.

381. Continuing our example, we can substitute each of the *x* variables in the 15 equations above for the corresponding *i* bit, yielding:

$$\begin{aligned}
y_1 &= i_4 \\
y_2 &= i_4 + i_5 \\
y_3 &= i_4 + i_5 + i_1 \\
y_4 &= i_4 + i_5 + i_1 + i_3 \\
y_5 &= i_4 + i_5 + i_1 + i_3 + i_1 \\
y_6 &= i_4 + i_5 + i_1 + i_3 + i_1 + i_2 \\
y_7 &= i_4 + i_5 + i_1 + i_3 + i_1 + i_2 + i_4 \\
y_8 &= i_4 + i_5 + i_1 + i_3 + i_1 + i_2 + i_4 + i_2 \\
y_9 &= i_4 + i_5 + i_1 + i_3 + i_1 + i_2 + i_4 + i_2 + i_1 \\
y_{10} &= i_4 + i_5 + i_1 + i_3 + i_1 + i_2 + i_4 + i_2 + i_1 + i_3 \\
y_{11} &= i_4 + i_5 + i_1 + i_3 + i_1 + i_2 + i_4 + i_2 + i_1 + i_3 + i_2 \\
y_{12} &= i_4 + i_5 + i_1 + i_3 + i_1 + i_2 + i_4 + i_2 + i_1 + i_3 + i_2 + i_5 \\
y_{13} &= i_4 + i_5 + i_1 + i_3 + i_1 + i_2 + i_4 + i_2 + i_1 + i_3 + i_2 + i_5 + i_5 \\
y_{14} &= i_4 + i_5 + i_1 + i_3 + i_1 + i_2 + i_4 + i_2 + i_1 + i_3 + i_2 + i_5 + i_5 + i_3 \\
y_{15} &= i_4 + i_5 + i_1 + i_3 + i_1 + i_2 + i_4 + i_2 + i_1 + i_3 + i_2 + i_5 + i_5 + i_3 + i_4
\end{aligned}$$

382. In modulo-2 addition, adding a bit to itself always results in 0; that is, two identical information bits cancel each other out. Therefore, if an information bit appears an even number of times in one of the equations above, it cancels out entirely, and if it appears an odd number of times, it effectively appears only once. Performing these cancellations yields:

Explicit Equation	**Recursive Equation**
$y_1 = i_4$	$y_1 = i_4$
$y_2 = i_4 + i_5$	$y_2 = y_1 + i_5$
$y_3 = i_4 + i_5 + i_1$	$y_3 = y_2 + i_1$
$y_4 = i_4 + i_5 + i_1 + i_3$	$y_4 = y_3 + i_3$
$y_5 = i_4 + i_5 + i_3$	$y_5 = y_4 + i_1$
$y_6 = i_4 + i_5 + i_3 + i_2$	$y_6 = y_5 + i_2$
$y_7 = i_5 + i_3 + i_2$	$y_7 = y_6 + i_4$
$y_8 = i_5 + i_3$	$y_8 = y_7 + i_2$
$y_9 = i_5 + i_3 + i_1$	$y_9 = y_8 + i_1$
$y_{10} = i_5 + i_1$	$y_{10} = y_9 + i_3$
$y_{11} = i_5 + i_1 + i_2$	$y_{11} = y_{10} + i_2$
$y_{12} = i_1 + i_2$	$y_{12} = y_{11} + i_5$
$y_{13} = i_1 + i_2 + i_5$	$y_{13} = y_{12} + i_5$
$y_{14} = i_1 + i_2 + i_5 + i_3$	$y_{14} = y_{13} + i_3$
$y_{15} = i_1 + i_2 + i_5 + i_3 + i_4$	$y_{15} = y_{14} + i_4$

383. The above table includes explicit and recursive equations for each of the *y* bits. The explicit equations on the left show that the *y* bits are "sums of bits in subsets of the information bits." For example, y_{14} is the sum of bits in the subset of the information bits containing i_1, i_2, i_5 and i_3. The recursive equations on the right illustrate that the process of computing the *y* bits using an "accumulation" operation in which each of the *y* bits (except the first) is the sum of the previous *y* bit and an *i* bit.

384. Therefore, as this example shows, the encoding shown in Figure 3 of Divsalar is an accumulation of "sums of bits in subsets of the information bits" (*i.e.*, the subsets of the *i* bits that appear in each of the explicit equations).

385. Because all of the variables in the above equations are bits, the "+" sign in Divsalar represents modulo-2 addition, or "exclusive-OR," as required by this

limitation. Further, because the *y* bits are the output of the encoding process, accumulation is the last step in the encoding process, the *y* bits form a codeword.

386. Therefore, for the reasons explained above, Divsalar teaches "performing an encoding operation using the information bits as an input, the encoding operation including an accumulation of mod-2 or exclusive-OR sums of bits in subsets of the information bits, the encoding operation generating at least a portion of a codeword," as required by claim 13 of the '781 patent.

387. Alternatively, a punctured version of the Divsalar code would satisfy the requirement of claim 13 that "the encoding operation include[e] an accumulation of mod-2 or exclusive-OR sums of bits in subsets of the information bits, the encoding operation generating at least a portion of a codeword." As explained above, the accumulation of Divsalar can be represented using the following equations:

$$
\begin{aligned}
&y_1 = x_1\\
&y_2 = x_1 + x_2\\
&y_3 = x_1 + x_2 + x_3\\
&y_4 = x_1 + x_2 + x_3 + x_4\\
&y_5 = x_1 + x_2 + x_3 + x_4 + x_5\\
&y_6 = x_1 + x_2 + x_3 + x_4 + x_5 + x_6\\
&y_7 = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7\\
&y_8 = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8\\
&y_9 = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9\\
&y_{10} = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10}\\
&y_{11} = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11}\\
&y_{12} = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12}\\
&y_{13} = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13}\\
&y_{14} = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14}\\
&y_{15} = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15}
\end{aligned}
$$

Outputting some, but not all of the parity bits y_i would result in a code in which each parity bit is the sum of the previous parity bit and the sum of the bits in a subset of the information bits. For example, if only y_1, y_7, y_9, *and* y_{15} were output by the encoder, the resulting code could be described using the following 4 equations:

$$y_1 = x_1$$
$$y_7 = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7$$
$$y_9 = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9$$
$$y_{15} = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15}$$

Substituting the explicit equations from the table above would yield

$$y_1 = i_4$$
$$y_7 = i_5 + i_3 + i_2$$
$$y_9 = i_5 + i_3 + i_1$$
$$y_{15} = i_1 + i_2 + i_5 + i_3 + i_4$$

In other words,

$$y_1 = i_4$$
$$y_7 = y_1 + (i_4 + i_5 + i_3 + i_2)$$
$$y_9 = y_7 + (i_2 + i_1)$$
$$y_{15} = y_9 + (i_4 + i_2)$$

Here, each parity bit, (*e.g.*, y_9) is calculated as the accumulation of the previous parity bit (*i.e.*, y_7) and the sum of a subset of the information bits (*i.e.*, $i_2 + i_1$). Outputting only some of the parity bits of Divsalar, while omitting others, is a technique called "puncturing," that was well known in the art by Caltech's alleged conception date (*see*, *e.g.*, Frey99 at 3, "… some extra parity bits must be punctured").

d) *"wherein the information bits appear in a variable number of subsets"*

388. As explained above, the encoding operation of Divsalar includes an accumulation of mod-2 sums of bits in subsets of information bits. One example

of such an accumulation, for $N = 5$, $q = 3$ and a particular interleaver, is given by the explicit equations given in the table above, namely:

$y_1 = i_4$
$y_2 = i_4 + i_5$
$y_3 = i_4 + i_5 + i_1$
$y_4 = i_4 + i_5 + i_1 + i_3$
$y_5 = i_4 + i_5 + i_3$
$y_6 = i_4 + i_5 + i_3 + i_2$
$y_7 = i_5 + i_3 + i_2$
$y_8 = i_5 + i_3$
$y_9 = i_5 + i_3 + i_1$
$y_{10} = i_5 + i_1$
$y_{11} = i_5 + i_1 + i_2$
$y_{12} = i_1 + i_2$
$y_{13} = i_1 + i_2 + i_5$
$y_{14} = i_1 + i_2 + i_5 + i_3$
$y_{15} = i_1 + i_2 + i_5 + i_3 + i_4$

Here, each bit *y* of the codeword is the sum of a different subset of information bits (denoted using the letter *i*).

389. As these equations show, different information bits appear in different numbers of subsets. For example, the information bit i_4 appears in seven of the equations above, while the information bit i_1 appears in nine of the equations.[54] Also, while the equations above result from using an interleaver that scrambles bits according to one particular permutation, different information bits will appear in different numbers of subsets no matter how the bits are permuted by the interleaver. In particular, for some of the example values of *N* and *q* explicitly disclosed in Divsalar in Figure 5, *e.g.*, $N = 1024$ and $q = 3$, regardless of what interleaver is used, the information bits will appear in a variable number of subsets. Thus,

[54] Because the "subsets" are the groups of *i* bits that appear on the right-hand side of each of the equations above, if an *i* bit appears in an equation, it is a member of the corresponding subset.

Divsalar teaches that the information bits appear in variable numbers of subsets, as required by the claim.

390. This limitation also holds for the "punctured" version of Divsalar discussed above. For example, it holds for the example given above, where the parity bits are represented by the equations:

$$y_1 = i_4$$
$$y_7 = y_1 + (i_4 + i_5 + i_3 + i_2)$$
$$y_9 = y_7 + (i_2 + i_1)$$
$$y_{15} = y_9 + (i_4 + i_2)$$

Here, the information bits appear in a variable number of subsets. The information bit i_4, for example, appears in three subsets, while the information bit i_5 appears in only one.

e) <u>*Summary*</u>

391. As explained above, Divsalar teaches every limitation of claim 13 and therefore anticipates claim 13.

ii) <u>Claim 13 of the '781 Patent is Obvious over Divsalar in View any one of Frey99, Luby, or MacKay</u>

392. As explained above, Divsalar teaches every limitation of, and therefore anticipates, claim 13 of the '781 patent. However, in the event Divsalar is found not to teach the "wherein the information bits appear in a variable number of subsets" limitation of claim 13, then claim 13 is obvious over the combination of Divsalar and any one of Frey99, Luby or MacKay.

393. Specifically, if the term "wherein the information bits appear in a variable number of subsets," is interpreted to require that the claimed encoding method be

irregular, then each of Frey99, Luby, and MacKay teaches this limitation.[55] As I explain above, one of ordinary skill in the art would have been motivated to combine Divsalar and Frey99, Luby or MacKay in general, and would specifically have been motivated to use the irregular repetition of Frey99, or the irregularity of Luby and MacKay, with the RA codes of Divsalar. I also explain why such a combination would represent a minor modification to the teachings of Divsalar, and would not fundamentally change its principle of operation.

394. For at least the reasons given above, claim 13 is obvious over the combination of Divsalar and any one of Frey99, Luby and MacKay.

iii) Claim 13 of the '781 Patent is Obvious over Ping in View of MacKay

395. I explain below, one limitation at a time, why claim 13 is rendered obvious by Ping in combination with MacKay.[56]

a) "A method of encoding a signal, comprising ..."

396. Ping teaches the preamble. As I explain above, Ping teaches constructing LDPC codes that can be encoded in two stages. In the first encoding stage, a generator matrix is applied to a sequence of information bits to produce sums of information bits. In the second stage, the sums of information bits are accumulated recursively to generate the parity bits (*see* Ping at 38).

[55] As noted above, during prosecution of the '781 patent, the applicant edited claims 9 and 23 effectively replacing "irregular" with "variable number of subsets." Response dated January 27, 2011, at 3 and 5-6. In its remarks regarding that amendment, the applicant stated, "It is believed that the meaning of the term 'irregular' in the claims is clear and is well known in the art… However, claims have been amended to recite '…wherein the information bits appear in a variable number of subsets' to obviate the objections." Response dated January 27, 2011, at 7 (emphasis original). In view of this file history, Caltech may argue that "variable number of subsets" requires irregularity. However, the applicant may have simply been broadening the claims when it replaced "irregular" with "variable number of subsets." In any case, the claims do not clearly require irregularity. However, for the sake of completeness, I have addressed the term under the two possible interpretations herein, one in which irregularity is required and one in which it is not.

[56] *See generally* Divsalar Tr. at 61:15-71:11.

397. A person of ordinary skill in the art would recognize that encoding/decoding signals is the purpose of the "LDPC + accumulate" codes taught by Ping.

b) *"receiving a block of data in the signal to be encoded, the block of data including information bits"*

398. Ping teaches this limitation. Ping deals exclusively with block codes, as I explain above. Specifically, Ping denotes the block of information bits to be encoded using the vector variable name **d** (*see* Ping at 38, "[d]ecompose the codeword **c** as **c** = [**p**, **d**], where **p** and **d** contain the parity and information bits, respectively"). Ping goes on to provide equations from which "**p** = $\{p_i\}$ can easily be calculated from a given **d** = $\{d_i\}$;" That is, Ping provides equations that describe the process of *encoding* a block of information bits **d**, as required by this limitation (*id.*). Thus, the vector of information bits **d** is a "block of data in the signal to be encoded, the block of data including information bits."

c) *"performing an encoding operation using the information bits as an input, the encoding operation including an accumulation of mod-2 or exclusive-OR sums of bits in subsets of the information bits, the encoding operation generating at least a portion of a codeword"*

399. Ping teaches this limitation. Specifically, Ping teaches an encoding operation that calculates the parity bits $\{p_i\}$ using the information bits $\{d_i\}$ as input. Ping's encoding scheme is encapsulated by the following equations:

$$\boldsymbol{p_1 = \sum_j h^d_{1j} d_j}$$

$$\boldsymbol{p_i = p_{i-1} + \sum_j h^d_{ij} d_j}$$

(Ping at 38)

400. In these equations, the variable h^d_{ij} represents the value at the i^{th} row and the j^{th} column of the parity check matrix $\mathbf{H}^{\mathbf{d}}$, and the variable d_j represents the value of the j^{th} information bit (*see id.*). Thus, the summation

$$\sum_j h^d_{ij} d_j$$

represents the sum of the bits in a subset of information bits. Specifically, it represents the sum of the subset of information bits d_j where $h^d_{ij} = 1$. As Ping explains, there are $kt/(n\text{-}k)$ information bits in each row of the parity check matrix, meaning that there $kt/(n\text{-}k)$ bits in each subset of the information bits (*id.*).

401. Further, the encoding taught by Ping includes an accumulation of these sums of bits in subsets of the information bits. The first parity bit of Ping, p_1, is calculated as the sum of a subset of information bits. As illustrated in the color-coded equation below, each subsequent parity bit p_i is calculated by adding together the previous parity bit p_{i-1} (shown in blue) and the sum of bits in a subset of information bits (shown in red):

$$p_i = p_{i-1} + \sum_j h^d_{ij} d_j$$

I explain above that this type of operation, in which each new element is calculated by adding something to the previous element, is called an "accumulation."

402. Also, the addition taught by Ping is modulo-2 or "mod-2" addition (which is the same operation as "exclusive-OR"). When the addition symbol "+"and the summation symbol "Σ" have bits as operands, as they do here, one of ordinary skill

in the art would understand that these symbols refer to modulo-2 addition and modulo-2 summation, respectively.

403. When complete, this process produces the parity bits $\{p_i\}$ which, as Ping explains, is a portion of the codeword "**c** = [**p**, **d**], where **p** and **d** contain the parity and information bits, respectively" (Ping at 38).

d) *"wherein the information bits appear in a variable number of subsets"*

404. As explained above, MacKay teaches implementing parity-check matrices in which every information bit corresponds to a column, where the weight of that column (*i.e.*, the number of 1s contained in that column of the parity-check matrix) represents the degree of the information bit.

405. As explained above, MacKay teaches parity-check matrices for which each column corresponds to an information bit or a parity bit, and each row corresponds to a parity check: "[t]he parity check matrix of a code can be viewed as defining a bipartite graph with 'bit' vertices corresponding to the columns and 'check' vertices corresponding to the rows. Each nonzero entry in the matrix corresponds to an edge connecting a bit to a check. The profile specifies the degrees of the vertices in this graph" (MacKay at 1449-1450).

406. Thus, each row in the parity-check matrices of MacKay corresponds to a subset of information bits that are summed during the encoding process. In a given row, if the entry corresponding to an information bit is a 1, that information bit is a member of the subset. If the entry corresponding to an information bit is 0, the information bit is not a member of the subset.

407. In the parity-check matrices of MacKay, the number of ones in a column that corresponds to an information bit (*i.e.*, the column weight) equals to the number of times that information bit appears in a subset. MacKay also notes that

"[t]he best known binary Gallager codes are *irregular* codes whose parity check matrices have *nonuniform* weight per column," meaning that the best codes are those in which the information bits appear in a variable number of subsets (Mackay at 1449) (emphasis in original).

e) <u>*Summary*</u>

408. As explained above, the combination of Ping and Mackay teaches every limitation of claim 13 of the '781 patent.

f) <u>*Motivations to Combine the teachings of Ping with those of MacKay*</u>

409. As I explain above, it would have been obvious to one of ordinary skill in the art to combine the LDPC-accumulate coders of Ping with the irregularity of MacKay. Specifically, it would have been obvious to one of ordinary skill in the art to incorporate irregularity into the LDPC-accumulate coders of Ping by making the column weights of the parity check matrix $\mathbf{H}^{\mathbf{d}}$ that correspond to information bits nonuniform, resulting in a code in which "the information bits appear in a variable number of subsets," as required by claim 13.

410. It would have been obvious to incorporate MacKay's irregularity into Ping because the two codes are so similar and it was known that irregularity improves coding as explained above. Alternatively, it would have been obvious to incorporate Ping's accumulate stage into MacKay. MacKay's irregular LDPC code teaches all limitations of claim 13 of the '781 patent except the "accumulation" limitation (i.e., as explained above, MacKay teaches (a) block codes and therefore teaches the "receiving" limitation, (b) LDPC codes and therefore teaches computing parity bits that are sums of bits in subsets of the information bits, and (c) irregular codes and therefore teaches the "information bits appear in a variable number of subsets" limitation). However, both Ping and Divsalar taught the benefit of adding an accumulation stage to an outer encoder. Thus, regardless of

whether one of ordinary skill incorporated MacKay's irregularity into Ping or Ping's accumulator into MacKay, the combination of Ping and MacKay renders claim 13 of the '781 patent obvious.

B. Claim 14 of the '781 Patent is Invalid

411. Claim 14 of the '781 patent reads:

> 14. The method of claim 13, further comprising: outputting the codeword, wherein the codeword comprises parity bits.

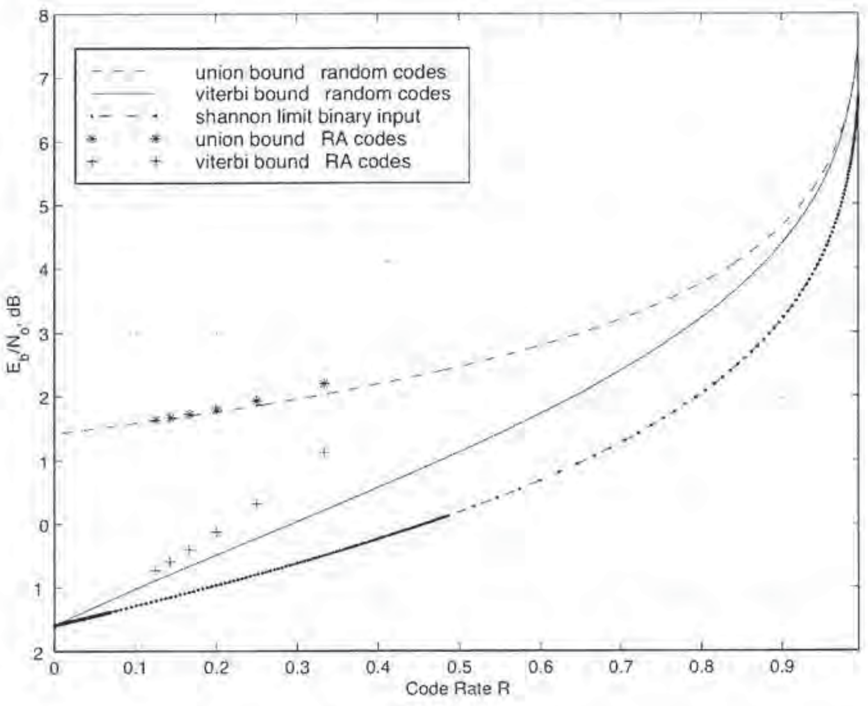
a) *Claim 14 is Anticipated by Divsalar, and rendered obvious by a combination of Divsalar and Frey99, Luby, or MacKay*

412. Claim 14 is anticipated by Divsalar, and is obvious over Divsalar in view of any one of Frey99, Luby or MacKay. As I explain above, claim 13 of the '781 patent is anticipated by Divsalar, and rendered obvious by a combination of Divsalar and any one of Frey99, Luby, or MacKay. Claim 14 adds to claim 13 "outputting the codeword, wherein the codeword comprises parity bits." Divsalar, Frey99, Luby, and MacKay all teach methods of encoding signals that comprise outputting a codeword that comprises parity bits.

413. As explained above, the encoder shown in Figure 3 of Divslar, reproduced above, produces an a "***output*** block [$y_1, \ldots, y_n$]" of parity bits that is included in the codeword (Divsalar at 5) (emphasis added).

414. Divsalar also describes the performance of the RA codes it teaches by graphing the code rate R against the normalized signal-to-noise ratio E_b/N_0:[57]

[57] The normalized signal-to-noise ratio E_b/N_0 is described in detail above and in Appendix A.



Divsalar, Figure 4

The concept of E_b/N_0 only makes sense in the context of outputting codewords, transmitting them over a noisy channel, and decoding them at the other end.

415. As described above, the irregular turbocoding techniques taught by Frey99 also involve outputting a codeword that comprises parity bits (*see generally,* Frey99 at 1-4). *See also* Frey Slides at 4-5. Like Divsalar, Frey99 includes experimental results; it includes a plot of BER against E_b/N_0 for various irregular turbocodes (*see* Frey99, Figure 4). *See also* Frey Slides at 9, 11, 12 (showing BER-$E_b/\underline{N}_0$ curves). As explained above, Luby and MacKay also teach outputting codewords that include parity bits (*see*, *e.g.*, Luby at 1:46-60, MacKay at Fig. 5).

b) Claim 14 is Rendered Obvious by a combination of Ping and MacKay

416. Claim 14 is rendered obvious by a combination of Ping and MacKay. As I explain above, claim 13 of the '781 patent is rendered obvious by a combination of Ping and MacKay. Claim 14 adds to claim 13 "outputting the codeword, wherein

the codeword comprises parity bits." Luby and MacKay also both teach outputting a codeword that comprises parity bits (*see*, *e.g.*, Luby at 1:46-60, MacKay at Fig. 5).

417. Ping teaches "outputting" the codeword. Like Divsalar and Frey99, Ping describes the performance of the codes it discloses using plots of the BER of various codes against the normalized signal-to-noise ratio E_b/N_0:

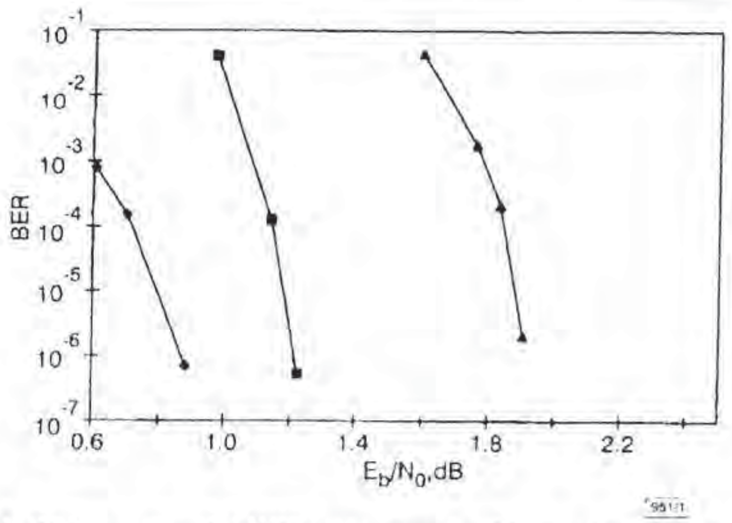


Fig. 1 *Performances of LDPC codes generated by semi-random parity check matrixes with k = 30000*

◆ $R = 1/3$
■ $R = 1/2$
▲ $R = 2/3$

Ping, Figure 1

As I explain above, the concepts of "BER" and E_b/N_0 only make sense in the context of generating codewords, transmitting them over a noisy channel, and decoding them at the other end.

418. Further, as I explain above, Ping teaches a coding scheme in which the codeword includes both information and parity bits.[58] Specifically, Ping teaches that "the codeword **c** as **c** = [**p**, **d**], where **p** and **d** contain the parity and information bits, respectively" (Ping at 38).

[58] Such codes are called *systematic codes*.

419. Therefore, claim 14 is rendered obvious by a combination of Ping and MacKay.

C. Claim 15 of the '781 Patent is Invalid

420. Claim 15 of the '781 patent reads:

> 15. The method of claim 14, wherein outputting the codeword comprises: outputting the parity bits; and outputting at least some of the information bits.

421. Claim 15 is rendered obvious by Divsalar alone or in combination with Ping, Frey99, MacKay or the '999 patent, and is also obvious over Divsalar in view of Luby, alone or in further in view of either Ping, Frey99, MacKay or the '999 patent (i.e., (Divsalar + Luby alone or (Divsalar + Luby + (Ping, Frey99, MacKay or the '999 patent)). As I explain above, claim 14 of the '781 patent is anticipated by Divsalar, and obvious over Divsalar in view of any one of Frey99, Luby, or MacKay. Claim 15 adds to claim 14 "wherein outputting the codeword comprises: outputting the parity bits; and outputting at least some of the information bits." That is, a systematic code would teach the limitations added by claim 15 and as explained above, systematic codes were known long before Caltech's alleged invention. Frey99 and MacKay teach systematic codes (*see* Frey99 at 3; MacKay at Fig. 5, showing "[b]its $t_1 \ldots t_k$ defined to be source bits") and it would have been obvious to make the codes of Divsalar or Luby systematic. Also, the additional limitation of claim 15 is taught explicitly by each of Ping and the '999 patent.

422. As explained above, Ping teaches a systematic code wherein "outputting the codeword comprises: outputting the parity bits; and outputting at least some of the information bits."

423. Also, as I explain above in the context of the '032 patent, one of ordinary skill in the art would have been motivated to combine the

teachings of Divsalar, Frey99, Luby or MacKay, and Ping. Therefore, claim 15 is rendered obvious by Divsalar in combination with Ping, and is also rendered obvious by the following combinations:

- Divsalar alone (systematic codes being well known) or in combination with Ping (Ping teaching the systematic code)
- Divsalar in combination with Frey99 or MacKay (Frey99 and MacKay each teaching both irregularity and systematic codes)
- Divsalar in combination with Luby (Luby teaching irregularity and systematic codes being well known)
- Divsalar in combination with Luby and Ping (Luby teaching irregularity and Ping teaching systematic codes)

424. The '999 patent also teaches a systematic code in which "outputting the codeword comprises: outputting the parity bits; and outputting at least some of the information bits":

> Suitable ***codewords*** for such schemes have been generated in a variety of ways. For systematic encoding of cyclic codes, one such method utilizes serial data input/output wherein each information word is applied to an (n-k)-stage shift register with feedback connections based on a generator polynomial. ***After the information bits are shifted into the register and simultaneously into the communication channel, the n-k parity bits formed in the register are shifted into the channel, thus forming the complete codeword.***
>
> ('999 Patent at 1:25-34) (emphasis added).

As indicated by the passage above, the '999 patent teaches encoding schemes in which a "complete codeword" includes both the "information bits" and "the n-k parity bits."

425. Like Divsalar, Frey99, Luby, and MacKay, the '999 patent relates to methods of improving the performance of linear error-correcting codes. It was filed in 1984 and granted in 1986, well over a decade before the claimed priority date of the patents-in-suit. By March 7, 2000, the alleged conception date of the patents-in-suit, the technology described in the '999 patent would have been well

known in the field, and one of ordinary skill in the art at the time would have been motivated to combine the teachings of Divsalar and Frey99, Luby, or MacKay with those of the '999 patent. Therefore, claim 15 is rendered obvious by Divsalar in combination with the '999 patent, and is also rendered obvious by a combination of Divsalar, Frey99, Luby, or MacKay, and the '999 patent.

426. Summarizing, systematic codes were notoriously well known before Caltech's alleged invention. Ping, Frey99, MacKay and the '999 patent are examples of references that teach systematic codes. It would have been obvious to make a code like the ones taught in Divsalar or Luby systematic in view of the general knowledge of one of ordinary skill, *e.g.*, as exemplified by Ping, Frey99, MacKay and the '999 patent.

427. Claim 15 of the '781 patent is also rendered obvious by a combination of Ping and MacKay. As I explain above, Ping and MacKay teach every element of claim 14, and Ping and MacKay also teach the additional limitation imposed by claim 15. Therefore, claim 15 is obvious over a combination of Ping and MacKay.

D. Claim 16 of the '781 Patent is Invalid

428. Claim 16 of the '781 patent reads:

> 16. The method of claim 15, wherein the parity bits follow the information bits in the codeword.

429. Claim 16 is obvious over Divsalar in view of either Ping, Frey99, MacKay or the '999 patent, and is also obvious over Divsalar in view of Luby, alone or further in view of either Ping, Frey99, MacKay or the '999 patent (i.e., (Divsalar + Luby alone or (Divsalar + Luby + (Ping, Frey99, MacKay or the '999 patent)).

430. As I explain above, claim 15 of the '781 patent is rendered obvious by Divsalar alone or in combination with references that teach irregularity (Frey99, MacKay or Luby), if claim 13 is found to require irregularity, and with references that teach systematic codes (Frey99, MacKay, Ping and the '999 patent).

431. Claim 16 adds to claim 15 "wherein the parity bits follow the information bits in the codeword." That is, claim 16 adds to claim 15 that in the systematic code the bits must appear in a particular order, with the parity bits following the information bits. Whether the parity bits precede or follow the systematic bits, or appear in some other order, is not significant and the limitation added by claim 16 is obvious in view of any teaching of a systematic code, and as stated above systematic codes were notoriously well known before Caltech's alleged invention.[59]

432. Further, the specification of the '781 patent offers no guidance regarding what it means for the parity bits to "follow" the information bits in the codeword, and Caltech has argued specifically that a sequence of bits can be a "codeword" even if that sequence is never transmitted (*see* Dkt. No. 67 at 17-19), so this claim limitation cannot be interpreted to require that the parity bits be transmitted at a later time than the information bits, or vice versa.

433. I conclude that "the parity bits follow the information bits in the codeword" requires only that the parity bits and the information bits not be intermingled within the codeword. A codeword therefore satisfies the requirements of claim 16 if the parity bits are located at one end of the codeword, with the information bits located at the other.

434. This requirement taught by Ping. As explained above, Ping teaches that "the codeword **c** as **c** = [**p**, **d**], where **p** and **d** contain the parity and information bits, respectively" (Ping at 38). This defines the codeword **c** as a vector, with the parity bits **p** at one end, and the information bits **d** at the other. Therefore, Ping teaches the additional limitation imposed by claim 16.

[59] The testimony of Dr. Dariush Divsalar (the author of the Divsalar reference) confirms my own opinion that the order of systematic and parity bits within a codeword is not significant (*see* Divsalar Dep. at 71:15-73:11).

435. Alternatively, to the extent that claim 16 requires that a codeword be transmitted such that the information bits are transmitted *earlier in time* than the parity bits, this limitation would have been obvious to a person of ordinary skill in the art based on the teachings of Ping alone. Ping does not specify any temporal relationship between the transmission of the parity bits **p** and the information bits **d**, but the teachings of Ping encompass schemes in which the parity bits are transmitted at a later time than the information bits.

436. The '999 patent also teaches this limitation, under either interpretation considered above. That is, it teaches that the parity bits are at one end of the codeword with the information bits at the other, and it also (and more specifically) teaches methods in which the information bits in a codeword are transmitted *earlier in time* than the parity bits:

> Suitable codewords for such schemes have been generated in a variety of ways. For systematic encoding of cyclic codes, one such method utilizes serial data input/output wherein each information word is applied to an (n-k)-stage shift register with feedback connections based on a generator polynomial. ***After** the information bits are shifted into the register and simultaneously into the communication channel, the n-k parity bits formed in the register are shifted into the channel*, thus forming the complete codeword.
>
> ('999 Patent at 1:25-34) (emphasis added).

As the above passage explains, the '999 patent teaches encoding schemes in which the information bits are shifted onto the communication channel (*i.e.*, transmitted) *earlier in time* than the associated "n-k parity bits," which are transmitted later.

437. One of ordinary skill in the art would have been motivated to combine the teachings of Divsalar and/or Frey99, Luby, or MacKay with those of either Ping or the '999 patent, for the reasons outlined above with reference to claim 15 of the '781 patent. Therefore, Claim 16 is rendered obvious by Divsalar alone or in combination with Ping or the '999 patent, and is also rendered obvious by a

combination of Divsalar, with Frey99, Luby, or MacKay, alone or in further combination with either Ping or the '999 patent.

438. Claim 16 of the '781 patent is also rendered obvious by a combination of Ping and MacKay. As I explain above, Ping and MacKay teach every element of claim 15, and Ping and MacKay also teach the additional limitation imposed by claim 16. Therefore, claim 16 is obvious over a combination of Ping and MacKay.

E. Claim 19 of the '781 Patent is Invalid

439. Claim 19 of the '781 patent reads:

> 19. A method of encoding a signal, comprising:
>
> receiving a block of data in the signal to be encoded, the block of data including information bits; and
>
> performing an encoding operation using the information bits as an input, the encoding operation including an accumulation of mod-2 or exclusive-OR sums of bits in subsets of the information bits, the encoding operation generating at least a portion of a codeword, wherein at least two of the information bits appear in three subsets of the information bits.

i) Claim 19 of the '781 Patent is Anticipated by Divsalar

440. I explain below, one limitation at a time, why claim 19 is anticipated by Divsalar.

a) "A method of encoding a signal, comprising ..."

441. Divsalar teaches the preamble, as I explain above with reference to claim 13 of the '781 patent.

b) "receiving a block of data in the signal to be encoded, the block of data including information bits"

442. Divsalar teaches this limitation, as I explain above with reference to claim 13 of the '781 patent.

c) *"performing an encoding operation using the information bits as an input, the encoding operation including an accumulation of mod-2 or exclusive-OR sums of bits in subsets of the information bits, the encoding operation generating at least a portion of a codeword"*

443. Divsalar teaches this limitation, as I explain above with reference to claim 13 of the '781 patent.

d) *"wherein at least two of the information bits appear in three subsets of the information bits"*

444. As explained above, the encoding operation of Divsalar includes an accumulation of mod-2 sums of bits in subsets of information bits. One example of such an accumulation, for $N = 5$ and $q = 3$, is given by the equations:

$y_1 = i_4$
$y_2 = i_4 + i_5$
$y_3 = i_4 + i_5 + i_1$
$y_4 = i_4 + i_5 + i_1 + i_3$
$y_5 = i_4 + i_5 + i_3$
$y_6 = i_4 + i_5 + i_3 + i_2$
$y_7 = i_5 + i_3 + i_2$
$y_8 = i_5 + i_3$
$y_9 = i_5 + i_3 + i_1$
$y_{10} = i_5 + i_1$
$y_{11} = i_5 + i_1 + i_2$
$y_{12} = i_1 + i_2$
$y_{13} = i_1 + i_2 + i_5$
$y_{14} = i_1 + i_2 + i_5 + i_3$
$y_{15} = i_1 + i_2 + i_5 + i_3 + i_4$

Here, each bit y of the codeword is the sum of a different subset of information bits (denoted using the letter i).

445. As these equations show, at least two of the information bits appear in at least three subsets of information bits. For example, the information bit i_4 appears

in seven of the equations above, and information bit i_1 appears in nine of the equations.

446. I interpret this limitation to be met as long as two information bits appear in three or more subsets, because information bits that appear in more than three subsets necessarily "appear in three subsets." For example, if an information bit appears in seven subsets (like, *e.g.*, i_4) and another information bit appears in nine subsets (like, *e.g.*, i_1) then both information bits appear in "three subsets" of information bits and this limitation is met.

447. While the equations above result from an example in which $N = 5$ and $q = 3$, other values of N and q will necessarily produce codes in which at least two of the information bits appear in three subsets of the information bits. For example an RA code with $N = 1024$, $q = 3$, which Divsalar specifically discloses, will produce a code in which at least two of the information bits appear in three subsets of the information bits (*see* Divsalar, Figure 5).

448. Also, while the equations above result from using an interleaver that scrambles bits according to one particular permutation, at least two information bits will necessarily appear in three subsets of the information bits no matter how the bits are permuted by the interleaver. Thus, Divsalar teaches that "at least two of the information bits appear in three subsets of the information bits," as required by claim 19. At a minimum, Divsalar renders claim 19 obvious because it would have been easy for one of ordinary skill to construct RA codes according to Divsalar in which "at least two of the information bits appear in three subsets of the information bits" as shown by the example above.

449. This limitation also holds for the "punctured" version of Divsalar discussed above. For example, it holds for the example given above, where the parity bits are represented by the equations:

$y_1 = i_4$
$y_7 = y_1 + (i_4 + i_5 + i_3 + i_2)$
$y_9 = y_7 + (i_2 + i_1)$
$y_{15} = y_9 + (i_4 + i_2)$

450. Here, the information bits appear in a variable number of subsets. The information bit i_4, for example, appears in three subsets, while the information bit i_5 appears in only one.

451.

e) *Summary*

452. As explained above, Divsalar teaches every limitation of, and therefore anticipates, claim 19.

ii) Claim 19 of the '781 Patent is Anticipated by Ping

453. I explain below, one limitation at a time, why claim 19 is anticipated by Ping.

a) *"A method of encoding a signal, comprising ..."*

454. Ping teaches this limitation, as I explain above with reference to claim 13 of the '781 patent.

b) *"receiving a block of data in the signal to be encoded, the block of data including information bits"*

455. Ping teaches this limitation, as I explain above with reference to claim 13 of the '781 patent.

c) *"performing an encoding operation using the information bits as an input, the encoding operation including an accumulation of mod-2 or exclusive-OR sums of bits in subsets of the information bits, the encoding operation generating at least a portion of a codeword"*

456. Ping teaches this limitation, as I explain above with reference to claim 13 of the '781 patent.

d) *"wherein at least two of the information bits appear in three subsets of the information bits"*

457. Ping teaches this limitation. As I explain above, there are $kt/(n-k)$ 1s in each row of the parity check matrix, and t 1s in each column (*see* Ping at 38). Because there are t ones in every column, each subset of information bits that is summed prior to accumulation contains exactly t bits.

458. Ping specifically teaches coding schemes "using t=4," in which each information bit appears in four distinct subsets of the information bits (Ping at 39). As I explain above, if two information bits appear in four subsets of the information bits, both information bits necessarily appear in three subsets of the information bits, and therefore Ping meets this limitation.

e) *Summary*

459. As explained above, Ping teaches every limitation of claim 19 and therefore anticipates claim 19.

IX. THE ASSERTED CLAIMS OF THE '833 PATENT ARE INVALID

460. As I explain below, asserted claims 1, 2, 4, and 8 of the '833 patent are invalid. A summary of the arguments set forth in this section is given in the table below:

'833 Claim	Divsalar + (Frey99 (or Frey slides), Luby or MacKay)	Ping + MacKay	Divsalar + (Frey99 (or Frey slides), Luby or MacKay) + '999 Patent	Ping + MacKay + '999 Patent
1	Obvious	Obvious	Obvious	Obvious
2	Obvious	Obvious	Obvious	Obvious
4	Obvious	Obvious	Obvious	Obvious
8	Obvious	Obvious	Obvious	Obvious

A. Claim 1 of the '833 Patent is Invalid

461. Claim 1 of the '833 patent reads:

1. An apparatus for performing encoding operations, the apparatus comprising:

a first set of memory locations to store information bits;

a second set of memory locations to store parity bits;

a permutation module to read a bit from the first set of memory locations and combine the read bit to a bit in the second set of memory locations based on a corresponding index of the first set of memory locations and a corresponding index of the second set of memory locations; and

an accumulator to perform accumulation operations on the bits stored in the second set of memory locations,

wherein two or more memory locations of the first set of memory locations are read by the permutation module different times from one another.

i) Claim 1 of the '833 Patent is obvious over Divsalar in view of Frey99, Luby or MacKay

462. I explain below, one limitation at a time, why claim 1 is obvious over Divsalar in view of any one of Frey99, Luby or MacKay (*i.e.*, Divsalar + (Frey99, Luby or MacKay)).

a) "an apparatus for performing encoding operations"

463. Divsalar teaches the preamble. As I explain above, Divsalar describes a "turbo-like" code called a repeat-accumulate code. A "coder" capable of encoding information bits using a repeat-accumulate code is shown in Figure 3 of Divsalar, reproduced above.

b) "a first set of memory locations to store information bits"

464. Divsalar teaches this limitation. A person of ordinary skill in the art would recognize that the input block comprising information bits would be stored in a set of memory locations (*i.e.*, the block of *N* bits input to the repeater as shown in Figure 3 would have been stored in memory locations).

465. It would have been understood by one of ordinary skill in the art that the encoder of Divsalar may be implemented on a general-purpose computer, where

the information bits would be stored in a buffer comprising a set of memory locations. Indeed, one of ordinary skill in the art would have understood that Divsalar *himself* implemented an RA encoder using a computer program. Divsalar states "[w]e wrote a ***computer program*** to implement this "turbo-like" decoding for RA codes with $q = 3$ (rate 1/3) and $q = 4$ (rate 1/4), and the results are shown in Figure 5" (Divsalar at 9) (emphasis added). Divsalar ran this decoding program, using sample encoded data as input, and measured the resulting coding performance, which is plotted in Figure 5. One of ordinary skill in the art would have understood Divsalar to have implemented an *encoder* program, in order to generate the sample encoded data provided to the decoder.

466. Even if the RA codes of Divsalar were implemented using special-purpose hardware components, an obvious implementation would have been to store the information bits in a first set of memory locations within a memory buffer.

467. Divsalar itself is silent regarding the first set of memory locations, but so is the specification of the '833 patent. If one of ordinary skill would have understood the claimed first set of memory locations and their use from the '833 specification, then they would have understood it from Divsalar too.[60]

c) *<u>"a second set of memory locations to store parity bits"</u>*

468. Divsalar teaches this limitation. A person of ordinary skill in the art would recognize that the "output block $[y_1, \ldots, y_n]$" comprising parity bits would be stored in a set of memory locations. *See* Divisalar at 5.

469. It would have been understood by one of ordinary skill in the art that the encoder of Divsalar may be implemented on a general-purpose computer, where the parity bits would be stored in a buffer comprising a set of memory locations.

[60] See Wicker Tr. at 109-11.

As explained above, Divsalar himself implemented an encoder program, in order to generate the sample encoded data provided to the decoder.

470. Even if the RA codes of Divsalar were implemented using special-purpose hardware components, an obvious implementation would have been to store the parity bits in a second set of memory locations within a memory buffer.

471. Divsalar itself is silent regarding the second set of memory locations, but so is the specification of the '833 patent. If one of ordinary skill would have understood the claimed second set of memory locations and their use from the '833 specification, then they would have understood it from Divsalar too.

d) *"a permutation module to read a bit from the first set of memory locations and combine the read bit to a bit in the second set of memory locations based on a corresponding index of the first set of memory locations and a corresponding index of the second set of memory locations"*

472. As explained above, the RA encoder shown in Fig. 3 of Divsalar comprises three stages: repeat, interleave, and accumulate. In an implementation of the repeat and interleave stages that would have been obvious to one of ordinary skill in the art, the repeat stage of the encoder reads each of the N information bits stored in the first set of memory locations q times, and sequentially writes the resulting duplicated bits to a set of $N \times q$ memory locations (*i.e.*, the "second set of memory locations). Interleaving is accomplished by writing the bits into the second set of memory locations in one order and then reading them out of the second set of memory locations in a different order (*e.g.*, writing the bits into the second set of memory locations in a pseudo-random order and then reading the bits out in sequential order).

473. The act of writing a bit to one of the second set of memory locations constitutes "combining" the scrambled bit with the destination value.[61] One of ordinary skill in the art would have recognized that memory buffers are generally initialized at the start of the encoding process, setting the contents of every memory location in the buffer to zero. When the permutation module writes a bit *b* into one of the second set memory locations after it has been initialized, it has the effect of "combining" *b* with the value already stored in the memory location (*i.e.*, a 0) using an XOR operation.[62]

474. Collectively, the repeat and interleaving stages of the encoder in this implementation constitute the claimed "permutation module."[63] The repeat stage reads each information bit multiple times from the first set of memory locations, and the interleaving stage changes the order of the repeated bits by writing them into the second set of memory locations in one order and reading them out in a different order.[64]

475. Divsalar is silent regarding how to perform the interleaving using the first and second sets of memory locations, but so is the specification of the '833 patent. If one of ordinary skill would have understood, from the '833 specification, how to

[61] Here I interpret the word "combine" according to the Court's construction of that term, which is to "perform logical operations on" (Claim Construction Order dated August 6, 2014, p. 18). An XOR operation is a "logical operation" that falls within the scope of this construction (*see, e.g.*, '833 patent at claim 2, which reads "[t]he apparatus of claim 1, wherein the permutation module is configured to perform the combine operation to include performing mod-2 or exclusive-OR sum").
[62] Use of such combinatorial logic feeding the input to memories is a common and obvious technique.
[63] Here I interpret the term "permutation module" to mean "a module that changes the order of data elements," as both parties have agreed in their Joint Claim Construction Statement.
[64] Strictly speaking, reading each bit multiple times from the first set of memory locations is not required. That is, a repeat can be accomplished by reading a bit once and writing it multiple times to different memory locations. However, it would not be inventive to read the bit multiple times, *i.e.*, once for every time the bit is repeated.

perform the claimed interleaving using the memory locations, then they would have understood it from Divsalar too.

e) *<u>"an accumulator to perform accumulation operations on the bits stored in the second set of memory locations"</u>*

476. Divsalar teaches this limitation. As I explain above, the final stage of the encoder shown in Fig. 3 of Divsalar is an accumulator, which performs accumulation operations on the interleaved, repeated information bits (Divslar at 5). In a software implementation of the RA encoder (such as the "computer program" that Divsalar himself used to measure the performance of RA codes) one of ordinary skill in the art would have recognized that an obvious way to implement the accumulation stage of the encoder would be to accumulate the scrambled bits *in place*. That is, the "output block [y_1, …, y_n]" would overwrite the "input block [x_1, …, x_n]" (Divsalar at 5). Because in-place accumulation uses the same set of memory locations for both its input and output, it has the benefit of not requiring any additional memory.

477. The two tables below illustrate how the accumulation is performed *in place* in the second set of memory locations. Initially, the bit x_1 is stored in the first location, x_2 is stored in the second location, and so on. For the accumulation operation, the contents of the first location need not change. But, the second location is replaced with the sum of the current value of the second location and the prior location, *i.e.*, $x_1 + x_2$. That operation effectively removed the value x_2 as a standalone quantity from the memory. But, that quantity is not needed for any of the future operations. Rather than x_2, the sum $x_1 + x_2$ is what is needed for the next operation, and that is exactly what is now stored in the second location. The next step is to add that quantity, $x_1 + x_2$, to the current value at the third location and that sum, $x_1 + x_2 + x_3$, is then stored at the third location. That process then continues

for the remainder of the bits. Because each value that is overwritten is no longer needed for future computations, the accumulation can be performed in place.

Index	1	2	3	4	…
Stored Value	x_1	x_2	x_3	x_4	…

2nd set of memory locations before accumulation

Index	1	2	3	4	…
Stored Value	x_1	x_1+x_2	$x_1+x_2+x_3$	$x_1+x_2+x_3+x_4$	…

2nd set of memory locations after accumulation

478. At the end of the accumulation process, the "output block [$y_1, \ldots, y_n$]" (*i.e.*, the parity bits) would be stored in the second set of memory locations (*see* Divsalar at 5).

479. Divsalar is silent as to how the accumulation would be performed within the second set of memory locations, but so is the specification of the '833 patent. If one of ordinary skill would have understood how to perform the claimed accumulation within the second set of memory locations from the '833 specification, then they would have understood it from Divsalar too.

f) *"wherein two or more memory locations of the first set of memory locations are read by the permutation module different times from one another"*

480. The repeat stage of Divsalar is regular, and so the permutation module reads every memory location in the first set of memory locations the same number of times. However, Frey99 teaches irregular repetition, and Luby and MacKay both teach the benefits of irregular coding, as explained above. Incorporating the irregular repetition of Frey99, or the irregularity of Luby or MacKay, into the implementation of the Divsalar encoder described above would result in an

irregular repeater which reads some of the first set of N memory locations more times than it reads others, as required by this limitation.[65]

481. Divsalar does not explicitly explain that the repeat is accomplished by reading the same bit out of memory more than once. However, neither does the specification of the '833 patent. If one of ordinary skill would have understood, from the '833 specification, how to perform repeating by reading bits more than once from memory, then they would have understood it from Divsalar too.

g) *Summary*

482. As explained above, the combination of Divsalar and Frey99, Luby or MacKay teaches every limitation of claim 1 of the '833 patent.

h) *Motivations to combine the irregular repeater of Frey99, or the irregularity of Luby or MacKay, with the RA codes of Divsalar*

483. As I explain above, one of ordinary skill in the art would have been motivated to combine Divsalar and Frey99, Luby or MacKay in general, and would specifically have been motivated to use the irregular repetition of Frey99, or the irregularity of Luby or MacKay, with the RA codes of Divsalar. Briefly, Frey99, Luby and MacKay all taught the benefits of irregular coding and one of ordinary skill would have understood that Divsalar's RA could would have benefited from making it irregular. I also explain above why such a combination would require only a minor modification to the teachings of Divsalar, and would not fundamentally change its principle of operation.

ii) Claim 1 of the '833 Patent is obvious over Ping in view of MacKay

484. I explain below, one limitation at a time, why claim 1 is obvious over Ping in view of MacKay.

[65] Here I interpret "different times from one another" to mean "a different number of times from one another," as both parties have agreed in their Joint Claim Construction Statement.

a) <u>"an apparatus for performing encoding operations"</u>

485. Ping teaches the preamble. As I explain above, Ping teaches constructing LDPC codes that can be encoded in two stages. In the first encoding stage, a generator matrix is applied to a sequence of *k* information bits to produce sums of information bits. In the second stage, the sums of information bits are accumulated recursively to generate *n-k* parity bits (*see* Ping at 38). One of ordinary skill in the art would have understood that the encoding process taught by Ping would be implemented by "an apparatus for performing encoding operations."

b) <u>"a first set of memory locations to store information bits"</u>

486. Ping teaches this limitation. A person of ordinary skill in the art would recognize that an implementation of Ping would store information bits in a set of memory locations. Specifically, it would have been obvious to one of ordinary skill in the art to implement the encoding processes disclosed by Ping using hardware (*e.g.*, a general-purpose computer or special-purpose electronic components) that comprises a first set of memory locations for storing information bits.

487. Like Divsalar, Ping is silent regarding the first set of memory locations, but so is the specification of the '833 patent. If one of ordinary skill would have understood the claimed first set of memory locations and their use from the '833 specification, then they would have understood it from Ping too.

c) <u>"a second set of memory locations to store parity bits"</u>

488. Ping teaches this limitation. A person of ordinary skill in the art would recognize that an implementation of Ping would store parity bits in a set of memory locations. Specifically, it would have been obvious to one of ordinary skill in the art to implement the encoding processes disclosed by Ping using hardware (*e.g.*, a general-purpose computer or special-purpose electronic

components) that comprises a second set of memory locations for storing parity bits.

489. Like Divsalar, Ping is silent regarding the second set of memory locations, but so is the specification of the '833 patent. If one of ordinary skill would have understood the claimed second set of memory locations and their use from the '833 specification, then they would have understood it from Ping too.

> *d)* *<u>"a permutation module to read a bit from the first set of memory locations and combine the read bit to a bit in the second set of memory locations based on a corresponding index of the first set of memory locations and a corresponding index of the second set of memory locations"</u>*

490. Ping teaches this limitation under Caltech's apparent interpretation of "permutation module." As explained above, Ping teaches constructing LDPC codes that can be encoded in two stages. In the first encoding stage, sums of subsets of the information bits are computed by reading each of the *k* information bits from the first set of memory locations and, and combining, using an XOR operation, the read bit with the bit stored in one of a second set of *n-k* memory locations. Eventually, each of the second set of memory locations will contain the sum of a subset of the information bits, which Ping denotes:

$$\sum_{j} h^{d}_{ij} d_j$$

(Ping at 38).

491. The first encoding stage of Ping does not "change the order of data elements," as required by the Court's construction of "permutation module." Rather, in Ping, the second set of memory locations stores sums of bits rather than reordered versions of the bits themselves. Plaintiff's infringement arguments, however, still

appear to be based on an interpretation of "permutation module" that does not require changing the order of bits themselves. Under Caltech's interpretation, the first encoding stage of Ping would constitute a "permutation module," as required by this limitation.[66]

492. Like Divsalar, Ping is silent regarding how to perform the interleaving using the first and second sets of memory locations. But so is the specification of the '833 patent. If one of ordinary skill would have understood, from the '833 specification, how to perform the claimed interleaving using the memory locations, then they would have understood it from Ping too.

e) *<u>"an accumulator to perform accumulation operations on the bits stored in the second set of memory locations"</u>*

493. Ping teaches this limitation. As I explain above, the final stage of the encoding process disclosed in Ping is an accumulation, in which the parity bits $\mathbf{p} = \{p_i\}$ are computed by accumulating the sums calculated during the first stage, defined recursively as follows:

$$p_1 = \sum_j h^d_{1j} d_j$$

$$p_i = p_{i-1} + \sum_j h^d_{ij} d_j$$

(Ping at 38)

494. In a software implementation of the encoding processes taught by Ping, one of ordinary skill in the art would have recognized that an obvious way to implement the accumulation stage would be to accumulate the scrambled bits *in*

[66] That is, in the LDPC codes of DVB-S2, the parity bits are all sums of two or more information bits. But there is no need to reorder the information bits to construct such parity bits.

place, as was explained above with respect to Divsalar. Because in-place accumulation uses the same set of memory locations for both its input and output, it has the benefit of not requiring any additional memory.

495. At the end of the accumulation process, the parity bits $\mathbf{p} = \{p_i\}$ would be stored in the second set of n-k memory locations.

496. Like Divsalar, Ping is silent as to how the accumulation would be performed within the second set of memory locations, but so is the specification of the '833 patent. If one of ordinary skill would have understood how to perform the claimed accumulation within the second set of memory locations from the '833 specification, then they would have understood it from Ping too.

f) *"wherein two or more memory locations of the first set of memory locations are read by the permutation module different times from one another"*

497. In the LDPC codes disclosed by Ping, the parity-check matrix **H** has a column weight of t, so the permutation module would read each information bit t times, combining the information bit into t different locations in the second set of memory locations (*see* Ping at 38).

498. However, as explained above, MacKay teaches parity check matrices with nonuniform column weights. Implementing the LDPC-accumulate coders disclosed by Ping using the irregular parity check matrices disclosed by MacKay would result in a permutation module that reads some of the first set of memory locations more times than it reads others, as required by this limitation.[67]

499. Like Divsalar, Ping and MacKay do not explicitly explain reading the same bit out of memory more than once. However, neither does the specification of

[67] As noted above with respect to Divsalar, strictly speaking the bits need not be read multiple times to repeat them, i.e., each bit could be read once and then written multiple times. However, implementing the repeat by reading each bit multiple times would have been obvious.

the '833 patent. If one of ordinary skill would have understood, from the '833 specification, reading bits more than once from memory, then they would have understood it from both Ping and MacKay too.

g) <u>Summary</u>

500. As explained above, the combination of Ping and MacKay teaches every limitation of claim 1 of the '833 patent.

h) <u>Motivations to combine the irregularity of MacKay with the LDPC-accumulate coders of Ping</u>

501. As I explain above, one of ordinary skill in the art would have been motivated to combine Ping and MacKay in general, and would specifically have been motivated to combine the LDPC-accumulate coders disclosed by Ping with the irregular parity check matrices disclosed by MacKay. I also explain why such a combination would require only a minor modification to the teachings of Ping, and would not fundamentally change its principle of operation.

502. For at least the reasons given above, claim 1 of the '833 patent is obvious over the combination of Ping and MacKay.

iii) <u>Claim 1 of the '833 Patent is obvious over Divsalar in view of one of Frey99, Luby or MacKay and further in view of the '999 Patent</u>

503. As explained above, Divsalar and Frey99, Luby or MacKay together teach every limitation of, and render obvious, claim 1 of the '833 patent. However, in the event Divsalar and Frey99, Luby or MacKay are found not to teach the "memory locations" recited in claim 1, then claim 1 is obvious over the combination of Divsalar, Frey99, Luby or MacKay and the '999 patent (*i.e.*, Divsalar + (Frey99, Luby or MacKay) +'999 patent).

504. At a high level, use of memories for implementing codes was notoriously well known in the art long before Caltech's alleged invention. One of ordinary

skill reading Divsalar, Frey99, Luby or MacKay would have understood that their codes are implemented using memories as described above. The '999 patent is merely an example of a reference showing generally how use of memories was known.

505. As I explain above, the '999 patent teaches an encoder for encoding information bits using a linear error-correcting code. The encoder taught by the '999 patent uses a plurality of memories that store values used during the encoding process ('999 patent at Abstract). While the memories taught by the '999 patent are read-only memories, one of ordinary skill in the art would have recognized that writable memories may also be used to implement the encoding process (*see* '999 patent at Abstract). I also explain above why one of ordinary skill in the art would have been motivated to combine Divsalar, Frey99, Luby or MacKay and the '999 patent.

506. Therefore, for at least the reasons given above, claim 1 is obvious over the combination of Divsalar, Frey99, Luby, or MacKay and the '999 patent.

iv) <u>Claim 1 of the '833 Patent is obvious over Ping in view of MacKay and the '999 Patent</u>

507. As explained above, Ping and MacKay together teach every limitation of, and render obvious, claim 1 of the '833 patent (under Caltech's apparent interpretation of "permutation module"). However, in the event Ping and MacKay are found not to teach the "memory locations" recited in claim 1, then claim 1 is obvious over the combination of Ping, MacKay, and the '999 patent.

508. As noted above, at a high level, use of memories for implementing codes was notoriously well known in the art long before Caltech's alleged invention. One of ordinary skill reading Ping or MacKay would have understood that their codes are implemented using memories as described above. The '999 patent is

merely an example of a reference showing generally how use of memories was known.

509. As I explain above, the '999 patent teaches an encoder for encoding information bits using a linear error-correcting code using memories that store values used during the encoding process ('999 patent at Abstract).

510. Further, one of ordinary skill in the art would have been motivated to combine Ping and MacKay with the '999 patent. Like Ping and MacKay, the '999 patent relates to methods of improving the performance of linear error-correcting codes. It was filed in 1984 and granted in 1986, well over a decade before the claimed priority date of the patents-in-suit. By March 7, 2000, the alleged conception date of the patents-in-suit, the technology described in the '999 patent would have been well known in the field, and one of ordinary skill in the art at the time would have been motivated to combine the teachings of Ping and MacKay with those of the '999 patent.

511. Therefore, for at least the reasons given above, claim 1 is obvious over the combination of Ping, MacKay, and the '999 patent.

B. Claim 2 of the '833 Patent is Invalid

512. Claim 2 of the '833 patent reads:

> 2. The apparatus of claim 1, wherein the permutation module is configured to perform the combine operation to include performing mod-2 or exclusive-OR sum.

513. Claim 2 is rendered obvious by a combination of Divsalar and Frey99, Luby or MacKay, and by a combination of Ping and MacKay. Claim 2 is also rendered obvious by each of these combinations considered in view of the '999 patent.

514. As I explained above, claim 1 is rendered obvious by each of these combinations. Claim 2 adds to claim 1 "wherein the permutation module is configured to perform the combine operation to include performing mod-2 or

exclusive-OR sum." This additional limitation of claim 2 is taught by each of Divsalar and Ping, as explained above.

515. Therefore, claim 2 is rendered obvious by a combination of Divsalar and Frey99, and by a combination of Ping and MacKay, and is also rendered obvious by each of these combinations in view of the '999 patent.

C. Claim 4 of the '833 Patent is Invalid

516. Claim 4 of the '833 patent reads:

> 4. The apparatus of claim 1, wherein the accumulator is configured to perform the accumulation operation to include a mod-2 or exclusive-OR sum of the bit stored in a prior index to a bit stored in a current index based on a corresponding index of the second set of memory locations.

517. Claim 4 is rendered obvious by a combination of Divsalar and Frey99, Luby or MacKay, and by a combination of Ping and MacKay. Claim 4 is also rendered obvious by each of these combinations, considered in view of the '999 patent.

518. As I explain above, claim 1 is rendered obvious by each of these combinations. Claim 4 adds to claim 1 "wherein the accumulator is configured to perform the accumulation operation to include a mod-2 or exclusive-OR sum of the bit stored in a prior index to a bit stored in a current index based on a corresponding index of the second set of memory locations."

519. This additional limitation of claim 4 is taught by each of Divsalar and Ping. For example, the accumulator of Divsalar calculates the parity bits y1, …, y_n as follows:

> [W]e prefer to think of [the accumulator] as a block coder whose input block [$x_1, \ldots, x_n$] and output block [$y_1, \ldots, y_n$] are related by the formula
>
> $$y_1 = x_1$$
> $$y_2 = x_1 + x_2$$
> $$y_3 = x_1 + x_2 + x_3$$
> $$y_n = x_1 + x_2 + x_3 + \ldots + x_n.$$

(Divsalar at 5)

520. From the above passage, one can see that all of the parity bits y_i (except the first parity bit y_1) can be defined by the recursive formula $y_i = y_{i-1} + x_i$. Thus, calculating y_i involves taking the mod-2 sum of the bit stored in a prior index (*i.e.*, parity bit y_{i-1}) and a bit stored in a current index (*i.e.*, repeated information bit x_i).

521. Similarly, Ping defines each parity bit p_i (except the first parity bit p_1) recursively as follows:

$$p_i = p_{i-1} + \sum_j h^d_{ij} d_j$$

(Ping at 38)

522. Thus, calculating p_i involves taking the mod-2 sum of the bit stored in a prior index (*i.e.*, parity bit p_{i-1}) and a bit stored in a current index (*i.e.*, the sum of the i^{th} subset of information bits $\sum_j h^d_{ij} d_j$).

523. Therefore, claim 4 is rendered obvious by a combination of Divsalar and Frey99, and by a combination of Ping and MacKay, and is also rendered obvious by each of these combinations in view of the '999 patent.

D. Claim 8 of the '833 Patent is Invalid

524. Claim 8 of the '833 patent reads:

> 8. A method of performing encoding operations, the method comprising:
>
> receiving a sequence of information bits from a first set of memory locations;
>
> performing an encoding operation using the received sequence of information bits as an input, said encoding operation comprising:
>
> reading a bit from the received sequence of information bits, and combining the read bit to a bit in a second set of memory locations based on a corresponding index of the first set of memory locations

for the received sequence of information bits and a corresponding index of the second set of memory locations; and

accumulating the bits in the second set of memory locations, wherein two or more memory locations of the first set of memory locations are read by the permutation module different times from one another.

i) Claim 8 of the '833 Patent is obvious over Divsalar in view of Frey99, Luby or MacKay

525. I explain below, one limitation at a time, why claim 8 is obvious over Divsalar in view of Frey99, Luby or MacKay.

a) "a method of performing encoding operations"

526. Divsalar teaches the preamble, as I explain above with reference to claim 1 of the '833 patent.

b) "receiving a sequence of information bits from a first set of memory locations"

527. Divsalar teaches this limitation, as I explain above with reference to claim 1 of the '833 patent.

c) "performing an encoding operation using the received sequence of information bits as an input"

528. Divsalar teaches this limitation, as I explain above with reference to claim 1 of the '833 patent.

d) "said encoding operation comprising: reading a bit from the received sequence of information bits, and combining the read bit to a bit in a second set of memory locations based on a corresponding index of the first set of memory locations for the received sequence of information bits and a corresponding index of the second set of memory locations"

529. Divsalar teaches this limitation, as I explain above with reference to claim 1 of the '833 patent.

e) *"accumulating the bits in the second set of memory locations, wherein two or more memory locations of the first set of memory locations are read by the permutation module different times from one another"*

530. Frey99, Luby and MacKay each teach this limitation, as I explain above with reference to claim 1 of the '833 patent.

f) *Summary*

531. As explained above, the combination of Divsalar and Frey99, Luby or MacKay teaches every limitation of claim 8 of the '833 patent.

g) *Motivations to combine the irregular repeater of Frey99, Luby or MacKay with the RA codes of Divsalar*

532. As I explain above, one of ordinary skill in the art would have been motivated to combine Divsalar and Frey99, Luby or MacKay in general, and would specifically have been motivated to use the irregular repetition of Frey99, or the irregularity of Luby or MacKay, with the RA codes of Divsalar. I also explain why such a combination would require only a minor modification to the teachings of Divsalar, and would not fundamentally change its principle of operation.

533. For at least the reasons given above, claim 8 of the '833 patent is obvious over Divsalar in view of Frey99, Luby or MacKay.

ii) Claim 8 of the '833 Patent is obvious over Ping in view of MacKay

534. I explain below, one limitation at a time, why claim 8 is obvious over Ping in view of MacKay.

a) *"a method of performing encoding operations"*

535. Ping teaches the preamble, as I explain above with reference to claim 1 of the '833 patent.

b) <u>*"receiving a sequence of information bits from a first set of memory locations"*</u>

536. Ping teaches this limitation, as I explain above with reference to claim 1 of the '833 patent.

c) <u>*"performing an encoding operation using the received sequence of information bits as an input"*</u>

537. Ping teaches this limitation, as I explain above with reference to claim 1 of the '833 patent.

d) <u>*"said encoding operation comprising: reading a bit from the received sequence of information bits, and combining the read bit to a bit in a second set of memory locations based on a corresponding index of the first set of memory locations for the received sequence of information bits and a corresponding index of the second set of memory locations"*</u>

538. Ping teaches this limitation, as I explain above with reference to claim 1 of the '833 patent.

e) <u>*"accumulating the bits in the second set of memory locations, wherein two or more memory locations of the first set of memory locations are read by the permutation module different times from one another"*</u>

539. MacKay teaches this limitation, as I explain above with reference to claim 1 of the '833 patent.

f) <u>*Summary*</u>

540. As explained above, the combination of Ping and MacKay teaches every limitation of claim 8 of the '833 patent.

g) <u>*Motivations to combine the irregularity of MacKay with the LDPC-accumulate coders of Ping*</u>

541. As I explain above, one of ordinary skill in the art would have been motivated to combine Ping and MacKay in general, and would specifically have

been motivated to combine the LDPC-accumulate coders disclosed by Ping with the irregular parity check matrices disclosed by MacKay. I also explain why such a combination would require only a minor modification to the teachings of Ping, and would not fundamentally change its principle of operation.

542. For at least the reasons given above, claim 8 of the '833 patent is obvious over Ping in view of MacKay.

iii) <u>Claim 8 of the '833 Patent is obvious over Divsalar in view of one of Frey99, Luby or MacKay and further in view of the '999 Patent</u>

543. As explained above, Divsalar and Frey99, Luby or MacKay together teach every limitation of, and render obvious, claim 8 of the '833 patent. However, in the event Divsalar and Frey, Luby or MacKay are found not to teach the "memory locations" recited in claim 8, then claim 8 is obvious over the combination of Divsalar, Frey99, Luby or MacKay and the '999 patent.

544. As I explain above, the '999 patent teaches an encoder for encoding information bits using a linear error-correcting code. The encoder taught by the '999 patent uses a plurality of memories that store values used during the encoding process ('999 patent at Abstract).

545. Further, I also explain above why one of ordinary skill in the art would have been motivated to combine Divsalar, Frey99, Luby or MacKay and the '999 patent. Therefore, for at least the reasons given above, claim 8 is obvious over the combination of Divsalar, Frey99, Luby or MacKay and the '999 patent.

iv) <u>Claim 8 of the '833 Patent is obvious over Ping in view of MacKay and the '999 Patent</u>

546. As explained above, Ping and MacKay together teach every limitation of, and render obvious, claim 8 of the '833 patent. However, in the event Ping and

MacKay are found not to teach the "memory locations" recited in claim 8, then claim 1 is obvious over the combination of Ping, MacKay, and the '999 patent.

547. As I explain above, the '999 patent teaches an encoder for encoding information bits using a linear error-correcting code using memories that store values used during the encoding process ('999 patent at Abstract).

548. Further, as I explain above, one of ordinary skill in the art would have been motivated to combine Ping and MacKay with the '999 patent. Therefore, for at least the reasons given above, claim 8 is obvious over the combination of Ping, MacKay, and the '999 patent.

E. Claims 1, 2, 4, and 8 are Invalid for Lack of Written Description.

549. Independent claims 1 and 8 recite "memory locations." Specifically, these claims recite "a first set of memory locations" for storing information bits, and "a second set of memory locations" for storing parity bits. They further require reading an information bit from one of the first set of memory locations and combining the read bit with a bit in the second set of memory locations based on a "corresponding index of the first set of memory locations." Dependent claims 2 and 4 inherit these limitations from independent claim 1, from which they depend.

550. However, the first reference to "memory locations," sets of "memory locations," or indices pointing to "memory locations" in the prosecution histories of the patents-in-suit appears in the claims of the '833 patent, filed on March 28, 2011. For this reason, it is my opinion that the claims of the '833 patent are invalid because the disclosure lacks sufficient written description of the claimed invention.

551. In the alternative, in the event that one or more claims of the '833 patent are found not to be invalid under the written description requirement, the earliest priority date to which those claims could be entitled is March 28, 2011, the date those claims were first filed.

552. The claims of the '833 patent, and in particular their use of memory locations and indexes may have been obvious to one of ordinary skill in the art in view of the '833 specification. As noted above, use of memories for coding was well known before Caltech's alleged invention. However, I understand that the test for written description is not whether the invention would have been obvious but whether the words or figures of the specification show the inventors were in possession of the invention. The specification of the '833 patent does not communicate to one of ordinary skill in the art that the inventors were in possession of the alleged invention.

F. All asserted claims are invalid over Hughes' products

553. As noted above, I have been told that a number of the accused products in this case were sold by Defendants prior to March 28, 2011. I have further been informed that Caltech has not varied its infringement contentions for any of the products, i.e., it has treated all accused products the same. I have not studied the accused products. However, if Caltech succeeds in demonstrating infringement of any claims of the '833 patent, then those claims would be invalid over the accused products that were sold prior to March 28, 2011. That is, I have been informed of the axiom of patent law, "that which infringes if later, anticipates if earlier." If Caltech proves that its '833 claims cover the accused products, then those same products that preceded the '833 claims would invalidate those claims.

X. SECONDARY CONSIDERATIONS

554. I have reviewed Caltech's Second Supplemental Responses to Defendants' First Set of Interrogatories, which relate in part to secondary considerations of non-obviousness (*see* Caltech's First and Second Supplemental Responses to Interrogatory Number 5). It is my opinion that the supposed indicia of nonobviousness identified by Caltech in these responses are not relevant to the

claims of the patents-in-suit, and fail to provide support for Caltech's position that the asserted claims are not obvious.

555. Caltech contends that the claimed invention was not obvious because it enjoyed commercial success, but to support this contention Caltech's responses merely present evidence that "irregular repeat accumulate (IRA) codes" have been commercially successful. While the patents-in-suit relate generally to IRA codes, the asserted claims do not cover all possible implementations of IRA codes. A product may use "IRA codes" without using the claimed invention. Therefore, pointing to the supposed commercial success of products that use "IRA codes" does not demonstrate that the claimed invention itself was not obvious. This objection is not limited to Caltech's evidence of "commercial success," but also applies to the other supposed indicia of non-obviousness that Caltech identifies. Even if it were true (and it is not) that IRA codes have been "widely praised by the industry," have "overcome skepticism from experts," or have achieved "unexpected results," these facts would not demonstrate the non-obviousness of the particular class of codes that is covered by the asserted claims. Also, as part of its allegation of commercial success, Caltech has pointed to sales of the accused products. However, I understand that the accused products have not been shown to infringe and, without such a showing, sales of those products do not demonstrate non-obviousness of the claims. Further, Caltech has not provided any evidence that any commercial success of the accused products was based on the features of those products.

556. Further, Caltech contends that the asserted claims are not obvious because they have "overcome skepticism from experts," and that high-performance, low complexity codes had "long eluded the telecommunications industry." However, these contentions are incorrect. In fact, experts were not skeptical about the feasibility of implementing codes characterized by both good error correcting

performance and computational efficiency. Rather, prior to Caltech's alleged invention, it was well known that codes could be designed that have these properties. For example, Ping states that "[t]he new method can achieve essentially the same performance as the standard LDPC encoding method with significantly reduced complexity" (Ping at 39). Similarly, MacKay describes a class of "fast encoding" Gallager codes that allow for reduced encoding complexity while demonstrating "equally good performance" (MacKay at 1449; *see also* MacKay at 1452, "Decoding Times: Not only do these irregular codes outperform the regular codes, they require fewer iterations ..."). Indeed, months before the alleged conception date of the patents-in-suit, I myself had suggested making repeat-accumulate codes irregular (*see* CALTECH000024021).

557. Caltech has also failed to demonstrate that the claimed codes achieved unexpected results. Prior to Caltech's alleged invention, it was well understood that making a code irregular would improve its performance (*see generally*, *e.g.*, Luby, MacKay, Frey99, Frey Slides, Luby97, Luby98, Richardson). For example, Frey99 demonstrates that irregular turbocodes outperform the corresponding regular turbo codes, and the improvement in performance is consistent with what was shown previously by Luby and MacKay. Based on these results, it was expected that adding irregularity to RA codes would also improve performance, which was later found to be the case. In conclusion, the supposed indicia of non-obviousness identified by Caltech fail to show that the claimed invention was obvious.

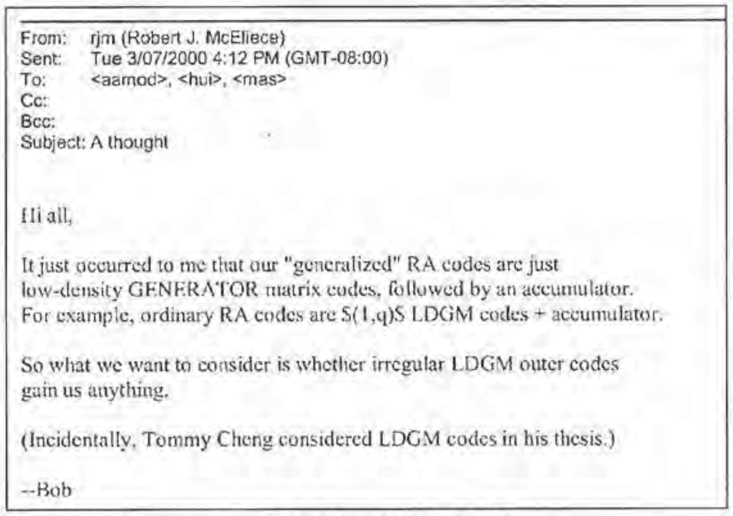
558. Also, I understand that simultaneous invention by others is evidence of obviousness. As noted above, I myself suggested to Dariush Divsalar that he make his RA codes irregular. *See* Email from Brendan Frey to Dariush Divsalar dated Dec. 8, 1999 (CALTECH000024021). If turning RA codes into IRA codes was inventive, I made that invention myself before Caltech claims to have done so.

Also, as noted above, David MacKay also produced IRA codes with his RA.c software before Caltech claims to have developed its alleged invention. My work and that of David MacKay shows simultaneous development by others, and further evidences the obviousness of Caltech's claims.

XI. THE MARCH 7 2000 MCELIECE EMAIL

559. I have been informed that Caltech argues that an email dated March 7, 2000 sent by Robert McEliece, one of the inventors of the patents-in-suit, is evidence of the conception of the inventions to which the asserted claims are directed.

560. In its entirety, the email reads as follows:


From: rjm (Robert J. McEliece)
Sent: Tue 3/07/2000 4:12 PM (GMT-08:00)
To: <aamod>, <hui>, <mas>
Cc:
Bcc:
Subject: A thought

Hi all,

It just occurred to me that our "generalized" RA codes are just low-density GENERATOR matrix codes, followed by an accumulator. For example, ordinary RA codes are S(1,q)S LDGM codes + accumulator.

So what we want to consider is whether irregular LDGM outer codes gain us anything.

(Incidentally, Tommy Cheng considered LDGM codes in his thesis.)

--Bob

CALTECH000008667

561. The emails suggests trying to incorporate irregularity into a class of known codes, but does not indicate whether the resulting irregular code would result in any improvement over the state of the art.The first paragraph describes a set of "generalized" RA codes that are "just low-density generator matrix codes, followed by an accumulator." These codes were known in the art, and described in, *e.g.*, the Divsalar reference, as I explain above.

562. The sole mention of irregularity in the email appears in the second paragraph, which contains only the sentence "[s]o what we want to consider is whether

irregular LDGM outer codes gain us anything." This sentence characterizes incorporating irregularity into LDGM-accumulate codes as an idea for further consideration, and not as a fully conceived invention.

563. CALTECH000008667 does not explain how to design or implement the "irregular LDGM outer codes" that are referenced in the second paragraph. In particular, irregular codes depend crucially on a feature known as a "degree profile" (as described in claim 6 of the '710 patent), but CALTECH000008667 nowhere mentions which degree profile to use or how the degree profile should be selected. In fact, CALTECH000008667 does not explicitly state that irregular *repetition* should be used for the LDGM outer code, which is required by many of the asserted claims of the patents-in-suit, as I explain above. Given the lack of a concrete suggestion as to *how* to incorporate irregularity into LDGM codes, the email above does not show that the inventors, at the time of the email, had made the invention claimed in the patents.

564. In my opinion, CALTECH000008667 at most expresses McEliece's *hope* that adding irregularity to LDGM-accumulate codes would produce desirable results (*see id.*, "so what we want to ***consider*** is ***whether*** irregular LDGM outer codes gain us anything") (emphasis added).

565. Finally, CALTECH000008667 only suggests adding irregularity to LDGM codes, but the claimed invention purports to be applicable to a broader class of codes than LDGM codes. For example, in the '710 claims, only dependent claims 7, 13, and 20 recite a first-encoding step involving a "low-density generator matrix." This implies that independent claims 1, 11, and 15 (from which claims 7, 13, and 20 depend, respectively) are intended to cover a broader class of codes than irregular LDGM-accumulate codes. Also, in the '710 claims, only dependent claims 4, 5 and 7 recite a second-encoding step involving an "accumulator." This implies that independent claims 1, 2 and 3 (from which claims 4, 5 and 7 depend)

are intended to cover a broader class of codes than irregular LDGM-accumulate codes. However, the email reproduced above mentions irregularity only in the context of LDGM-accumulate codes, and does not suggest incorporating irregularity into a broader class of codes.

566. Further, the attached email is silent about limitations in the claims of the asserted patents (e.g., "obtaining a block of data" of the '710 claims; the equations of claim 1 of the '032 patent; the message passing decoder or Tanner graph of claim 18 of the '032 patent; or the memory locations or indices of the '833 claims).

567. Further, none of the other documents identified by Caltech as evidence of Conception predate the provisional applications to which the patents-in-suit claim priority.

XII. INVENTORSHIP

568. Divsalar's 1998 paper disclosed everything in '781 claim 19. I have been informed that Divsalar worked jointly in collaboration with the named inventors. Divsalar should have been named an inventor on that patent. All of the other asserted claims rely on the repeat-accumulate code disclosed by Divsalar. Divsalar should also have been named an inventor on the other patents.

XIII. MATERIALITY

569. I have been asked for my opinion on whether the following three references were material to the patentability of the claimed invention:

- Luby, M. et al., "Practical Loss-Resilient Codes," *STOC '97* (1997) (hereinafter, "Luby97")
- Luby, M. et al., "Analysis of Low Density Codes and Improved Designs Using Irregular Graphs," *STOC '98*, p. 249-259 (1998) (hereinafter, "Luby98")

- Richardson, T. et al. "Design of provably good low-density parity check codes," *IEEE Transactions on Information Theory* (1999) (preprint) (hereinafter, "Richardson99")

570. For the reasons set forth in detail below, each of these references was material to the patentability of the claims of the patents-in-suit. In particular, each reference teaches irregularity, a concept that is central to the claimed invention but which was not taught by any of the references that were before the Patent Office during prosecution.

571. For each of these three references, I rely upon the entire disclosure of the reference in forming my opinions with respect to materiality. Without limiting that basis in any way, certain aspects of each reference demonstrating their materiality are briefly discussed below. Additional aspects are set forth in the claim charts attached as Exhibits F, G, and H.

A. Luby97

572. Luby97 is material to the patentability of all asserted claims of the patents-in-suit because it teaches the concept of irregularity. (*See* Luby 97, *passim*; *see also* Khandekar Thesis at CALTECH000003301). In particular, Luby 97 teaches that making a regular code irregular will improve that code's performance. For example, Luby 97 teaches: "In contrast with many applications of random graphs in computer science, our graphs are not regular. Indeed, the analysis in Section 6 shows that it is not possible to approach channel capacity with regular graphs." (Luby 97 at 153.) Indeed, Luby 97 teaches that because regular graphs "cannot yield codes that are close to optimal," "irregular graphs are a necessary component of our design." (*Id.* at 151-52.) Thus, Luby 97 concludes, "irregular degree sequences are better than regular degree sequences." (*Id.* at 158.) This concept was not taught by any of the references that were before the Patent Office during prosecution.

573. Luby 97 describes the performance gain from converting a regular code into an irregular code. Luby 97 states, for example: "In this paper, we present codes that can be encoded and decoded in linear time while providing near optimal loss protection." (*Id.* at 151.) Irregular codes, Luby 97 explains, "can transmit over lossy channels at rates extremely close to capacity." (*Id.* at 150.)

574. Luby 97 also teaches how to convert a regular code into an irregular code. This "requires the careful choice of a random ***irregular*** bipartite graph, where the structure of the ***irregular*** graph is extremely important" (Luby97 at Abstract) (emphasis added). Luby 97 goes on to explain:

> "Our encoding and decoding algorithms are almost symmetrical. Both are extremely simple, computing exactly one exclusive-or operation for each edge in a randomly chosen bipartitie graph. As in many similar applications, the graph is chosen to be sparse, which immediately implies that the encoding and decoding algorithms are fast. Unlike many similar applications, the graph is not regular; instead it is quite irregular with a carefully chosen degree sequence."

575. (*Id.* at 151-52.) Note that the term "degree sequence" is equivalent to the term "degree profile", as referred to in the patents-in-suit, in Frey99 and in MacKay. Luby 97 goes on to provide the "tools" for how "to *design* good irregular degree sequences." (*Id.* at 152 (emphasis in original); *see also id.* at 153-59.)

576. Importantly, Luby 97 teaches that one way of encoding an irregular code is by using an irregular low-density generator matrix ("LDGM"):

> "The βn check bits of the code *C(B)* described in Section 2 can be computed by multiplying the vector of *n* message bits by the $\beta n \times n$ matrix, *M(B)*, whose (i, j)-th entry is 1 if there is an edge in *B* between left node *i* and right node *j* and is 0 otherwise (the multiplication is over the field of two elements). We choose our graphs *B* to be sparse, so that the resulting matrix *M(B)*is sparse and the multiplication can be performed quickly."

(*Id.* at 157; *see also id.* (teaching that "the average number of 1s per row in $M(B)$ is $n \ln(1/\varepsilon)$; so, the Gaussian elimination can be performed in time $O(n \ln(1/\varepsilon))$"); Divsalar Tr. (232:18-25, 238:20-241:8 ("what we've got here is a – in Luby '97, an irregular low-density generator matrix").)

577. Irregular LDGM codes are central to the claimed invention of the patents-in-suit. (*See, e.g.,*'710 patent, col. 3:54-55 and Fig. 4.) The inventors discussed among themselves that IRA codes "are just low-density GENERATOR matrix codes, followed by an accumulator." (*See* CALTECH000008667.) Each of the patents states that the outer coder of the claimed IRA codes "may be a low-density generator matrix (LDGM) coder that performs an irregular repeat of the k bits in the block, as shown in Fig.4." (*See, e.g.*, '710 patent, col. 3:51-54.) And dependent claims of the patents recite this embodiment explicitly. (*See, e.g.* '710 patent, claims 7 and 20; '032 patent, claim 6; '781 patent, claim 5.) Luby 97's disclosure of an irregular LDGM in the prior art would thus have been material to a Patent Examiner considering the patentability of the claims of the patents-in-suit.

578. In his doctoral thesis, Dr. Khandekar acknowledges that "Luby et al. also introduced the concept of irregularity" in error correction codes, which was a "major breakthrough" in 1997, and that IRA codes are merely an application of Luby's "concept of irregularity to the ensemble of RA codes" described in Divsalar. (CALTECH000003345, 3346; *see also* CALTECH000003293 (IRA codes "are adapted from the previously known class of repeat-accumulate (RA) codes"); CALTECH000003350 ("Having reviewed the basic properties of irregular LDPC codes, let us now apply the concept of irregularity to the ensemble of RA codes defined in Section 1.2.6 to get the ensemble of irregular RA codes."). Dr. Khandekar's thesis even shows the Tanner graph of Luby 97's irregular LDPC code (Fig. 3.1), the Tanner graph of Divsalar's RA code (Fig. 1.6), and how when combined these produce the Tanner graph of an IRA code (Fig. 3.2).

(CALTECH000003315, 3347, and 3350.) Showing similar awareness of Luby's materiality, Dr. Jin wrote to a Caltech colleague on May 4, 2000 – just fourteen days before filing his provisional application with the Patent Office – that "the papers on codes achieving BEC capacity are most written by Luby," that Luby's subject is "irregular low density parity check codes," and that Dr. Jin's "group is also working on that subject, … but that hasn't been disclosed yet." (CALTECH000008875 (emphasis added).) The Patent Examiner had a copy of Divsalar during prosecution of the patents-in-suit, but was never provided a copy of Luby 97 or informed that the claimed IRA codes were merely an application of Luby's "concept of irregularity" to Divsalar's RA codes. Nor was the Patent Examiner informed of the fact that, as Dr. Jin testified to during his deposition, the accumulator of the patents-in-suit is identical to the accumulator disclosed in Divsalar. (Jin Tr. at 122:7-13, 129:5-15, 134:12-18; *see also* Wicker Tr. at 87:2-9, 95:15-20, 109:9-20; Khandekar Tr. at 306:6-17.) Nor was the Patent Examiner told what Dr. Jin freely admitted to his Caltech colleague – that Luby 97 disclosed the same "subject" as his and his named co-inventor's work. It is more likely than not that the claims would not have issued in their present form had this information been disclosed to the Patent Office.

579. The applicants also made affirmative representations to the Patent Office regarding the patentablity of their then-pending patent claims that they could not have made had they disclosed Luby 97. In an office action dated September 3, 2004, the Patent Examiner rejected then pending claims of the '710 patent as invalid in light of U.S. Patent No. 6,014,411 to Wang (hereinafter, "Wang"). (CALTECH000000117-118, 120-124.) The applicants responded on November 22 2004 by arguing that their claims were patentable over Wang:

580. The encoding arrangement shown in Figure 5 of Wang uses a fixed repetition rate "r" …

> ...
>
> There is no indication in Wang that the rate r is irregular. Rather, all bits are repeated the same number of times, i.e., regularly.
>
> Each of independent claims 11, 15, and 24 recites that in a first encoding, bits are repeated "irregularly" or "a different number of times". Accordingly, Applicant submits that claims 11, 15, and 24, and their dependencies, are allowable.

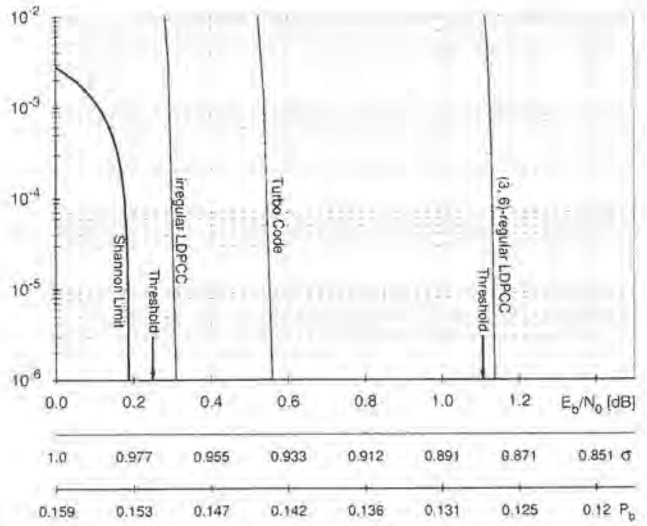
(CALTECH000000110-111.) The applicants' sole argument for the patentability of their claims over Wang was thus that "[t]here is no indication in Wang that the rate r is irregular," and "[r]ather, [in Wang] all bits are repeated the same number of times, i.e., regularly." (*Id.*) Because Luby 97 teaches that replacing a regular code with an irregular code produces substantially improved performance, as discussed above, this reference supplies the precise element that the applicants claimed was missing from Wang. Had Luby 97's prior art teaching to improve the performance of regular codes by making them irregular been disclosed to the Patent Office, the argument would have been significantly weakened, and the claims would not have issued in their present form (because, *e.g.*, the Examiner would have been equipped to respond by pointing out that making a regular code irregular – the basis for Caltech's distinction – was already well known in the art). For at least the reasons given above, Luby97 is material to the patentability of the claimed invention.

B. Luby98 and Richardson99

581. Luby98 and Richardson99 are also material to the patentability of the claims of the patents in suit because they teach irregular LDPC codes, which are not taught by any of the references that were before the Patent Office during prosecution.

582. Irregular LDPC codes are the primary focus of the Richardson99 paper (Richardson99 at 1) ("In this paper we present *irregular* low-density parity check

codes (LDPCCs) which exhibit performance extremely close to the best possible as determined by the Shannon capacity formula") (emphasis in original). Richardson99 includes experimental data indicating that irregular LDPC codes exhibit significantly lower error rates than both regular LDPC codes and regular turbocodes, as shown in Figure 2, reproduced below:



Richardson, Fig. 2, comparing performance of various codes

583. Acknowledging its materiality, Dr. Jin testified that Richardson99 is "very relevant to [the] patent," and that it "represents the best codes in irregular LDPC code." (Jin Tr. at 199:9-16.) After testifying that Richardson99 was material, however, Dr. Jin testified that he chose to instead disclose a non-prior art 2001 version of the paper to the Patent Office. (Jin Tr. at 211:7-14 ("Q. The question is: The version of Richardson and Urbanke that's actually disclosed here on the patents themselves is the 2001 version, correct? A. This is correct. We had their original preprint and I think that after their publication become official, that we changing [sic] to the official version of that paper."); *see also* CALTECH000023593 (showing publication of Richardson99 in April 1999).)

584. Similarly, Luby98 describes "error-correcting codes based on random ***irregular*** bipartite graphs, which we call ***irregular*** *codes*" (Luby98 at 249)

(emphasis added). In particular, Luby98 describes irregular Gallager codes (*i.e.*, LDPC codes), and teaches that adding irregularity to known regular Gallager codes, can significantly enhance decoding performance (*see id.*).

585. The idea of irregular codes is central to the claimed invention, and is not taught by any of the references that were before the Patent Office during prosecution of the patents-in-suit. As I explain above, the importance of irregularity to the claimed invention is underscored by the applicant's own statements about the Wang reference during prosecution of the '710 patent and their disclosure of the 2001 non-prior art version of Richardson.

586. For at least these reasons, the Luby98 and Richardson99 references are material to the patentability of the claimed invention.

XIV. <u>CLAIM CHARTS</u>

587. Attached hereto as exhibits B-E are claim charts that summarize the invalidity analysis presented herein. Attached hereto as exhibits F-H are claim charts that summarize the materiality analysis presented herein. The evidence presented in these charts is intended as a representative sample of the evidence relied upon in this report; it is not an exhaustive list of evidence upon which I rely.

XV. <u>TRIAL EXHIBITS</u>

588. I may rely on visual aids and demonstrative exhibits that demonstrate the bases of my opinions. Examples of these visual aids and demonstrative exhibits may include, for example, claim charts, patent drawings, excerpts from patent specifications, file histories, interrogatory responses, deposition testimony and deposition exhibits, as well as charts, diagrams, videos and animated or computer-generated video.

589. Other than as referred to in this report, I have not yet prepared any exhibits for use at trial as a summary or support for the opinions expressed in this report, but I expect to do so in accordance with the Court's scheduling order.

XVI. COMPENSATION

590. I am being paid at my ordinary and customary hourly rate of $600, plus expenses, for my time spent working on this matter. My compensation does not depend on the outcome of this case.

XVII. SUPPLEMENTATION OF OPINIONS

591. I reserve the right to supplement my opinions after I have and the opportunity to review expert reports or other materials from Plaintiff or other additional documents or materials that are brought to my attention.

APPENDIX A

Mathematical Representations of Error-Correcting Codes

592. Coding theorists often think of error-correcting codes in linear-algebraic terms. Linear algebra is the branch of mathematics that deals with vectors and the linear transformations that can be applied to vectors.[68]

593. In linear-algebraic terms, a k-bit block of information bits is a k-dimensional vector of bits and an n-bit codeword is an n-dimensional vector of bits. The encoding process, which converts blocks of information bits into codewords, is a linear transformation that maps k-dimensional bit vectors to n-dimensional bit vectors. This transformation is represented by a $k \times n$ matrix G called a *generator matrix*. For an information vector $\mathbf{u} = [u_1, u_2, u_3, \ldots, u_k]$ the codeword $\mathbf{x} = [x_1, x_2, x_3, \ldots, x_n]$ is given by: $\mathbf{x} = \mathbf{u}G$, where:

$$\mathbf{x} = \mathbf{u}G = \begin{bmatrix} \sum_{i=1}^{k} G_{i,1} u_i \\ \sum_{i=1}^{k} G_{i,2} u_i \\ \sum_{i=1}^{k} G_{i,3} u_i \\ \vdots \\ \sum_{i=1}^{k} G_{i,n} u_i \end{bmatrix}$$

594. The image of G, denoted $\mathrm{Im}(G)$, represents the set of n-dimensional vectors that are valid codewords. Because $k < n$, G is not surjective, meaning that not all n-dimensional vectors are valid codewords. A $(n\text{-}k) \times n$ matrix H, called a *parity check matrix*, can be used to determine whether a particular n-dimensional vector

[68] A linear transformation is a mathematical function that preserves addition and scalar multiplication. More formally, a function f is linear if and only if, for all x, y, and α: $\alpha f(x + y) = f(\alpha x) + f(\alpha y)$. Every matrix represents a linear transformation.

is a valid codeword. In particular, for an n-dimensional vector $\mathbf{x}$**,** $\mathbf{x}$ is a valid codeword if and only if $H\mathbf{x} = \mathbf{0}$. In linear-algebraic terms, the image of G is equal to the kernel of H.

595. Each of the n - k rows of the parity-check matrix H represents an equation that a valid codeword must satisfy. For example, consider a codeword $\mathbf{x}$ and a parity check matrix H given as follows:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \quad H = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

596. If $\mathbf{x}$ is a valid codeword, the product $H\mathbf{x}$ must be equal to $\mathbf{0}$, so we have:

$$H\mathbf{x} = \begin{bmatrix} x_3 + x_4 \\ x_1 + x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \mathbf{0}$$

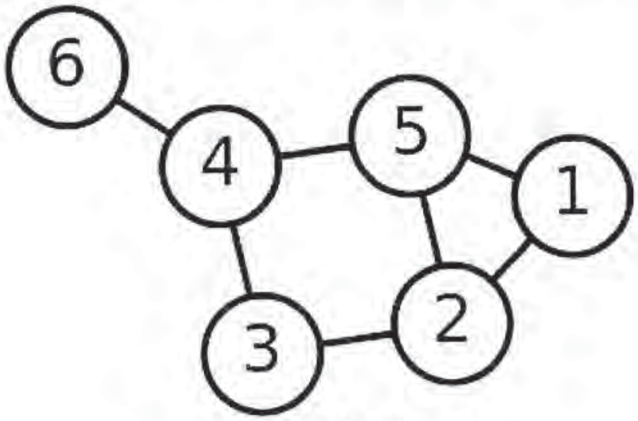
As this equation shows, the first row of H represents the constraint that $x_3 + x_4 = 0$, and the second row of H represents the constraint that $x_1 + x_2 = 0$. If the vector $\mathbf{x}$ satisfies both of these constraints, it is a valid codeword.

597. In practice, parity-check matrices have hundreds or thousands of rows, each of which represents an equation of the form $x_a + x_b + \ldots + x_z = 0$, similar to those shown in the above example. These equations are called *parity-check* equations.

598. Another popular mathematical representation of error-correcting codes is the "Tanner Graph." Tanner graphs were named after R. Michael Tanner, who described the concept in a 1981 publication titled "A Recursive Approach to Low

Complexity Codes."[69] A Tanner graph is a graphical depiction of the parity matrix *H*.

599. A "graph" in this context is a group of objects, or *nodes*, that may be linked together by connections called *edges*. A simple graph is shown below:

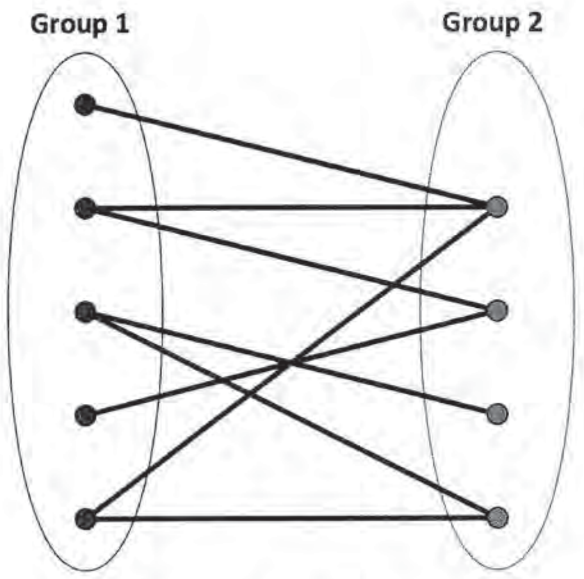


A Simple Graph

600. The nodes in the graph above are represented by the circles labeled 1 through 6, and the edges are represented as lines connecting the nodes (*e.g.*, the straight line connecting nodes 6 and 4 is an edge). When two nodes are connected by an edge, we say that the nodes are *adjacent*.

601. Tanner graphs are part of a class of graphs called *bipartite* graphs. A bipartite graph is a graph whose nodes can be divided into two groups, such that every edge connects a node from one group to a node from the other (*i.e.*, no two nodes in the same group are adjacent). A bipartite graph is shown below:

[69] Tanner, R. M., "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, pp. 533–547 (September 1981).



A Simple Bipartite Graph

The two groups of nodes in the bipartite graph above are labeled "Group 1" and "Group 2." Note that no node in Group 1 is connected to any other node in Group 1, and no node in Group 2 is connected to any other node in Group 2.

602. Tanner graphs are bipartite graphs that represent error-correcting codes. A Tanner graph includes one group of nodes called *variable nodes* that correspond to the information and parity bits, and a second group of nodes *check nodes* that represent the relationship between the parity and information bits. The variable nodes include two types of nodes: *information nodes* that correspond to information bits input to the code, and *parity nodes*, that correspond to parity bits. Generally, a Tanner graph will have n variable nodes and $n - k$ check nodes, where n is the number of bits in the codeword, and k is the number of information bits per block. Variable nodes are not connected to other variable nodes, and check nodes are not connected to other check nodes (this is what makes Tanner graphs bipartite).

603. Intuitively, one can think of a Tanner graph as a representation of the interrelationships among the bits of a codeword. Each variable node v_i corresponds to a bit b_i in the codeword. Each check node represents a mathematical

relationship among the bits to which it is connected. Specifically, when a check node is connected to variable nodes v_1, v_2, v_3, … v_r, it means that the corresponding bits of the codeword must add up to 0. That is: $b_1 \oplus b_2 \oplus b_3 \oplus \ldots \oplus b_r = 0$. Each check node of the Tanner graph represents a different group of encoded bits that must sum to 0.

604. As I mentioned above, a Tanner graph for a particular code is a graphical depiction of that code's parity-matrix H. In a Tanner graph, each of the variable nodes v_1 ... v_n represents a bit in the codeword, and each of the check nodes c_1 ... c_{n-k} represents a parity-check equation. As I explained earlier, each column of H represents a bit of the codeword, and each row of H represents a parity-check equation that a valid codeword must satisfy. Thus, the variable nodes and check nodes correspond to the columns and rows of H, respectively. The edges of the Tanner graph correspond to the 1s in the parity-check matrix: if there is a 1 at the i^{th} row and the j^{th} column of H (*i.e.*, if $H_{i,j} = 1$), then there is an edge connecting the i^{th} check node to the j^{th} variable node. Conversely, if $H_{i,j} = 0$, the i^{th} check node and the j^{th} variable node are not connected.

605. Matrices and Tanner graphs are two equivalent ways of describing error-correcting cods. Every linear code has a matrix representation and a Tanner graph representation.

606. A related graphical representation of codes is the factor graph (Kschischang et al, IEEE Trans Inform Theory Vol 47, No 2, pp 498-519, February 2001). A factor graph is more general than a Tanner graph in two ways: (a) Some of the variable nodes may be unobserved, i.e., in the context of coding some variables may not correspond to information bits or parity bits; (b) The check nodes can implement more general functional relationships between the variables and can even represent continuous relationships such as those found in probability theory. Convolutional codes can be represented by factor graphs, where there are three

types of variable: (a) variables corresponding to information bits; (b) variables corresponding to parity bits; and (c) variables that correspond to the memory of the convolutional code. The latter variables are usually not transmitted over the channel. In a truncated convolutional code, the information bits themselves may not be transmitted, resulting in a non-systematic code. Or, some parity bits may be punctured. In all of these scenarios, an iterative sum-product decoding algorithm can be used to determine the codeword and the information bits, given the output of the channel.

607. It is widely recognized that Tanner graphs can be modified slightly to allow for variables that are not transmitted across the channel, such as variables corresponding to the memory of a convolutional code. Consequently, within the context of coding, Tanner graphs can be used to represent codes with such "unobserved variables", and in this report I frequently refer to Tanner graphs with this additional functionality.

Date: January 28, 2015

[illegible]

Dr. Brendan Frey, PhD.