US 20140229788A1

(19) **United States**

(12) **Patent Application Publication**

Richardson

(10) **Pub. No.: US 2014/0229788 A1**

(43) **Pub. Date: Aug. 14, 2014**

(54) **LDPC DESIGN FOR HIGH RATE, HIGH PARALLELISM, AND LOW ERROR FLOOR**

(71) Applicant: **QUALCOMM Incorporated**, San Diego, CA (US)

(72) Inventor: **Thomas Joseph Richardson**, South Orange, NJ (US)

(73) Assignee: **QUALCOMM Incorporated**, San Diego, CA (US)

(21) Appl. No.: **14/179,871**

(22) Filed: **Feb. 13, 2014**

Related U.S. Application Data

(60) Provisional application No. 61/764,476, filed on Feb. 13, 2013.

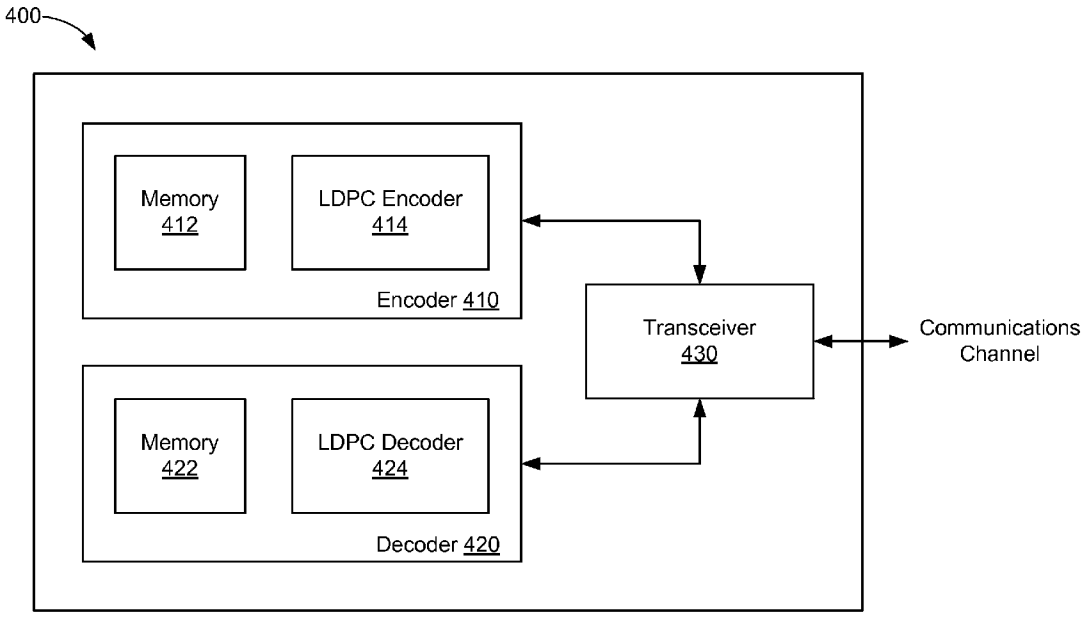
Publication Classification

(51) **Int. Cl.**
H03M 13/11 (2006.01)

(52) **U.S. Cl.**
CPC ***H03M 13/1105*** (2013.01)
USPC **714/752**

(57) **ABSTRACT**

A method of data encoding is disclosed. An encoder receives a set of information bits and performs a lifted LDPC encoding operation on the information bits to produce a codeword. The encoder then punctures all lifted bits of the codeword that correspond to one or more punctured base bits of a base LDPC code used for the LDPC encoding operation. The base LDPC code has no multiple edges, and the one or more punctured base bits are those that correspond with one or more punctured base nodes, respectively, of the base LDPC code. For some embodiments, the one or more punctured base nodes correspond to one or more degree 2 variable nodes.



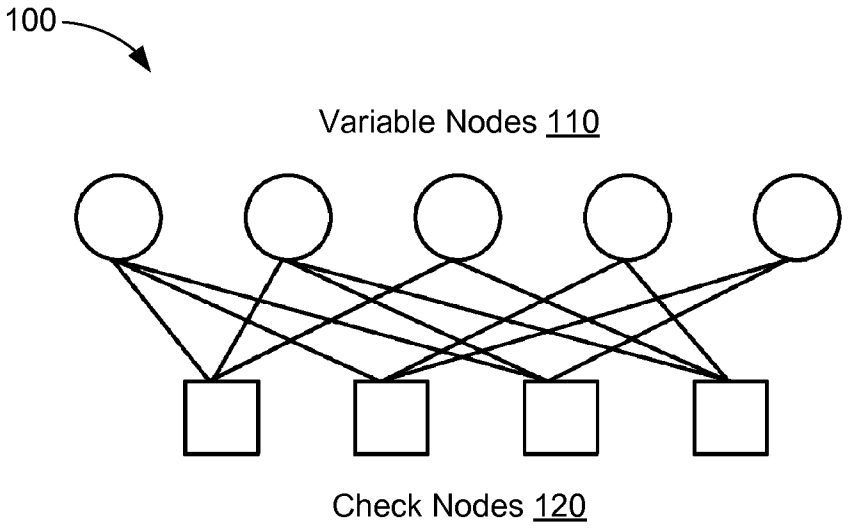

100
Variable Nodes 110
Check Nodes 120

FIG. 1A

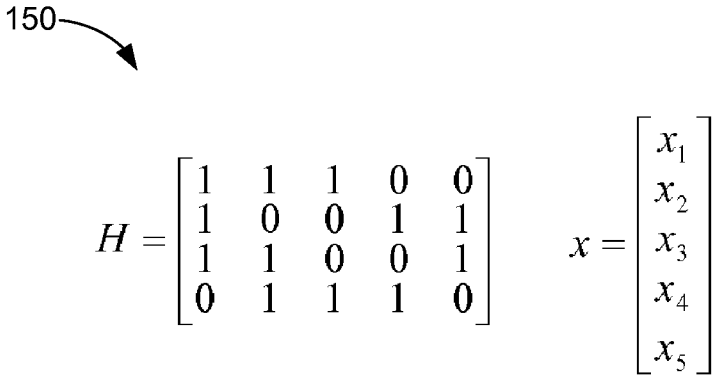

150
$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$
$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$

FIG. 1B

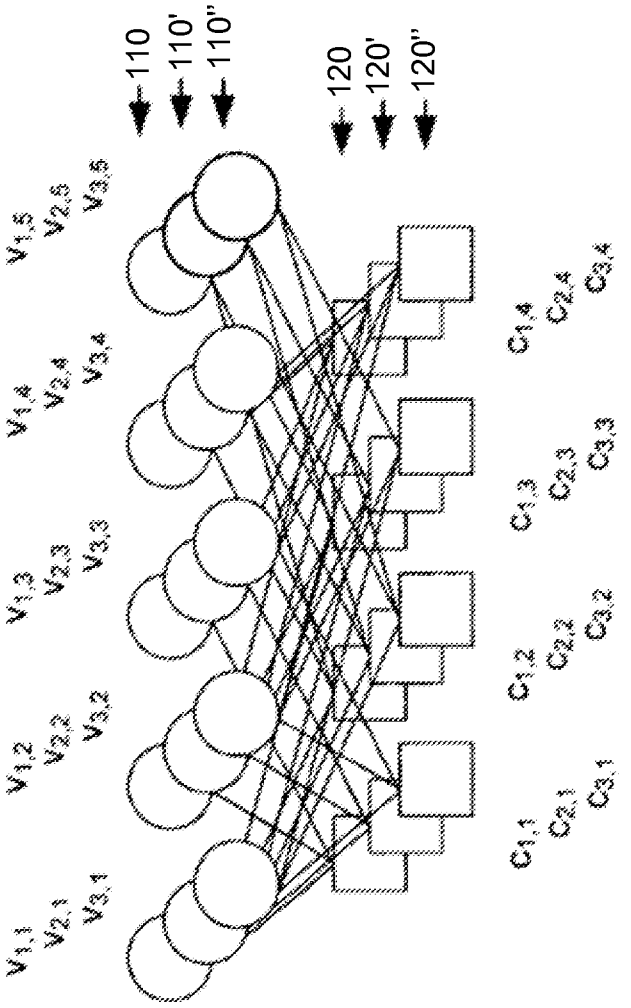

110
110'
110''
120
120'
120''
V1,1
V2,1
V3,1
V1,2
V2,2
V3,2
V1,3
V2,3
V3,3
V1,4
V2,4
V3,4
V1,5
V2,5
V3,5
C1,1
C2,1
C3,1
C1,2
C2,2
C3,2
C1,3
C2,3
C3,3
C1,4
C2,4
C3,4

FIG. 2

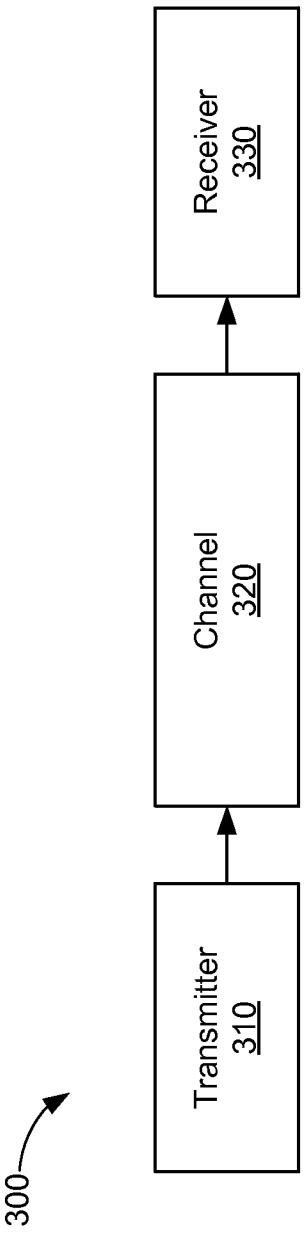


FIG. 3

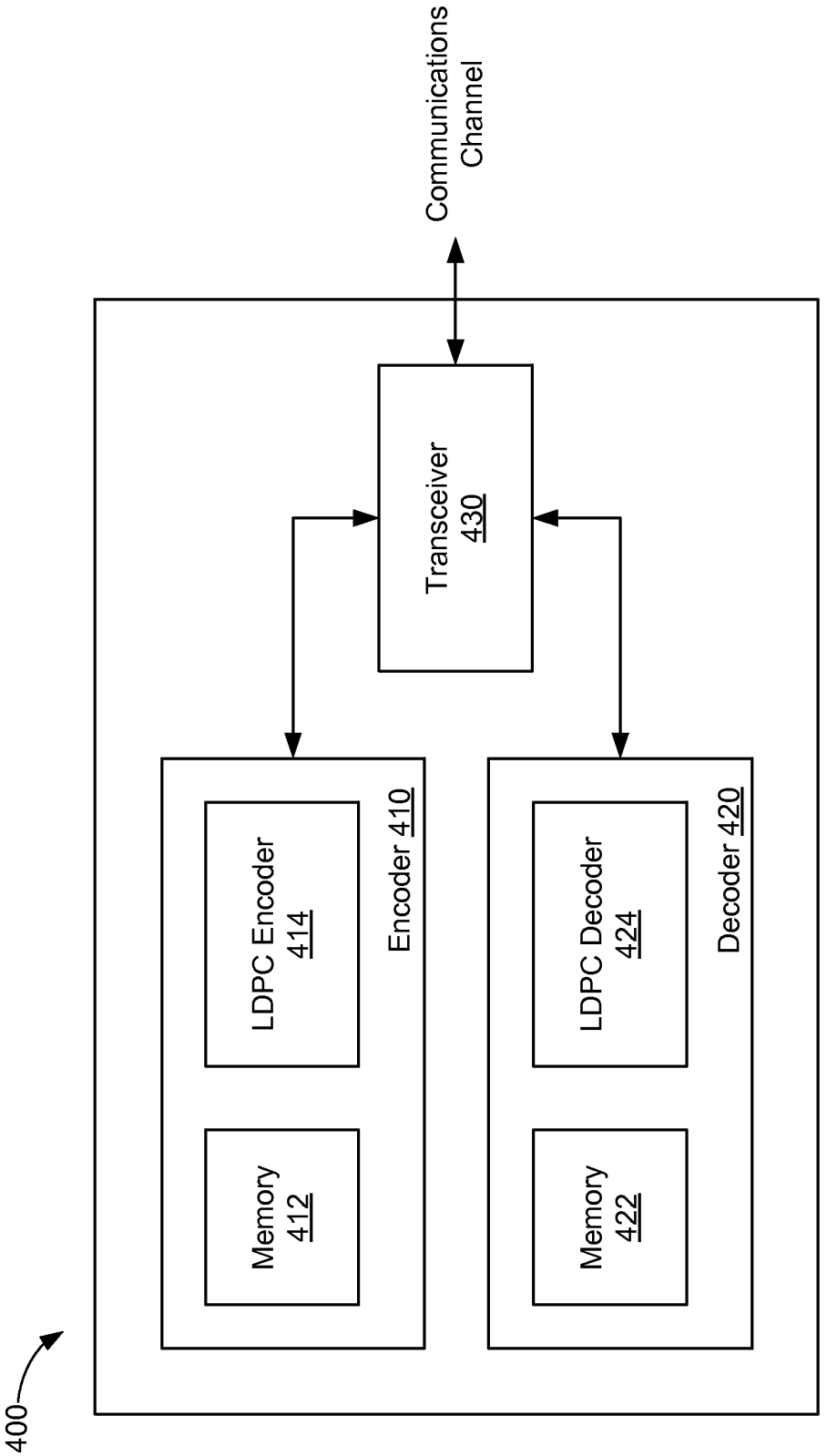

Communications Channel
Transceiver 430
LDPC Encoder 414
Memory 412
Encoder 410
LDPC Decoder 424
Memory 422
Decoder 420
400

FIG. 4

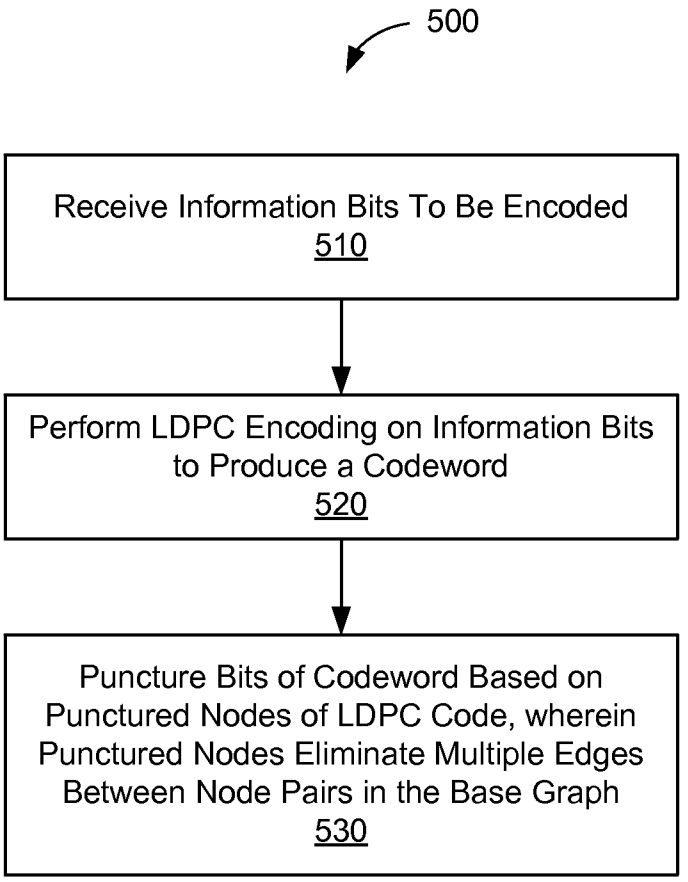

500
Receive Information Bits To Be Encoded
510
Perform LDPC Encoding on Information Bits to Produce a Codeword
520
Puncture Bits of Codeword Based on Punctured Nodes of LDPC Code, wherein Punctured Nodes Eliminate Multiple Edges Between Node Pairs in the Base Graph
530

FIG. 5

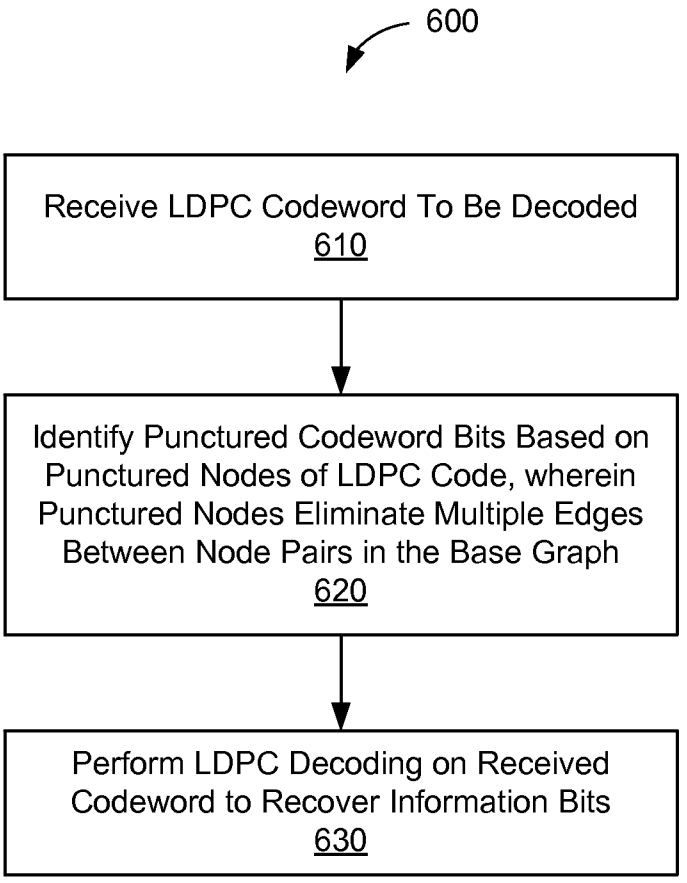

600
Receive LDPC Codeword To Be Decoded
610
Identify Punctured Codeword Bits Based on Punctured Nodes of LDPC Code, wherein Punctured Nodes Eliminate Multiple Edges Between Node Pairs in the Base Graph
620
Perform LDPC Decoding on Received Codeword to Recover Information Bits
630

FIG. 6

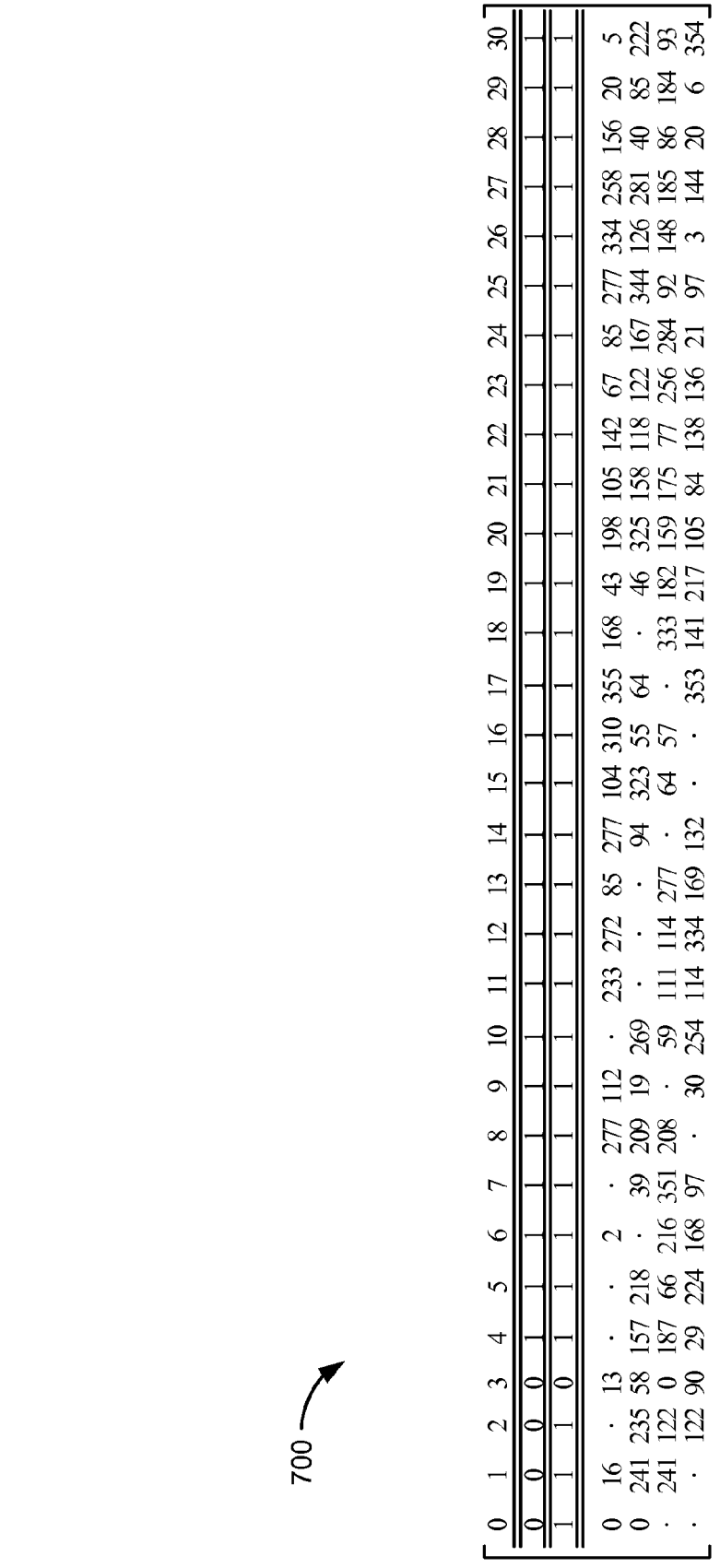

700
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 16 · 13 · · 2 · 277 112 · 233 272 85 277 104 310 355 168 43 198 105 142 67 85 277 334 258 156 20 5
0 241 235 58 157 218 · 39 209 19 269 · · · 94 323 55 64 · 46 325 158 118 122 167 344 126 281 40 85 222
· 241 122 0 187 66 216 351 208 · 59 111 114 277 · 64 57 · 333 182 159 175 77 256 284 92 148 185 86 184 93
· · 122 90 29 224 168 97 · 30 254 114 334 169 132 · · 353 141 217 105 84 138 136 21 97 3 144 20 6 354

FIG. 7

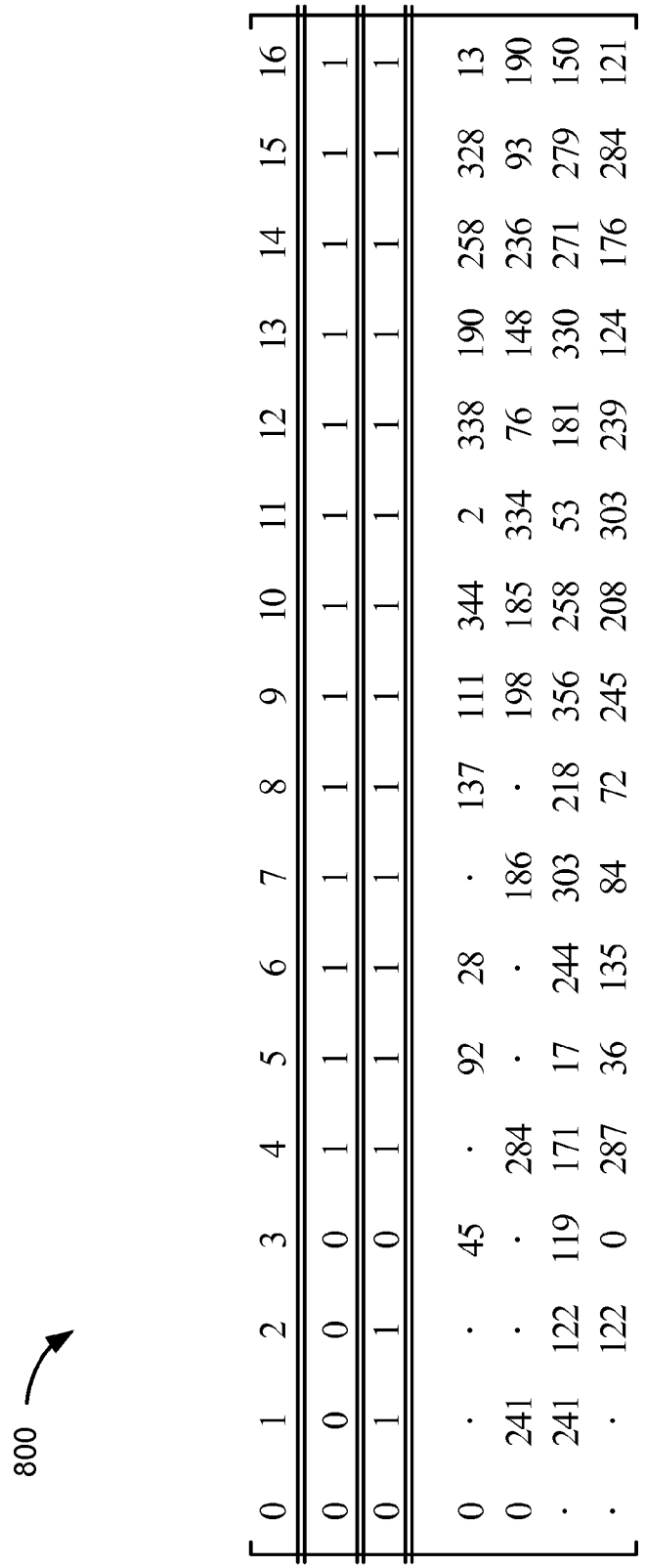

800
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1
0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1
0 . . 45 . 92 28 . 137 111 344 2 338 190 258 328 13
0 241 . . 284 . . 186 . 198 185 334 76 148 236 93 190
. 241 122 119 171 17 244 303 218 356 258 53 181 330 271 279 150
. . 122 0 287 36 135 84 72 245 208 303 239 124 176 284 121

FIG. 8

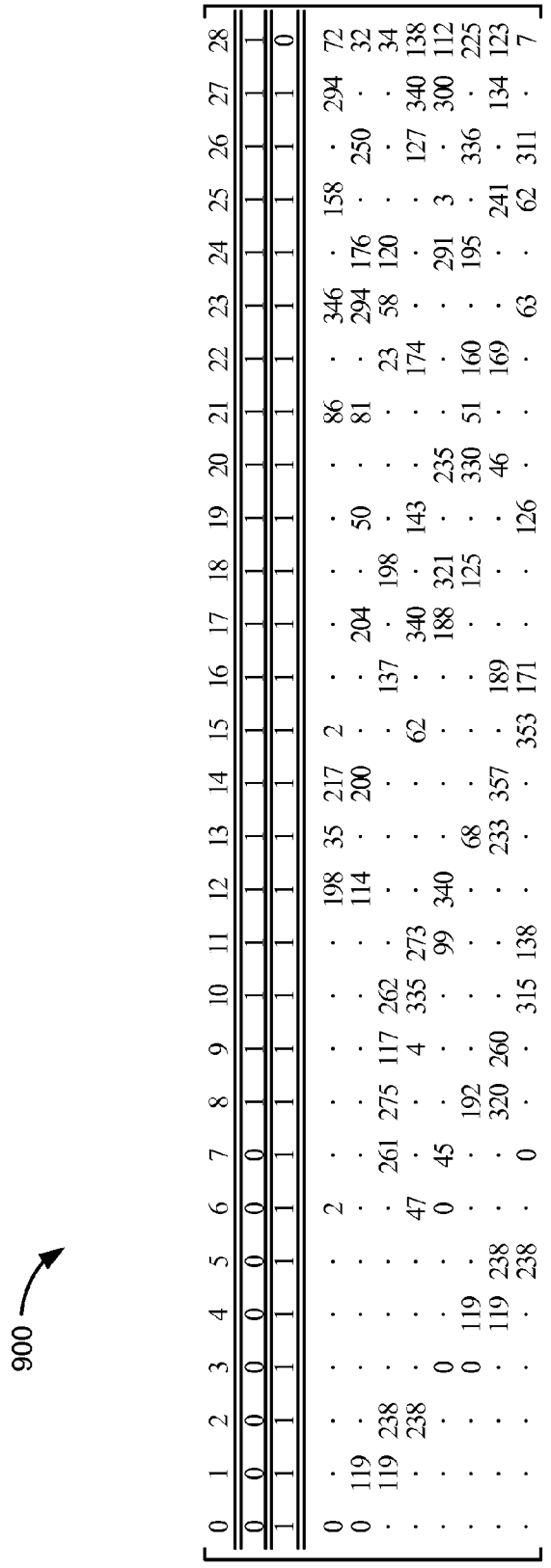

900
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
0 . . . . . 2 . . . . . 198 35 217 2 . . . . 86 . 346 . 158 . 294 72
0 119 . . . . 114 . 200 . . 204 . 50 . 81 . 294 176 . 250 . 32
. 119 238 . . . . . 261 275 117 262 . . . 137 . 198 . . . 23 58 120 . . . 34
. . 238 . . . . 47 . . 4 335 273 . . . 62 . 340 . 143 . . . 174 . . . 127 340 138
. . . 0 . . 0 45 . . . 99 340 . . . . . 188 321 . 235 . . . . 291 3 . 300 112
. . . 0 119 . . . 192 . . . . . 68 . . . . 125 . 330 51 160 . . 195 . 336 . 225
. . . . 119 238 . . 320 260 . . . 233 357 . 189 . . . 46 . 169 . . . 241 . 134 123
. . . . . 238 . 0 . . . 315 138 . . . 353 171 . . 126 . . . 63 . . 62 311 . 7

FIG. 9

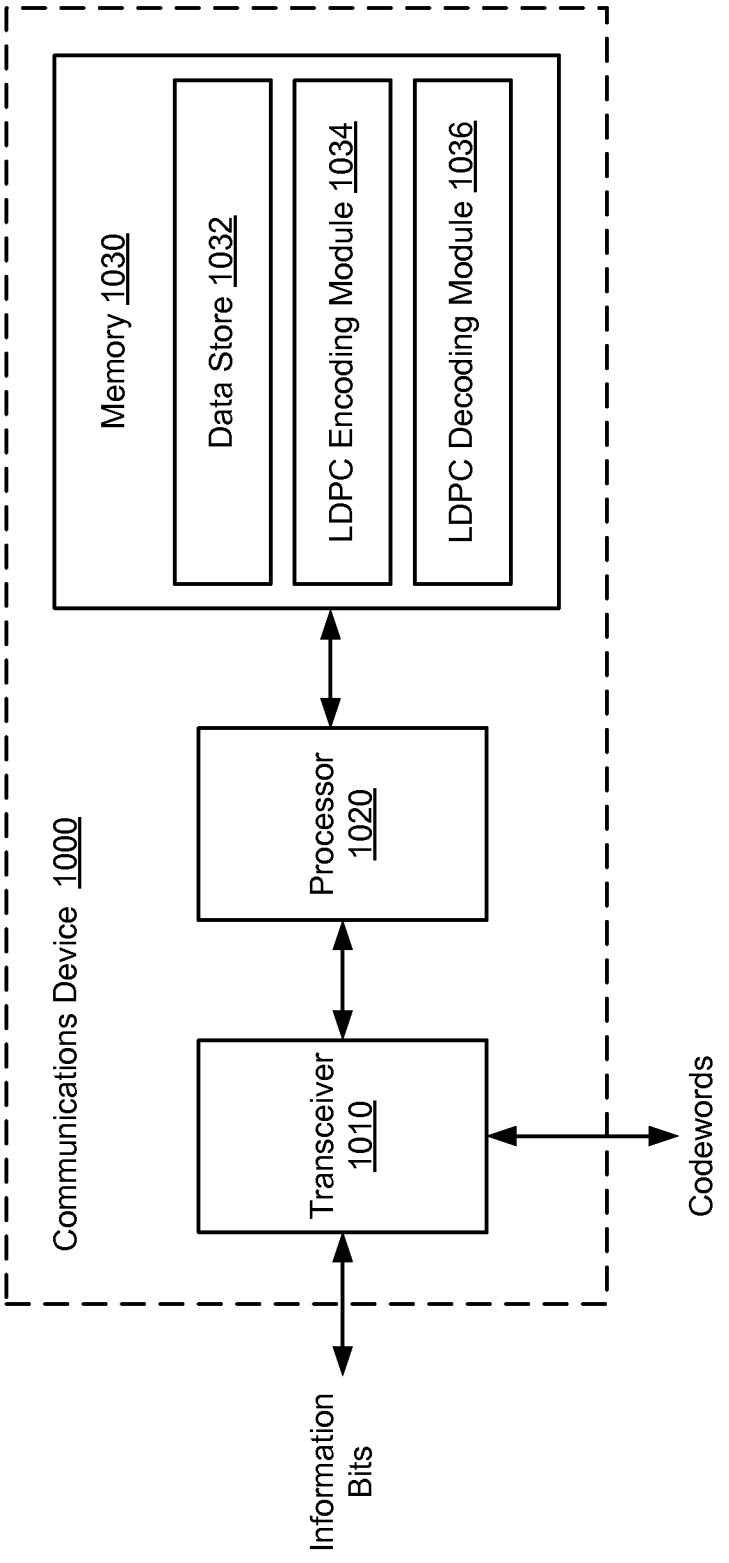

Communications Device 1000
Memory 1030
Data Store 1032
LDPC Encoding Module 1034
LDPC Decoding Module 1036
Processor 1020
Transceiver 1010
Information Bits
Codewords

FIG. 10

LDPC DESIGN FOR HIGH RATE, HIGH PARALLELISM, AND LOW ERROR FLOOR

RELATED APPLICATIONS

[0001] This application claims priority under 35 U.S.C. §119(e) of the co-pending and commonly-owned U.S. Provisional Patent Application No. 61/764,476, titled "LDPC Design for High Rate, High Parallelism, Low Error Floor, and Simple Encoding," filed Feb. 13, 2013, which is hereby incorporated by reference in its entirety.

TECHNICAL FIELD

[0002] The present embodiments relate generally to communications and data storage systems, and specifically to communications and data storage systems that use LDPC codes.

BACKGROUND OF RELATED ART

[0003] Many communications systems use error-correcting codes. Specifically, error correcting codes compensate for the intrinsic unreliability of information transfer in these systems by introducing redundancy into the data stream. Low density parity check (LDPC) codes are a particular type of error correcting codes which use an iterative coding system. LDPC codes can be represented by bipartite graphs (often referred to as "Tanner graphs"), wherein a set of variable nodes corresponds to bits of a codeword, and a set of check nodes correspond to a set of parity-check constraints that define the code. A variable node and a check node are considered "neighbors" if they are connected by an edge in the graph. A bit sequence having a one-to-one association with the variable node sequence is a valid codeword if and only if, for each check node, the bits associated with all neighboring variable nodes sum to zero modulo two (i.e., they include an even number of 1's).

[0004] For example, FIG. **1**A shows a bipartite graph **100** representing an exemplary LDPC code. The bipartite graph **100** includes a set of 5 variable nodes **110** (represented by circles) connected to 4 check nodes **120** (represented by squares). Edges in the graph **100** connect variable nodes **110** to the check nodes **120**. FIG. **1**B shows a matrix representation **150** of the bipartite graph **100**. The matrix representation **150** includes a parity check matrix H and a codeword vector x, where x1-x5 represent bits of the codeword x. More specifically, the codeword vector x represents a valid codeword if and only if Hx=0. FIG. **2** graphically illustrates the effect of making three copies of the graph of FIG. **1**A, for example, as described in commonly owned U.S. Pat. No. 7,552,097. Three copies may be interconnected by permuting like edges among the copies. If the permutations are restricted to cyclic permutations, then the resulting graph corresponds to a quasi-cyclic LDPC with lifting Z=3. The original graph from which three copies were made is referred to herein as the base graph.

[0005] A received LDPC codeword can be decoded to produce a reconstructed version of the original codeword. In the absence of errors, or in the case of correctable errors, decoding can be used to recover the original data unit that was encoded. LDPC decoder(s) generally operate by exchanging messages within the bipartite graph **100**, along the edges, and updating these messages by performing computations at the nodes based on the incoming messages. For example, each variable node **110** in the graph **100** may initially be provided with a "soft bit" (e.g., representing the received bit of the codeword) that indicates an estimate of the associated bit's value as determined by observations from the communications channel. Using these soft bits the LDPC decoders may update messages by iteratively reading them, or some portion thereof, from memory and writing an updated message, or some portion thereof, back to, memory. The update operations are typically based on the parity check constraints of the corresponding LDPC code. In implementations for lifted LDPC codes, messages on like edges are often processed in parallel.

[0006] LDPC codes designed for high speed applications often use quasi-cyclic constructions with large lifting factors and relatively small base graphs to support high parallelism in encoding and decoding operations. LDPC codes with higher code rates (e.g., the ratio of the message length K to the codeword length N) tend to have relatively fewer parity checks. If the number of base parity checks is smaller than the degree of a variable node (e.g., the number of edges connected to a variable node), then, in the base graph, that variable node is connected to at least one of the base parity checks by two or more edges (e.g., the variable node may have a "double edge"). Having a based variable node and a base check node connected by two or more edges is generally undesirable for parallel hardware implementation purposes. For example, such double edges may result in multiple concurrent read and write operations to the same memory locations, which in turn may create data coherency problems. Pipelining of parallel message updates can be adversely affected by the presence of double edges.

SUMMARY

[0007] This Summary is provided to introduce in a simplified form a selection of concepts that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to limit the scope of the claimed subject matter.

[0008] A device and method of operation are disclosed that may aid in the encoding and/or decoding of low density parity check (LDPC) codewords. It is noted that the addition of a punctured variable node (also known as a state variable node) into the base graph design can effectively increase the number of checks in the graph by one without changing the rate parameters (k and n) of the code. For some embodiments, an encoder may receive a set of information bits and perform an LDPC encoding operation on the information bits to produce a codeword. The device may then puncture a set of lifted codeword bits corresponding to one or more base variable nodes based on a lifted LDPC code used for the LDPC encoding operation, wherein the punctured bits correspond with one or more punctured base variable nodes, respectively, of the base LDPC graph. It is understood that punctured variable nodes in the graphical description of the code can be eliminated from the description by a check node combining process operating on the lifted parity check matrix. Therefore, at least one of the one or more punctured base nodes is understood to eliminate multiple edges between node pairs of the base graph for the lifted LDPC code when the elimination of the punctured variable node results in multiple edges.

[0009] For some embodiments, the one or more punctured nodes may include a variable node having a degree equal to, or one less than, a number of check nodes of the LDPC code. For example, at least one of the punctured nodes may be a highest-degree variable node of the LDPC code. In such an

embodiment, the high degree of the node is often desirable for enhancing the performance of the code. For example, the puncturing allows higher variable node degree while avoiding double edges in the base graph. The presence of the punctured variable node in the graph effectively increases the number of check nodes that would otherwise be present in a base graph of a code of the same size and rate. For other embodiments, at least one of the punctured nodes may be a degree two variable node used to split a check node that would otherwise be connected to a variable node of the LDPC code by two or more edges. A punctured degree two node can be eliminated from the description by adding the two parity checks to which it is connected. The at least one punctured base degree two variable node may thus be used to eliminate double edges in the base LDPC graph. Similarly, a high degree punctured node may be eliminated from a parity check matrix representation by an elimination process summing constraint nodes to effectively reduce the degree of the variable node to one. A degree one punctured node can be eliminated from the graph along with its neighboring check node without altering the code. Such an elimination process is likely to introduce double or multiple edges into the representation which is undesirable for parallel implementation of decoding.

[0010] By eliminating or reducing double (or multiple) edges from the base LDPC graph, the present embodiments may reduce the complexity of the hardware that performs LDPC decoding operations in parallel, thereby increasing the processing efficiency of LDPC decoders that implement lifted LDPC codes. This further simplifies read and/or write operations performed in memory, and ensures that the read and write operations are not performed out of order. By allowing larger variable node degrees, while avoiding double edges, the present embodiments may also improve the error correcting performance of the LDPC coding system.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] The present embodiments are illustrated by way of example and are not intended to be limited by the figures of the accompanying drawings, where:

[0012] FIGS. **1**A-**1**B show graphical and matrix representations of an exemplary LDPC code;

[0013] FIG. **2** graphically illustrates the effect of making three copies of the graph of FIG. **1**A;

[0014] FIG. **3** shows a communications system in accordance with some embodiments;

[0015] FIG. **4** is a block diagram of a communications device in accordance with some embodiments;

[0016] FIG. **5** is an illustrative flow chart depicting an LDPC encoding operation in accordance with some embodiments;

[0017] FIG. **6** is an illustrative flow chart depicting an LDPC decoding operation in accordance with some embodiments;

[0018] FIG. **7** shows an exemplary parity check matrix associated with an LDPC code with rate r=27/30;

[0019] FIG. **8** shows an exemplary parity check matrix associated with an LDPC code with rate r=13/15;

[0020] FIG. **9** shows an exemplary parity check matrix associated with an LDPC code with rate r=21/28; and

[0021] FIG. **10** is a block diagram of a communications device in accordance with some embodiments.

DETAILED DESCRIPTION

[0022] In the following description, numerous specific details are set forth such as examples of specific components, circuits, and processes to provide a thorough understanding of the present disclosure. The term "coupled" as used herein means connected directly to or connected through one or more intervening components or circuits. Also, in the following description and for purposes of explanation, specific nomenclature is set forth to provide a thorough understanding of the present embodiments. However, it will be apparent to one skilled in the art that these specific details may not be required to practice the present embodiments. In other instances, well-known circuits and devices are shown in block diagram form to avoid obscuring the present disclosure. Any of the signals provided over various buses described herein may be time-multiplexed with other signals and provided over one or more common buses. Additionally, the interconnection between circuit elements or software blocks may be shown as buses or as single signal lines. Each of the buses may alternatively be a single signal line, and each of the single signal lines may alternatively be buses, and a single line or bus might represent any one or more of a myriad of physical or logical mechanisms for communication between components. The present embodiments are not to be construed as limited to specific examples described herein but rather to include within their scope all embodiments defined by the appended claims.

[0023] FIG. **3** shows a communications system **300** in accordance with some embodiments. A transmitter **310** transmits a signal onto a channel **320**, and a receiver **330** receives the signal from the channel **320**. The transmitter **310** and receiver **330** may be, for example, computers, switches, routers, hubs, gateways, and/or similar devices. In some embodiments, the channel **320** is wireless. In other embodiments, the channel **320** is a wired link (e.g., a coaxial cable or other physical connection).

[0024] Imperfections of various components in the communications system **300** may become sources of signal impairment, and thus cause signal degradation. For example, imperfections in the channel **320** may introduce channel distortion, which may include linear distortion, multi-path effects, and/or Additive White Gaussian Noise (AWGN). To combat potential signal degradation, the transmitter **310** and the receiver **330** may include LDPC encoders and decoders. Specifically, the transmitter **310** may perform LDPC encoding on outgoing data to produce a codeword that can be subsequently decoded by the receiver **330** (e.g., through an LDPC decoding operation) to recover the original data. For some embodiments, the transmitter **310** may transmit LDPC-encoded codewords with one or more "punctured" bits, for example, based on an LDPC code with one or more punctured variable nodes.

[0025] "Lifting" enables LDPC codes to be implemented using parallel encoding and/or decoding implementations while also reducing the complexity typically associated with large LDPC codes. More specifically, lifting is a technique for generating a relatively large LDPC code from multiple copies of a smaller base code. For example, a lifted LDPC code may be generated by producing a number (Z) of parallel copies of the base graph and then interconnecting the parallel copies through permutations of edge clusters of each copy of the base graph. A more detailed discussion of lifted LDPC codes may be found, for example, in the book titled, "Modern Cod-

ing Theory," published Mar. 17, 2008, by Tom Richardson and Ruediger Urbanke, which is hereby incorporated by reference in its entirety.

[0026] For example, when processing a codeword with lifting size Z, an LDPC decoder may utilize Z processing elements to perform parity check or variable node operations on all Z edges of a lifted graph concurrently. Specifically, each parity check operation may involve reading a corresponding soft bit value from memory, combining the soft bit value with other soft bit values associated with the check node and writing a soft bit back to memory that results from the check node operation. Double edges in the base graph may trigger parallel reading of the same soft bit value, in a memory location, twice during a single parallel parity check update. Additional circuitry may thus be needed to combine the soft bit values that are written back to memory, so as to properly incorporate both updates. Eliminating double edges in the base graph helps to avoid this extra complexity.

[0027] By eliminating or reducing double (and/or multiple) edges from the base LDPC code, the puncturing may reduce the complexity of the hardware that performs parallel check node or variable node operations, thereby increasing the parallel processing efficiency of a corresponding LDPC decoder. This further simplifies read and/or write operations performed in memory, and ensures that the read and write operations are not performed out of order.

[0028] Puncturing is the act of removing bits from a codeword to yield a shorter codeword. Thus, punctured variable nodes correspond to codeword bits that are not actually transmitted. Puncturing a variable node in an LDPC code creates a shortened code (e.g. due to the removal of a bit), while also effectively removing a check node. Specifically, for a matrix representation of an LDPC code, including bits to be punctured, where the variable node to be punctured has a degree of one (such a representation may be possible through row combining provided the code is proper), puncturing the variable node removes the associated bit from the code and effectively removes its single neighboring check node from the graph. As a result, the number of check nodes in the graph is reduced by one. If the base transmitted block length is n−p, where p is the number of punctured columns, and the number of base parity checks is m, then the rate is (n−m)/(n−p). The binary information block size is (n−m)*Z, and the transmitted block size is (n−p)*Z. Note that if we increase n and p by 1 we may increase m by 1 and leave the rate and block size unchanged.

[0029] As an example, consider a rate 0.9 code in which the base block length is 30. Without puncturing, the number of check nodes that would be used to define the base code is 3, resulting in a (27, 30) code (e.g., K=27 message bits, N=30 codeword bits). Such a code is likely to have double (or more) edges connecting at least one check node to a variable node (e.g., unless all variable nodes have a maximum degree of 3). However, it may be desirable to have larger degree variable nodes (e.g., degree≥4), for example, to ensure a deep error floor. If a punctured variable node is introduced into the LDPC code, thereby increasing the total number of variable nodes to 31, then the number of base check nodes increases to 4. It is now possible to have degree 4 base variable nodes without double edges in the base graph. We note, however, such an LDPC code is still a (27, 30) code.

[0030] In the bipartite graph representation of an LDPC code, a punctured degree-two variable node effectively merges its two neighboring check nodes into a single check node. The punctured degree two variable node effectively indicates that its two neighboring check nodes have, absent the degree two node, the same parity. Accordingly, punctured degree-two variable nodes may be used to "split" check nodes, thereby appearing to increase the total number of check nodes. This mechanism may therefore be used to remove multiple edges from an LDPC code. A variable node is typically connected to at least one check node by two or more edges if the degree of the variable node is greater than the total number of check nodes (N) in the base graph. Thus, multiple edges in a base graph may be avoided and/or eliminated by introducing one or more degree-two variable nodes (i.e., assuming at least one variable node in the base graph has a degree greater than N).

[0031] Puncturing a high degree base variable node of the LDPC code can also increase the number of check nodes. In addition, high degree check nodes can be desirable in high performing LDPC design. For example, the highest degree variable node may correspond to a variable node having a degree equal to (or one less than) the total number of check nodes in the base graph. Such a high degree variable node can evidently be present in a base graph without any double edges. As described in greater detail below, a punctured variable node is treated as "erased" at decoding. Thus, for codes targeting low error rates, it may be desirable to prevent such nodes from participating in the combinatorial structures (e.g., trapping sets or near codewords) that give rise to error floor events. Having a high degree generally makes it less likely that a node will contribute to an error floor event.

[0032] Furthermore, high degree punctured variable nodes in the graph may improve the performance of the code. It is known that punctured nodes in a graph can improve the so-called iterative threshold of the code structure. In standard irregular LDPC designs (i.e., without punctured variable nodes), thresholds can be improved by increasing the average degree in the bipartite graph, and thus increasing the degrees of the variable and check nodes. With punctured variable nodes, the same effect may be achieved with lower average degree, thereby reducing the complexity of the LDPC code. Furthermore, LDPC code structures having lower average degrees may perform better on smaller graphs. Thus, puncturing high-degree variable nodes may both increase the number of check nodes (thus allowing higher degrees) and improve the performance of codes with limited maximum variable node degrees.

[0033] FIG. **4** is a block diagram of a communications device **400** in accordance with some embodiments. The communications device **400** includes an encoder **410**, a decoder **420**, and a transceiver **430**, which transmits and/or receives LDPC-encoded codewords via a communications channel. The encoder **410** includes a memory **412**, and an LDPC encoder **414**. The memory **412** may be used to store data (i.e., information bits) to be encoded by the LDPC encoder **414**. The LDPC encoder **414** processes the information bits stored in the memory **412** by generating codewords, based on an LDPC code, to be transmitted to another device.

[0034] For some embodiments, the LDPC code may be a lifted LDPC code. Further, for some embodiments, the base LDPC code may include one or more punctured nodes. The LDPC encoder **414** may thus puncture one or more bits of the codeword which correspond with respective punctured nodes of the base LDPC code. These punctured codeword bits are not transmitted by the transceiver **430**. For some embodiments, the punctured nodes may include a base variable node having a degree equal to, or one less than, a number of check

nodes of the LDPC code. For example, at least one of the punctured nodes may be a highest-degree variable node of the LDPC code. For other embodiments, at least one of the punctured nodes may be used to split a check node that is connected to a variable node of the LDPC code by two or more edges. Such a punctured node may be used to eliminate double edges in the base graph for the lifted LDPC code.

[0035] The decoder **420** includes a memory **422** and an LDPC decoder **424**. The memory **422** stores codewords, received via the transceiver **430**, to be decoded by the LDPC decoder **424**. The LDPC decoder **424** processes the codewords stored in the memory **424** by iteratively performing parity check operations, using an LDPC code, and attempting correcting any bits that may have been received in error. For some embodiments, the LDPC code may be a lifted LDPC code. Further, for some embodiments, the received codeword may include one or more puncture bits as determined, for example, based on a set of punctured nodes of the corresponding LDPC code. As described above, with reference to FIG. **3**, the punctured nodes may be determined based on the degrees of the variable nodes of the LDPC code. The LDPC decoder **424** may thus treat these punctured nodes as erased for purposes of decoding. For example, the LDPC decoder **424** may set the log-likelihood ratios (LLRs) of the punctured nodes to zero at initialization.

[0036] For some embodiments, the LDPC decoder **424** may include a plurality of processing elements to perform the parity check or variable node operations in parallel. For example, when processing a codeword with lifting size Z, the LDPC decoder **424** may utilize a number (Z) of processing elements to perform parity check operations on all Z edges of a lifted graph, concurrently. Specifically, each parity check operation may involve reading a corresponding soft bit value from memory **422**, combining the soft bit value with other soft bit values associated with the check node and writing a soft bit back to memory **422** that results from the check node operation. A double edge in a base LDPC code may trigger parallel reading of the same soft bit value memory location twice during a single parallel parity check update. Thus, additional circuitry is typically needed to combine the soft bit values that are written back to memory, so as to properly incorporate both updates. However, eliminating double edges in the LDPC code, for example, as described above with respect to FIG. **3**, helps to avoid this extra complexity.

[0037] FIG. **5** is an illustrative flow chart depicting an LDPC encoding operation **500** in accordance with some embodiments. With reference, for example, to FIG. **4**, the encoder **410** first receives a set of information bits to be encoded (**510**). The information bits may correspond to data intended to be transmitted to another device (e.g., a receiving device) over a communications channel or network. For example, the information bits may be received from a central processing unit (CPU) and stored in memory **412**.

[0038] The encoder **410** may then perform an LDPC encoding operation on the information bits to produce an LDPC codeword (**520**). For some embodiments, the LDPC encoder **414** may encode the information bits into LDPC codewords based on an LDPC code that is shared by the encoder **410** and a corresponding decoder (e.g., of the receiving device). Each codeword may include the original information bits, or a portion thereof, as well as a set of parity bits which may be used (e.g., by the decoder) to perform parity check operations on and/or recover the original information bits.

The encoder **410** may further puncture one or more bits of the LDPC codeword based on base punctured variable nodes of the LDPC code (**530**). For example, the one or more punctured codeword bits may correspond with one or more base variable punctured nodes, respectively, of the base LDPC code. Specifically, at least some of the punctured nodes are provided to eliminate multiple edges between node pairs in the base graph of the lifted LDPC code. For some embodiments, the punctured nodes may include a variable node having a degree equal to, or one less than, a number of check nodes of the LDPC code. For other embodiments, at least one of the punctured nodes may be a degree 2 variable node. For example, the degree 2 variable node may be used to split a check node that would otherwise be connected to another variable node of the LDPC code by two or more edges. For some embodiments both may occur, specifically, both a high degree punctured variable node and a degree two punctured node may occur in the base graph. For some embodiments, the LDPC code may be a lifted LDPC code. Still further, the LDPC code may be based on a quasi-cycling lifting, wherein the permutations of edge clusters are cyclic permutations.

[0039] FIG. **6** is an illustrative flow chart depicting an LDPC decoding operation **600** in accordance with some embodiments. With reference, for example, to FIG. **4**, the decoder **420** first receives an LDPC codeword to be decoded (**610**). The LDPC codeword may be received from a transmitting device, for example, in the form of a quadrature amplitude modulated (QAM) data signal. Accordingly, the LDPC codeword may correspond with a subset of labeling bits of the de-mapped QAM data signal.

[0040] The decoder **420** may identify one or more punctured bits of the LDPC codeword based on base punctured nodes of the LDPC code (**620**). For example, the one or more punctured codeword bits may correspond with one or more base punctured nodes, respectively, of the LDPC code. As described above, at least some of the base punctured nodes are provided to eliminate multiple edges between node pairs in the base graph of the lifted LDPC code. For some embodiments, the punctured nodes may include a variable node having a degree equal to, or one less than, a number of check nodes of the LDPC code. For other embodiments, at least one of the punctured nodes may be a degree 2 variable node. As described above, the degree 2 variable node may be used to split a check node that would otherwise be connected to another variable node of the LDPC code by two or more edges. For some embodiments, the LDPC code may be a lifted LDPC code (e.g., based on a quasi-cyclic lifting).

[0041] The decoder **420** may then perform an LDPC decoding operation on the received codeword to recover the original information bits (**630**). For example, the LDPC decoder **424** may process the codeword by iteratively performing parity check operations, using the LDPC code, and attempting to correct any bits that may have been received in error. For some embodiments, the LDPC decoder **424** may treat the punctured codeword bits as erased during the decoding operation, for example, by setting the LLRs of the punctured nodes to zero at initialization.

[0042] In the present embodiments, each of the LDPC codes may be viewed as a two dimensional binary array of size $Z \times n$, where n is the base (transmission) block length. For some embodiments, the proposed downstream codes are defined such that $Z=360$. In each constellation, k bits may be taken at a time, per dimension (e.g., for 1024QAM, $k=5$). Furthermore, k is a factor of 360, and k bits may be taken at a

time columnwise, thus generating 360/k dimensions or 180/k symbols per column. It should thus be noted that k is a factor of 60 for the set $k \in \{1, 2, 3, 4, 5, 6\}$, in the cases of interest.

[0043] FIGS. **7**, **8**, and **9** show exemplary parity check matrices **700**, **800**, and **900**, respectively, in accordance with some embodiments. In each of the parity check matrices **700**, **800**, and **900**, the top row indexes columns of H. The second row indicates information (1) and parity (0) columns. The third row indicates transmitted columns (1) and punctured columns (0).

[0044] Note that the parity check matrices **700** and **800**, which are associated with LDPC codes with rates r=27/30 and r=13/15, respectively, are systematic. However, the parity check matrix **900**, which is associated with an LDPC code with rate r=21/28, has a punctured information column, and is therefore not fully systematic.

[0045] Further, the parity check matrix **700** has a punctured degree two variable node (index 0). Such a node may split a single parity check into two. This ensures that the base matrix has no double edges, and facilitates some of the embodiments described herein. An equivalent code representation can be constructed by merging the two parity checks and eliminating the punctured degree 2 variable node. Moreover, such an equivalent representation shall have double or multiple edges in the base graph.

[0046] FIG. **10** is a block diagram of a communications device **1000** in accordance with some embodiments. The communications device **1000** includes a transceiver **1010**, a processor **1020**, and memory **1030**. The transceiver **1010** may be used for communicating data to and/or from the communications device **1000**. For example, the transceiver **1010** may receive and/or transmit information bits between the communications device **1000** and a CPU. The encoder interface **1010** may also output and/or receive LDPC codewords between the communications device **1000** and another communications device in a network.

[0047] Memory **1030** may include a data store **1032** that may be used as a local cache to store the received information bits and/or codewords. Furthermore, memory **1030** may also include a non-transitory computer-readable storage medium (e.g., one or more nonvolatile memory elements, such as EPROM, EEPROM, Flash memory, a hard drive, etc.) that can store the following software modules:

- **[0048]** an LDPC encoding module **1034** to encode a set of information bits, using an LDPC code, to produce a codeword;
- **[0049]** an LDPC decoding module **1036** to decode LDPC codewords using an LDPC code.

Each software module may include instructions that, when executed by the processor **1020**, may cause the encoder **1000** to perform the corresponding function. Thus, the non-transitory computer-readable storage medium of memory **1030** may include instructions for performing all or a portion of the operations described above with respect to FIGS. **5-6**. It should be noted that, while the modules **1034-1036** are depicted as software in memory **1030**, any of the module may be implemented in hardware, software, firmware, or a combination of the foregoing.

[0050] The processor **1020**, which is coupled between the encoder interface **1010** and the memory **1030**, may be any suitable processor capable of executing scripts of instructions of one or more software programs stored in the decoder **1000** (e.g., within memory **1030**). For example, the processor **1020** may execute the LDPC encoding module **1034** and/or the LDPC decoding module **1036**.

[0051] The LDPC encoding module **1034** may be executed by the processor **1020** to encode the information bits, using the LDPC code, to produce a codeword. For example, the processor **1020**, in executing the LDPC encoding module **1034**, may perform an LDPC encoding operation on the information bits based on an LDPC code that is shared by the LDPC encoding module **1034** and a decoding module of a corresponding receive device. Each codeword may include the original information bits as well as a set of parity bits which may be used to perform parity checks on and/or recover the original information bits. For some embodiments, the LDPC code may be a lifted LDPC code (e.g., based on a quasi-cyclic lifting).

[0052] The processor **1020**, in executing the LDPC encoding module **1034**, may further puncture one or more bits of the codeword based on the corresponding LDPC code. For example, the one or more punctured codeword bits may correspond with one or more punctured nodes, respectively, of the LDPC code. As described above, at least some of the punctured nodes are provided to eliminate multiple edges between node pairs in the base graph for the lifted LDPC code. For some embodiments, the punctured nodes may include a variable node having a degree equal to, or one less than, a number of check nodes of the LDPC code. For other embodiments, at least one of the punctured nodes may be a degree 2 variable node (e.g., used to split a check node that would otherwise be connected to another variable node of the LDPC code by two or more edges).

[0053] The LDPC decoding module **1036** may be executed by the processor **1020** to decode LDPC codewords using the LDPC code. For some embodiments, the processor **1020**, in executing the LDPC decoding module **1036**, may first identify one or more punctured bits of the received codeword based on the LDPC code. The processor **1020** may then perform an LDPC decoding operation on the received codeword, while treating the punctured codeword bits as erased. For example, the LDPC decoding module **1036**, as executed by the processor **1020**, may set the LLRs of the punctured nodes to zero at initialization. For some embodiments, the LDPC code may be a lifted LDPC code (e.g., based on a quasi-cyclic lifting).

[0054] As described above, the punctured codeword bits may correspond with respective punctured nodes of the LDPC code, wherein at least some of the punctured nodes are provided to eliminate multiple edges between node pairs in the base graph for the lifted LDPC code. For some embodiments, the punctured nodes may include a variable node having a degree equal to, or one less than, a number of check nodes of the LDPC code. For other embodiments, at least one of the punctured nodes may be a degree 2 variable node (e.g., used to split a check node that would otherwise be connected to another variable node of the LDPC code by two or more edges).

[0055] In the foregoing specification, the present embodiments have been described with reference to specific exemplary embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader scope of the disclosure as set forth in the appended claims. The specification and drawings are, accordingly, to be regarded in an illustrative sense rather than a restrictive sense. For example, the method steps depicted in the flow charts of FIGS. **5-6** may be performed in

other suitable orders, multiple steps may be combined into a single step, and/or some steps may be omitted.

What is claimed is:

1. A method of data encoding, the method comprising:

receiving a set of information bits;

performing a lifted low density parity check (LDPC) encoding operation on the set of information bits to produce a codeword; and

puncturing all lifted bits of the codeword that correspond to one or more punctured base bits of a base LDPC code used for the LDPC encoding operation, wherein:

the one or more punctured base bits are those that correspond with one or more punctured base nodes, respectively, of the base LDPC code; and

the base LDPC code has no multiple edges.

2. The method of claim **1**, wherein the one or more punctured base nodes correspond to one or more variable nodes having a degree equal to, or one less than, a number of check nodes of the base LDPC code.

3. The method of claim **1**, wherein at least one of the one or more punctured base nodes corresponds to a highest-degree variable node of the base LDPC code.

4. The method of claim **1**, wherein the one or more punctured base nodes correspond to one or more degree 2 variable nodes.

5. The method of claim **4**, wherein the one or more punctured base nodes split one or more respective check nodes that are each connected to another variable node, and wherein each of the other variable nodes is connected by edges to both elements of the corresponding split check node.

6. The method of claim **4**, wherein the one or more punctured base nodes eliminate double edges in the base LDPC code.

7. The method of claim **1**, wherein a quasi-cyclic lifting is applied to the base LDPC code, and wherein permutations of edge clusters in the quasi-cyclic lifting are cyclic permutations.

8. A method of data decoding, the method comprising:

receiving an LDPC codeword;

identifying all lifted bits of the LDPC codeword that correspond to one or more punctured based bits of a base LDPC code, wherein:

the one or more punctured base bits are those that correspond with one or more punctured base nodes, respectively, of the base LDPC code; and

the base LDPC code has no multiple edges; and

performing an LDPC decoding operation on the received codeword to recover a set of information bits, wherein the identified lifted bits are treated as erased for purposes of decoding.

9. The method of claim **8**, wherein the one or more punctured base nodes correspond to one or more variable nodes having a degree equal to, or one less than, a number of check nodes of the base LDPC code.

10. The method of claim **8**, wherein at least one of the one or more punctured base nodes corresponds to a highest-degree variable node of the base LDPC code.

11. The method of claim **8**, wherein the one or more punctured base nodes correspond to one or more degree 2 variable nodes.

12. The method of claim **11**, wherein the one or more punctured base nodes split one or more respective check nodes that are each connected to another variable node, and wherein each of the other variable nodes is connected by edges to both elements of the corresponding split check node.

13. The method of claim **11**, wherein the one or more punctured base nodes eliminate double edges in the base LDPC code.

14. The method of claim **8**, wherein a quasi-cyclic lifting is applied to the base LDPC code, and wherein permutations of edge clusters in the quasi-cyclic lifting are cyclic permutations.

15. A computer-readable storage medium containing program instructions that, when executed by a processor provided within a communications device, causes the device to:

receive a set of information bits;

perform a lifted LDPC encoding operation on the set of information bits to produce a codeword; and

puncture all lifted bits of the codeword that correspond to one or more punctured base bits of a base LDPC code used for the LDPC encoding operation, wherein:

the one or more punctured base bits are those that correspond with one or more punctured base nodes, respectively, of the base LDPC code; and

the base LDPC code has no multiple edges.

16. The computer-readable storage medium of claim **15**, wherein the one or more punctured base nodes correspond to one or more variable nodes having a degree equal to, or one less than, a number of check nodes of the base LDPC code.

17. The computer-readable storage medium of claim **15**, wherein at least one of the one or more punctured base nodes corresponds to a highest-degree variable node of the base LDPC code.

18. The computer-readable storage medium of claim **15**, wherein the one or more punctured base nodes correspond to one or more degree 2 variable nodes.

19. The computer-readable storage medium of claim **18**, wherein the one or more punctured base nodes split one or more respective check nodes that are each connected to another variable node, and wherein each of the other variable nodes is connected by edges to both elements of the corresponding split check node.

20. The computer-readable storage medium of claim **18**, wherein the one or more punctured base nodes eliminate double edges in the base LDPC code.

21. The computer-readable storage medium of claim **15**, wherein a quasi-cyclic lifting is applied to the base LDPC code, and wherein permutations of edge clusters in the quasi-cyclic lifting are cyclic permutations.

22. A computer-readable storage medium containing program instructions that, when executed by a processor provided within a communications device, causes the device to:

receive an LDPC codeword;

identify all lifted bits of the LDPC codeword that correspond to one or more punctured based bits of a base LDPC code, wherein:

the one or more punctured base bits are those that correspond with one or more punctured base nodes, respectively, of the base LDPC code; and

the base LDPC code has no multiple edges; and

perform an LDPC decoding operation on the received codeword to recover a set of information bits, wherein the identified lifted bits are treated as erased for purposes of decoding.

23. The computer-readable storage medium of claim **22**, wherein the one or more punctured base nodes correspond to

one or more variable nodes having a degree equal to, or one less than, a number of check nodes of the base LDPC code.

24. The computer-readable storage medium of claim **22**, wherein at least one of the one or more punctured base nodes corresponds to a highest-degree variable node of the base LDPC code.

25. The computer-readable storage medium of claim **22**, wherein the one or more punctured base nodes correspond to one or more degree 2 variable nodes.

26. The computer-readable storage medium of claim **25**, wherein the one or more punctured base nodes split one or more respective check nodes that are each connected to another variable node, and wherein each of the other variable nodes is connected by edges to both elements of the corresponding split check node.

27. The computer-readable storage medium of claim **25**, wherein the one or more punctured base nodes eliminate double edges in the base LDPC code.

28. The computer-readable storage medium of claim **22**, wherein a quasi-cyclic lifting is applied to the base LDPC code, and wherein permutations of edge clusters in the quasi-cyclic lifting are cyclic permutations.

29. A communications device, comprising:

a memory to store a set of information bits; and

an encoder to:

perform a lifted LDPC encoding operation on the set of information bits to produce a codeword; and

puncture all lifted bits of the codeword that correspond to one or more punctured base bits of a base LDPC code used for the LDPC encoding operation, wherein:

the one or more punctured base bits are those that correspond with one or more punctured base nodes, respectively, of the base LDPC code; and

the base LDPC code has no multiple edges.

30. The device of claim **29**, wherein the one or more punctured base nodes correspond to one or more variable nodes having a degree equal to, or one less than, a number of check nodes of the base LDPC code.

31. The device of claim **29**, wherein at least one of the one or more punctured base nodes corresponds to a highest-degree variable node of the base LDPC code.

32. The device of claim **29**, wherein the one or more punctured base nodes correspond to one or more degree 2 variable nodes.

33. The device of claim **32**, wherein the one or more punctured base nodes split one or more respective check nodes that are each connected to another variable node, and wherein each of the other variable nodes is connected by edges to both elements of the corresponding split check node.

34. The device of claim **32**, wherein the one or more punctured base nodes eliminate double edges in the base LDPC code.

35. The device of claim **29**, wherein a quasi-cyclic lifting is applied to the base LDPC code, and wherein permutations of edge clusters in the quasi-cyclic lifting are cyclic permutations.

36. A communications device, comprising:

a memory to store an LDPC codeword; and

a decoder to:

identify all lifted bits of the LDPC codeword that correspond to one or more punctured based bits of a base LDPC code, wherein:

the one or more punctured base bits are those that correspond with one or more punctured base nodes, respectively, of the base LDPC code; and

the base LDPC code has no multiple edges; and

perform an LDPC decoding operation on the received codeword to recover a set of information bits, wherein the identified lifted bits are treated as erased for purposes of decoding.

37. The device of claim **36**, wherein the one or more punctured base nodes correspond to one or more variable nodes having a degree equal to, or one less than, a number of check nodes of the base LDPC code.

38. The device of claim **36**, wherein at least one of the one or more punctured base nodes corresponds to a highest-degree variable node of the base LDPC code.

39. The device of claim **36**, wherein the one or more punctured base nodes correspond to one or more degree 2 variable nodes.

40. The device of claim **39**, wherein the one or more punctured base nodes split one or more respective check nodes that are each connected to another variable node, and wherein each of the other variable nodes is connected by edges to both elements of the corresponding split check node.

41. The device of claim **39**, wherein the one or more punctured base nodes eliminate double edges in the base LDPC code.

42. The device of claim **36**, wherein a quasi-cyclic lifting is applied to the base LDPC code, and wherein permutations of edge clusters in the quasi-cyclic lifting are cyclic permutations.

43. An encoder, comprising:

means for receiving a set of information bits;

means for performing a LDPC encoding operation on the set of information bits to produce a codeword; and

means for puncturing all lifted bits of the codeword that correspond to one or more punctured base bits of a base LDPC code used for the LDPC encoding operation, wherein:

the one or more punctured base bits are those that correspond with one or more punctured base nodes, respectively, of the base LDPC code; and

the base LDPC code has no multiple edges.

44. A decoder, comprising:

means receiving an LDPC codeword;

means for identifying all lifted bits of the LDPC codeword that correspond to one or more punctured based bits of a base LDPC code, wherein:

the one or more punctured base bits are those that correspond with one or more punctured base nodes, respectively, of the base LDPC code; and

the base LDPC code has no multiple edges; and

means for performing an LDPC decoding operation on the received codeword to recover a set of information bits, wherein the identified lifted bits are treated as erased for purposes of decoding.

* * * * *