(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
2 June 2016 (02.06.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/086045 A1

(51) International Patent Classification:
H04L 1/00 (2006.01) *H04L 25/14* (2006.01)
H03M 13/03 (2006.01) *H04L 25/493* (2006.01)
H04L 7/033 (2006.01) *H04L 1/24* (2006.01)
H04L 25/02 (2006.01)

(21) International Application Number:
PCT/US2015/062511

(22) International Filing Date:
24 November 2015 (24.11.2015)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/084,998 26 November 2014 (26.11.2014) US
62/216,692 10 September 2015 (10.09.2015) US
62/236,522 2 October 2015 (02.10.2015) US
14/949,290 23 November 2015 (23.11.2015) US

(71) Applicant: QUALCOMM INCORPORATED [US/US]; ATTN: International IP Administration, 5775 Morehouse Drive, San Diego, California 92121-1714 (US).

(72) Inventor: SENGOKU, Shoichiro; 5775 Morehouse Drive, San Diego, California 92121-1714 (US).

(74) Agent: SMYTH, Anthony; Loza & Loza, LLP, 305 North Second Avenue #127, Upland, California 91786 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

(54) **Title**: MULTI-WIRE SYMBOL TRANSITION CLOCKING SYMBOL ERROR CORRECTION

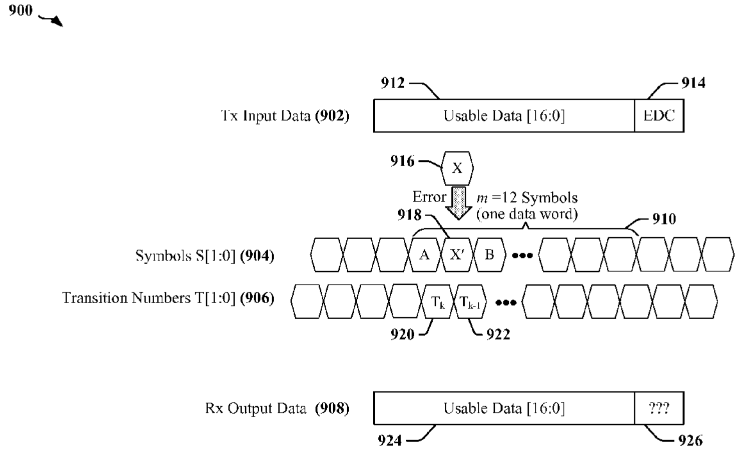


FIG. 9

(57) **Abstract**: Apparatus, systems and methods for error detection in transmissions on a multi-wire interface are disclosed. A method for correcting transmission errors in multi-wire transition-encoded interface may include determining whether a symbol error is present in the sequence of symbols based on a value of an error detection code (EDC) in the received plurality of bits, generating one or more permutations of the sequence of symbols, where each permutation includes one symbol that is different from corresponding symbols in the sequence of symbols and different from corresponding symbols in other permutations. A permutation in the one or more permutations may be identified as including a corrected sequence of symbols when it produces a decoded EDC value that matches an expected EDC value. The expected EDC value may correspond to a predefined value for EDCs transmitted over the multi-wire interface to enable detection of up to two symbol errors at the receiver.

MULTI-WIRE SYMBOL TRANSITION CLOCKING SYMBOL ERROR CORRECTION

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority to and the benefit of U.S. Provisional Application No. 62/236,522 filed in the U.S. Patent and Trademark Office on October 2, 2015, U.S. Provisional Application No. 62/216,692 filed in the U.S. Patent and Trademark Office on September 10, 2015, U.S. Provisional Application No. 62/084,998 filed in the U.S. Patent and Trademark Office on November 26, 2014, and U.S. Non-Provisional Application No. 14/949,290 filed in the U.S. Patent and Trademark Office on November 23, 2015, the entire contents of which are being incorporated herein by reference.

BACKGROUND

Field

[0002] The present disclosure pertains to enabling efficient operations over data communication interfaces and, more particularly, facilitating error correction over data communication interfaces that employ symbol transition clocking transcoding.

Background

[0003] Data communication interfaces may employ symbol transition clocking transcoding to embed clock information in sequences of symbols that encode data to be transmitted over an interface that has multiple signal wires, thereby obviating the need for dedicated clock signal wires.

[0004] In certain examples of multi-signal data transfer, multi-wire differential signaling such as N-factorial (*N*!) low-voltage differential signaling (LVDS), transcoding (e.g., the digital-to-digital data conversion of one encoding type to another) may be performed to embed symbol clock information by causing symbol transition at every symbol cycle, instead of sending clock information in separate data lanes (differential transmission paths). Embedding clock information by such transcoding can also minimize skew between clock and data signals, as well as to eliminate the need for a phase-locked loop (PLL) to recover the clock information from the data signals. In one example, a two-wire serial bus operated in accordance with conventional Inter-Integrated Circuit (I2C) protocols or camera control interface (CCI) protocols can be adapted to provide a CCI extension (CCIe) bus, which uses symbol transition clocking transcoding. The CCIe bus supports support a higher bit rate than an I2C bus or a CCI bus.

[0005] Error detection and correction can be problematic in data transfer interfaces that employ transition encoding because there is typically no direct association between a signaling state error and errors in data decoded from the data transfer interface. The disassociation between data bits and signaling state can render conventional error detection techniques ineffective when applied to transition encoding interfaces.

[0006] It would be desirable to provide reliable error detection and correction in transmissions between devices that use symbol transition clocking transcoding to communicate.

SUMMARY

[0007] Various aspects of the disclosure relate to a method, a computer program, and an apparatus that improve functionality and operational efficiency of devices that can reliably detect errors that affect two or more symbols transmitted over a multi-wire transition-encoded interface, and that can correct a single symbol error for each sequence of symbols that encodes a single data word.

[0008] According to certain aspects, a method of correcting transmission errors at a receiver in a multi-wire interface includes receiving a sequence of symbols representative of the signaling state of a plurality of connectors, and decoding the sequence of symbols. The sequence of symbols may be decoded by converting the sequence of symbols into a transition number, and converting the transition number into a received plurality of bits. Each digit of the transition number may represent a transition between two consecutive symbols transmitted on the plurality of connectors. The method of correcting transmission errors may include determining whether a symbol error is present in the sequence of symbols based on a value of an error detection code (EDC) in the received plurality of bits. When the symbol error is determined to be present in the sequence of symbols, one or more permutations of the sequence of symbols may be generated, where each permutation includes one symbol that is different from corresponding symbols in the sequence of symbols and different from corresponding symbols in other permutations. The one or more permutations may be decoded and a permutation may be identified as a corrected

sequence of symbols when it produces a decoded EDC value that matches an expected EDC value. The expected EDC value may correspond to a predefined value for EDCs transmitted over the multi-wire interface to enable detection of up to two symbol errors at the receiver.

[0009] According to certain aspects, an apparatus includes means for receiving a sequence of symbols representative of the signaling state of a plurality of connectors, and means for decoding the sequence of symbols. The means for decoding the sequence of symbols may be configured to convert the sequence of symbols into a transition number, each digit of the transition number representing a transition between two consecutive symbols transmitted on the plurality of connectors, and convert the transition number into a received plurality of bits. The apparatus may include means for determining whether a symbol error is present in the sequence of symbols based on a value of an EDC in the received plurality of bits, and means for correcting symbol errors operative when the symbol error is determined to be present in the sequence of symbols. The means for correcting symbol errors may be configured to generate one or more permutations of the sequence of symbols, each permutation including one symbol that is different from corresponding symbols in the sequence of symbols and different from corresponding symbols in other permutations, decode the one or more permutations, and identify a permutation as a corrected sequence of symbols when it produces a decoded EDC value that matches an expected EDC value. The expected EDC value may correspond to a predefined value for EDCs transmitted over the multi-wire interface to enable detection of up to two symbol errors at the receiver.

[0010] According to certain aspects, an apparatus includes a communications transceiver configured to provide a sequence of symbols representative of the signaling state of a plurality of connectors, a decoder circuit configured to convert a transition number representative of transitions between consecutive symbols in the sequence of symbols into a plurality of bits, an error detection circuit configured to determine whether symbol errors are present in the sequence of symbols based on a value of an EDC in the received plurality of bits, and an error correction circuit. The error correction circuit may be responsive to a determination that a symbol error is present in the sequence of symbols and may be configured to generate one or more permutations of the sequence of symbols, each permutation including one symbol that is different from corresponding symbols in the sequence of symbols and different from corresponding symbols in other permutations, decode the one or more permutations, and identify a permutation as a corrected sequence of symbols when it produces a decoded EDC value that matches an expected EDC value. The expected EDC value may correspond to a predefined value for EDCs transmitted over the multi-wire interface to enable detection of up to two symbol errors at the receiver.

[0011] According to certain aspects, a processor readable storage medium may have instructions stored thereon. The instructions may be executable by the processor. The instructions may include instructions for receiving a sequence of symbols representative of the signaling state of a plurality of connectors, and decoding the sequence of symbols. In one example, the instructions may include instructions for converting the sequence of symbols into a transition number, each digit of the transition number representing a transition between two consecutive symbols transmitted on the plurality of connectors, and converting the transition number into a received plurality of bits. The instructions may include instructions for determining whether a symbol error is present in the sequence of symbols based on a value of an EDC in the received plurality of bits, and when the symbol error is determined to be present in the sequence of symbols generating one or more permutations of the sequence of symbols, each permutation including one symbol that is different from corresponding symbols in the sequence of symbols and different from corresponding symbols in other permutations, decoding the one or more permutations, and identifying a permutation as a corrected sequence of symbols when it produces a decoded EDC value that matches an expected EDC value. The expected EDC value may correspond to a predefined value for EDCs transmitted over the multi-wire interface to enable detection of up to two symbol errors at the receiver.

DRAWINGS

[0012] Various features, nature, and advantages may become apparent from the detailed description set forth below when taken in conjunction with the drawings in which like reference characters identify correspondingly throughout.

[0013] FIG. 1 depicts an apparatus employing a data link between IC devices that selectively operates according to one of a plurality of available standards.

[0014] FIG. 2 illustrates a system architecture for an apparatus employing a data link between IC devices.

[0015] FIG. 3 illustrates an example of an *N*! interface provided between two devices.

[0016] FIG. 4 illustrates a transmitter and a receiver that may be adapted according to certain aspects disclosed herein.

[0017] FIG. 5 illustrates an encoding scheme that may be used to control conversions between transition numbers and sequential symbols.

[0018] FIG. 6 illustrates the relationship between symbols and transition numbers in one example of a transition-encoding interface.

[0019] FIG. 7 illustrates possible transition number-to-symbol encoding at a symbol boundary in a 3! interface.

[0020] FIG. 8 illustrates a mathematical relationship between transition numbers and symbols in a 3! interface.

[0021] FIG. 9 illustrates an example in which a sequence of symbols transmitted over a multi-wire communication interface is affected by a single symbol error.

[0022] FIG. 10 is a diagram that illustrates a mathematical relationship characterizing a single symbol error in a sequence of symbols transmitted over a multi-wire communication interface.

[0023] FIG. 11 tabulates values of r^n, where n lies in the range 0-15, and when r=3 and r=5.

[0024] FIG. 12 tabulates error coefficients corresponding to a single symbol error in a sequence of symbols.

[0025] FIG. 13 illustrates the longest non-zero LSB portion in an error coefficient.

[0026] FIG. 14 illustrates cases in which a single symbol error results in an error in a single transition number.

[0027] FIG. 15 illustrates a first example of signaling errors affecting two symbols in a sequence of symbols transmitted over a multi-wire communication interface.

[0028] FIG. 16 illustrates a second example of signaling errors that affect two consecutive symbols transmitted over a multi-wire communication interface.

[0029] FIG. 17 illustrates the number of bits provided in an EDC for detection of two symbol errors in accordance with certain aspects disclosed herein.

[0030] FIG. 18 illustrates the operation of a circuit that may be used to correct a single symbol error in accordance with certain aspects disclosed herein.

[0031] FIG. 19 illustrates a first configuration of decoders that may be used to correct symbol errors in accordance with certain aspects disclosed herein.

[0032] FIG. 20 illustrates a second configuration of decoders that may be used to correct symbol errors in accordance with certain aspects disclosed herein.

[0033] FIG. 21 illustrates certain aspects of decoders used to correct symbol errors that may exploited to reduce decoding overhead in accordance with certain aspects disclosed herein.

[0034] FIG. 22 is a block diagram illustrating an example of an apparatus employing a processing system that may be adapted according to certain aspects disclosed herein.

[0035] FIG. 23 is a flow chart of a method for correcting a symbol error in accordance with certain aspects disclosed herein.

[0036] FIG. 24 is a flow chart illustrating an example of a method for decoding symbols in accordance with certain aspects disclosed herein.

[0037] FIG. 25 is a diagram illustrating a second example of a hardware implementation for an apparatus used in an interface that provides symbol error detection according to certain aspects disclosed herein.

DETAILED DESCRIPTION

[0038] In the following description, specific details are given to provide a thorough understanding of the embodiments. However, it will be understood by one of ordinary skill in the art that the embodiments may be practiced without these specific detail. For example, circuits may be shown in block diagrams in order not to obscure the embodiments in unnecessary detail. In other instances, well-known circuits, structures, and techniques may not be shown in detail in order not to obscure the embodiments.

Overview

[0039] Certain data transfer interfaces employ transition encoding, including 3-phase and N! multi-wire LVDS interfaces, and multi-wire single-ended interfaces including the CCIe interface. Transition encoding embeds clock information in signaling states transmitted over the interface. In certain instances, data is transcoded to transition numbers, where each transition number selects a next symbol to be transmitted after a current symbol. Each symbol may represent signaling state of the interface. For example, the transition number may represent an offset used to select between symbols in an ordered set of symbols that can be transmitted on the interface. By ensuring that consecutive symbols are different from one another, a change in signaling state of the interface occurs at each symbol boundary providing information used to generate a receive clock at the receiver.

[0040] Errors in signaling state that change a transmitted symbol S_1 to a received symbol Se_1 can cause a receiver to produce an incorrect transition number T_1+e_1 associated with the transition between an immediately preceding symbol S_2 and the changed symbol Se_1. T_1 represents the difference between S_2 and the correctly transmitted symbol S_1, and e_1 is the value of an offset introduced by the signaling error. A second incorrect transition number T_0+e_0 is associated with the changed symbol Se_1, where T_0 represents the difference between the correctly transmitted symbol S_1 and a next symbol S_0, with e_0 representing the value of the offset introduced by the signaling error. The values of e_1 and e_0 do not directly correspond to the error in signaling state, and the disassociation between data bit errors and signaling state errors can render conventional error detection techniques ineffective when applied to transition encoding interfaces.

[0041] According to certain aspects disclosed herein, reliable error detection is enabled in transition-encoded interfaces by providing an error detection constant (EDC) that can detect two symbol errors in a sequence of symbols that is used to transmit a word of data. The EDC may include a predefined number of bits having a known, fixed value. The value of the EDC may have a zero value, in one example, and may be provided as a predefined or preselected number of least significant bits (LSBs) appended to each word to be transmitted on the multi-wire interface.

[0042] According to certain aspects, a single symbol error per transmitted data word may be corrected. In one example, an EDC appended to a data word may be capable of detecting two symbol errors, and this capability may be leveraged to implement an error correction scheme that is effective for correcting a single symbol error. In operation, a receiver may monitor the EDC value in decoded words and determine that at least one symbol error has occurred when, for example, the EDC has an unexpected value. Error correction may be attempted when an error is detected.

[0043] An error correction scheme may operate by testing different changes in the value of each symbol in the received sequence of symbols corresponding to the received data word that is associated with an EDC that does not match the predefined value inserted by the transmitter. For example, permutations of the sequence of symbols are decoded with alternative values of one symbol while the other symbols are unmodified. Error correction may be accomplished when a modified symbol causes the EDC appended to the resultant decoded data word to match the predefined value expected for an EDC after error-free transmission. Since the EDC detects two symbol errors, any modifications made by error correction logic to a correctly-received symbol introduces a second symbol error into the sequence of symbols, that may be detected by comparing EDC values.

Example Of A Device Employing Transition Encoding

[0044] FIG. 1 depicts one example of an apparatus 100 that employs a communication link between IC devices. In one example, the apparatus 100 may include a wireless communication device that communicates through an RF transceiver with a radio access network (RAN), a core access network, the Internet and/or another network. The apparatus 100 may include a communications transceiver 106 operably coupled to processing circuit 102. The processing circuit 102 may include one or more IC devices, such as an application-specific IC (ASIC) 108. The ASIC 108 may include one or more processing devices, logic circuits, and so on. The processing circuit 102 may include and/or be coupled to processor readable storage such as a memory device 112 that may maintain instructions and data that may be executed by processing circuit 102. The processing circuit 102 may be controlled by one or more of an operating system and an application programming interface (API) 110 layer that supports and enables execution of software modules residing in storage media, such as the memory device 112 of the wireless device. The memory device 112 may include read-only memory (ROM) or random-access memory (RAM), electrically erasable programmable ROM (EEPROM), flash cards, or any memory device that can be used in processing systems and computing platforms. The processing circuit 102 may include or access a local database 114 that can maintain operational parameters and other information used to configure and operate the apparatus 100. The local database 114 may be implemented using one or more of a database module, flash memory, magnetic media, EEPROM, optical media, tape, soft or hard disk, or the like. The processing circuit may also be operably coupled to external devices such as an antenna 122, a display 124, operator controls, such as button 128 and keypad 126 among other components.

[0045] FIG. 2 is a block schematic drawing illustrating certain aspects of an apparatus 200 connected to a communication link 220, where the apparatus 200 may be embodied in one or more of a wireless mobile device, a mobile telephone, a mobile computing system, a wireless telephone, a notebook computer, a tablet computing device, a media player, s gaming device, or the like. The apparatus 200 may include a plurality of IC devices 202 and 230 that exchange data and control information through a communication link 220. The communication link 220 may be used to connect IC devices 202 and 230 that are located in close proximity to one another, or physically located in different parts of the apparatus 200. In one example, the communication link 220 may be provided on a chip carrier, substrate or circuit board that carries the IC devices 202 and 230. In another example, a first IC device 202 may be located in a keypad section of a flip-phone while a second IC device 230 may be located in a display section of the flip-phone. In another example, a portion of the communication link 220 may include a cable or optical connection.

[0046] The communication link 220 may include multiple channels 222, 224 and 226. One or more channels 226 may be bidirectional, and may operate in half-duplex and/or full-duplex modes. One or more channels 222 and 224 may be unidirectional. The communication link 220 may be asymmetrical, providing higher bandwidth in one direction. In one example described herein, a first communication channel 222 may provide or be referred to as a forward link while a second communication channel 224 may provide or be referred to as a reverse link. The first IC device 202 may be designated as a host system or transmitter, while the second IC device 230 may be designated as a client system or receiver, even if both IC devices 202 and 230 are configured to transmit and receive on the communication channel 222. In one example, a forward link may operate at a higher data rate when communicating data from a first IC device 202 to a second IC device 230, while a reverse link may operate at a lower data rate when communicating data from the second IC device 230 to the first IC device 202.

[0047] The IC devices 202 and 230 may each have a processor 206, 236 and/or a processing and/or computing circuit or other such device or circuit. In one example, the first IC device 202 may perform core functions of the apparatus 200, including maintaining wireless communications through a wireless transceiver 204 and an antenna 214, while the second IC device 230 may support a user interface that manages or operates a display controller 232. The first IC device 202 or second IC device 230 may control operations of a camera or video input device using a camera controller 234. Other features supported by one or more of the IC devices 202 and 230 may include a keyboard, a voice-recognition component, and other input or output devices. The display controller 232 may include circuits and software drivers that support displays such as a liquid crystal display (LCD) panel, touch-screen display, indicators and so on. The storage media 208 and 238 may include transitory and/or non-transitory storage devices adapted to maintain instructions and data used by respective processors 206 and 236, and/or other components of the IC devices 202 and 230. Communication between each processor 206, 236 and its corresponding storage media 208 and 238 and other modules and circuits may be facilitated by one or more bus 212 and 242, respectively.

[0048] The reverse link (here, the second communication channel 224) may be operated in the same manner as the forward link (here, the first communication channel 222), and the first communication channel 222 and second communication channel 224 may be capable of transmitting at comparable speeds or at different speeds, where speed may be expressed as data transfer rate and/or clocking rates. The forward and reverse data rates may be substantially the same or differ by orders of magnitude, depending on the application. In some applications, a single bidirectional link (here, the third communication channel 226) may support communications between the first IC device 202 and the second IC device 230. The first communication channel 222 and/or second communication channel 224 may be configurable to operate in a bidirectional mode when, for example, the forward and reverse links share the same physical connections and operate in a half-duplex manner. In one example, the communication link 220 may be operated to communicate control, command and other information between the first IC device 202 and the second IC device 230 in accordance with an industry or other standard.

[0049] In one example, forward and reverse links may be configured or adapted to support a wide video graphics array (WVGA) 80 frames per second LCD driver IC without a frame buffer, delivering pixel data at 810 Mbps for display refresh. In another example, forward and reverse links may be configured or adapted to enable communications between with dynamic random access memory (DRAM), such as double data rate synchronous dynamic random access memory (SDRAM). Encoding devices 210 and/or 230 can encode multiple bits per clock transition, and multiple sets of wires can be used to transmit and receive data from the SDRAM, control signals, address signals, and so on.

[0050] Forward and reverse channels may comply or be compatible with application-specific industry standards. In one example, the MIPI standard defines physical layer interfaces between an IC device 202 that includes an application processor and an IC device 230 that controls and/or supports the camera or display in a mobile device. The MIPI standard includes specifications that govern the operational characteristics of products that comply with MIPI specifications for mobile devices. The MIPI standard may define interfaces that employ complimentary metal-oxide-semiconductor (CMOS) parallel busses.

[0051] In one example, the communication link 220 of FIG. 2 may be implemented as a wired bus that includes a plurality of signal wires (denoted as *N* wires). The *N* wires may be configured to carry data encoded in symbols, where each symbol defines a signaling state of the *N* wires, and where clock information is embedded in a sequence of the symbols transmitted over the plurality of wires.

[0052] FIG. 3 is a diagram illustrating one example of an *N*-wire interface 300 provided between two devices. At a transmitter 302, a transcoder 306 may be used to encode data bits 304 and clock information in symbols to be transmitted over a set of *N* wires 314 using *N*-factorial (*N!*) encoding. The clock information is derived from a transmit clock 312 and may be encoded in a sequence of symbols transmitted in $_NC_2$ differential signals over the *N* wires 314 by ensuring that a signaling state transition occurs on at least one of the $_NC_2$ signals between consecutive symbols. When *N!* encoding is used to drive the *N* wires 314, each bit of a symbol is transmitted as a differential signal by one of a set of differential line drivers 310, where the differential drivers in the set of differential line drivers 310 are coupled to different pairs of the *N* wires. The number of available combinations of wire pairs ($_NC_2$) determines the number of signals that can be transmitted over the *N* wires 314. The number of data bits 304 that can be encoded in a symbol may be calculated based on the number of available signaling states available for each symbol transmission interval.

[0053] A termination impedance (typically resistive) couples each of the *N* wires 314 to a common center point 318 in a termination network 316. It will be appreciated that the signaling states of the *N* wires 314 reflects a combination of the currents in the termination network 316 attributed to the differential line drivers 310 coupled to each wire. It will be further appreciated that the center point 318 is a null point, whereby the currents in the termination network 316 cancel each other at the center point.

[0054] The *N!* encoding scheme need not use a separate clock channel and/or non-return-to-zero decoding because at least one of the $_NC_2$ signals in the link transitions between consecutive symbols. Effectively, the transcoder 306 ensures that a transition occurs between each pair of symbols transmitted on the *N* wires 314 by producing a sequence of symbols in which each symbol is different from its immediate predecessor symbol. In the example depicted in FIG. 3, *N*=4 wires are provided, and the 4 wires can carry ${}_4C_2 = 6$ differential signals. The transcoder 306 may employ a mapping scheme to generate raw symbols for transmission on the *N* wires 314. The transcoder 306 may map data bits 304 to a set of transition numbers. The transition numbers may then be used to select a raw symbol for transmission based on the value of the preceding symbol such that the selected raw symbol is different from the preceding raw symbol. In one example, a transition number may be used to lookup a data value corresponding to the second of the consecutive raw symbols with reference to the first of the consecutive raw symbols. At the receiver 320, a transcoder 328 may employ a mapping to determine a transition number that characterizes a difference between a pair of consecutive raw symbols in a lookup table, for example. The transcoders 306, 328 operate on the basis that every consecutive pair of raw symbols includes two different symbols.

[0055] The transcoder 306 at the transmitter 302 may select between the *N*! - 1 symbols that are available at every symbol transition. In one example, a 4! system provides 4! - 1 = 23 signaling states for the next symbol to be transmitted at each symbol transition. The bit rate may be calculated as $\log_2$(*available_ states*) per transmit clock cycle. In a system using double data rate (DDR) clocking, whereby symbol transitions occur at both the rising edge and falling edge of the transmit clock 312, two symbols are transmitted per transmit clock cycle. The total available states in the transmit clock cycle for *N*=4 is $(N!-1)^2 = (23)^2 = 529$ and the number of data bits 304 that can transmitted per symbol may be calculated as $\log_2(529) = 9.047$ bits.

[0056] The receiver 320 receives the sequence of symbols using a set of line receivers 322 where each receiver in the set of line receivers 322 determines differences in signaling states on one pair of the *N* wires 314. Accordingly, ${}_NC_2$ receivers are used, where *N* represents the number of wires. The ${}_NC_2$ receivers produce a corresponding number of raw symbols as outputs. In the depicted *N*=4 wire example, the signals received on the four wires 314 are processed by 6 receivers (${}_4C_2 = 6$) to produce a state transition signal that is provided to a corresponding CDR 324 and deserializer 326. The CDR 324 may produce a receive clock signal 334 that can be used by the deserializer 326. The receive clock signal 334 may be a DDR clock signal that can be used by external circuitry to receive data provided by the transcoder 328. The transcoder 328 decodes a block of received symbols from the deserializer 326 by comparing each next symbol to its immediate predecessor. The transcoder 328 produces output data 330 corresponding to the data bits 304 provided to the transmitter 302.

Transition Encoding Example

[0057] FIG. 4 is a block diagram illustrating a transmitter 400 and a receiver 420 configured according to certain aspects disclosed herein. The transmitter 400 and receiver 420 may be adapted for use with a variety of encoding techniques, including transition encoding used in *N*! and CCIe interfaces. The transmitter 400 includes a first converter 404 configured to convert data 402 into transition numbers 414. The transition numbers 414 may be used to select a next symbol for transmission based on the value of a current symbol, where the next symbol is different from a current symbol. A second converter, such as the encoder 406, receives the transition numbers and produces a sequence of symbols for transmission on the interface using suitably configured line drivers 408. Since no pair of consecutive symbols includes two identical symbols, a transition of signaling state occurs in at least one of the signal wires 418 of the interface at every symbol transition. At the receiver 420, a set of line receivers 426 provides raw symbols (SI) 436 to a CDR 428 that extracts a receive clock 438 and provides captured symbols (S) 434 to a circuit that converts the captured symbols 434 to transition numbers 432. The transition numbers may be decoded by a circuit 422 to provide output data 430.

[0058] In the example of a 3! system, the transmitter 400 may be configured or adapted to transcode data 402 into quinary (base-5) transition numbers 414 represented by 3 bits. In the example of a CCIe system, the transmitter 400 may be configured or adapted to transcode data 402 into ternary (base-3) transition numbers 414 represented by 2 bits. The transition numbers 414 may be encoded in a sequence of symbols 416 to be transmitted on the signal wires 418. The data 402 provided to the transmitter 400 may be one or more words, each word having 20-bits. The first converter 404, which may be a transcoder, receives the data 402 and produces a sequence of transition numbers 414 for each data element. The sequence of transition numbers 414 may include a sufficient number of ternary numbers to encode a predefined number of bits of data, error detection and other information.

The encoder 406 produces a sequence of symbols 416 that are transmitted through line drivers 408. In one example, the line drivers 408 may include open-drain output transistors. In another example, the line drivers 408 may include push-pull drivers. The output sequence of symbols 416 generated by the encoder has a transition in the state of at least one of the signal wires 418 between each pair of consecutive symbols in the sequence of symbols 416 by ensuring that no pair of consecutive symbols include two identical symbols. The availability of a transition of state in at least one of the signal wires permits a receiver 420 to extract a receive clock 438 from the sequence of symbols 416.

[0059] FIG. 5 is a drawing illustrating a simple example of an encoding scheme 500. In this example, the encoding scheme may be used by the encoder 406 configured to produce a sequence of symbols 416 for transmission on a two-wire CCIe interface. The encoding scheme 500 is also used by a transcoder 424 to extract data from symbols received from signals transmitted on the signal wires 418 of the interface. In the illustrated encoding scheme 500, the use of two signal wires 418 permits definition of 4 basic symbols S: {0, 1, 2, 3}. Any two consecutive symbols in the sequence of symbols 416, 434 have different states, and the symbol sequences 0,0, 1,1, 2,2 and 3,3 are invalid combinations of consecutive symbols. Accordingly, only 3 valid symbol transitions are available at each symbol boundary, where the symbol boundary is determined by the transmit clock and represents the point at which a first symbol (Ps) terminates and a second symbol (Cs) begins. The first symbol may be referred to as the preceding or previous symbol 522 terminates and the second symbol may be referred to as the current symbol 524.

[0060] According to certain aspects disclosed herein, the three available transitions are assigned a transition number (T) 526 for each previous symbol 522. The value of T 526 can be represented by a ternary number. In one example, the value of transition number 526 is determined by assigning a symbol-ordering circle 502 for the encoding scheme. The symbol-ordering circle 502 allocates locations 504a-504d on the symbol-ordering circle 502 for the four possible symbols, and a direction of rotation 506 between the locations 504a-504d. In the depicted example, the direction of rotation 506 is clockwise. The transition number 526 may represent the separation between the valid current symbols 524 and the immediately preceding previous symbol 522. Separation may be defined as the number of steps along the direction of rotation 506 on the symbol-ordering circle 502 required to reach the current symbol 524 from the previous symbol 522. The number of steps can be expressed as a single digit base-3 number. It will be appreciated that a three-step difference between symbols can be represented as a $0_{\text{base-3}}$. The table 520 in FIG. 5 summarizes an encoding scheme employing this approach.

[0061] At the transmitter 400, the table 520 may be used to lookup a current symbol 524 to be transmitted, given knowledge of the previous symbol 522 and an input ternary number, which is used as a transition number 526. At the receiver 420, the table 520 may be used as a lookup to determine a transition number 526 that represents the transition between the previous symbol 522 and the current symbol 524. The transition number 526 may be output as a ternary number.

[0062] The use of a transcoder that embeds clock information in a sequence of symbols can disassociate data 402 received for transmission by a transmitter 400 from the sequence of symbols 416 transmitted on signal wires 418. Consequently, a received raw symbol 436 cannot be directly decoded to obtain the data 402 provided to the transmitter 400 without consideration of at least one previously transmitted symbol. This disassociation can render conventional error correction techniques ineffective. For example, a conventional system may append an error correction code (ECC) to data 402, where the ECC may be a cyclic redundancy code (CRC) calculated from a predefined block size of data 402 or a packet length. The ECC may be used to identify and/or correct occurrences of errors during transmission in a conventional interface, where the errors may include one or more bit errors.

[0063] In an interface that uses transition encoding, symbol errors manifest in bursts of bit errors at the receiver. That is, multiple bit errors can be caused by a single symbol transmission error. In these circumstances, a CRC often exceeds Hamming distance and is not a practical solution for error detection.

[0064] FIG. 6 is a timing diagram 600 that illustrates the relationship between symbols 602 and transition numbers 604, which may also be referred to herein as "transition symbols." In this example, each data word is encoded in *m* symbols transmitted on the multi-wire interface. A word transmitted in *m* symbols may be decoded using the formula:

$$\sum_{k=0}^{m-1} T_k r^k$$

where T_k is the transition number at the k^{th} iteration, and r is number of available symbols at each transition between symbols. For example, in a 3! interface where a self-transition is prohibited (to ensure that a receive clock can be reliably generated), r=5 states of the 6 defined states are available at each symbol transition. In various examples, the 3! interface may encode data in sequences of m=4 symbols or m=7 symbols. In a 4! interface, r=23 states of the 24 defined states are available at each symbol transition, and, the 4! interface may encode data in sequences of m=2 symbols. In a CCIe interface, r=3 states of the 4 defined states are available at each symbol and data words may be encoded in sequences of m=12 symbols. For a 3-wire single-ended interface, values of m=12 and r=7 may be used. For a 4-wire single-ended interface, values of m=10 and r=15 may be used.

[0065] FIG. 7 is a drawing 700 that illustrates transition number-to-symbol encoding for a 3! interface. In this example, there are 6 possible symbols, S: {0, 1, 2, 3, 4, 5}, arranged around the symbol-ordering circle 702. Clock information is embedded in sequences of symbols by ensuring that the same symbol does not appear in any two consecutive symbol intervals. In this example, r=5, and a transition number (T) may be assigned a different value for each type of transition 704, 706, 708, 710, 712. The value of the transition number may indicate the location of a next symbol on the symbol-ordering circle 702 relative to the position of a current symbol on the symbol-ordering circle 702. The transition number may take a value in the range 1-5. Since the current symbol cannot be the same as the previous symbol, the number of steps between the current and next symbols cannot be zero.

[0066] A transition number may be assigned in accordance with the formula:

$$T = Ps{+}1 \leq Cs \ \ ?\ Cs - (Ps{+}1) : Cs - (Ps{+}1) + 6$$

Conversely, the current sequential symbol number (Cs) may be assigned according to:

$$Cs = Ps{+}1 + T < 6\ ?\ Ps{+}1 + T : Ps{+}1{+}T{-}6$$

where Cs is the current symbol, and Ps is the previously received symbol.

[0067] FIG. 8 is a diagram that illustrates a generalized example 800 of symbol transition clocking transcoding. In this example 800, an interface provides six possible signaling states per symbol transmitted on a multi-wire communication interface, with clock information embedded at each transition between consecutive symbols by ensuring that each pair of consecutively transmitted symbols includes two different symbols. Accordingly, 5 states are available at each transition between symbols. A data word is encoded by converting the bits of the data word to a transition number, which selects the next symbol to be transmitted based on the symbol being currently transmitted. In the example 800, three sequential symbols 812, 814, 816 are transmitted over the multi-wire communication interface, where each symbol 812, 814, 816 defines one of the six signaling states of the multi-wire communication interface. Data and clock information are encoded in the transitions between consecutive pairs of the symbols 812, 814, 816. The transitions may be represented as digits of transition numbers 808, 810. Each digit of the transition number identifies a transition between a pair of consecutive symbols in the sequence of symbols, and in this context, the digits may also be referred to as transition numbers. As noted herein, for a sequence of m symbols data is encoded as:

$$data = \sum_{k=0}^{m-1} T_k r^k$$

where k has a value between 0 and m-1. A first transition number (T_k) 808 corresponds to the transition between a first symbol 812 (A) and a second symbol 814 (X), and a second transition number (T_{k-1}) 810 corresponds to the transition between the second symbol 814 (X) and a third symbol 816 (B). Here, the first symbol 812 may encode the most significant bits of a data word.

[0068] In one example, a multi-bit data word may be converted to a sequence of m transition numbers. Each transition number may be expressed using a ternary number, quaternary number, quinary number, senary number, or using some other numeral system that can represent r transitions. That is, the numeral system may be a base r system providing numbers that can span the range 0 to r-1. Each transition number may select a next symbol for transmission based on the current symbol being transmitted. The next symbol is selected from symbols that are different from the current symbol in order to ensure a signaling state transition occurs in order to embed clock information in the sequence of symbols 802. That is, the transmission of two different symbols in a consecutive pair of symbols results in a change in signaling state of at least one wire of a multi-wire interface, and a receiver can generate a receive clock based on the changes detected in signaling state between consecutive symbols.

[0069] The symbol-ordering circle 806 illustrates one method of selecting a next symbol in the example 800. Here, the transition number may be expressed as a quinary number (base-5), with possible values {0, 1, 2, 3, 4}. For each of six possible symbols 804a-804f, one of six signaling states is transmitted on the multi-wire communication interface. The six symbols 804a-804f are arranged in different positions around the symbol-ordering circle 806. Given a current symbol location on the symbol-ordering circle 806 a transition number T may be encoded by selecting, as a next symbol, the symbol located T clockwise steps on the symbol-ordering circle 806. In one example, when the current symbol is Symbol-0 804a, a transition number value of T=1 selects Symbol-1 804b as the next symbol, a transition number value of T=2 selects Symbol-2 804c as the next symbol, a transition number value of T=3 selects Symbol-3 804c as the next symbol, and a transition number value of T=4 selects Symbol-4 804d as the next symbol. A transition number value of T=0 may cause a rollover in that the transition number selects the symbol 5 clockwise steps (or 1 counterclockwise steps) from the current symbol (Symbol-0 804a), thereby selecting Symbol-5 804f as the next symbol.

[0070] In the example of the transmitted sequence of symbols 802, the first symbol 812 in the sequence of symbols 802 may correspond to Symbol-1 804b. Input data may be processed to produce the first transition number 808 with a value of $T_k = 2$, and the second transition number 810 with a value of $T_{k-1} = 1$. The second symbol 814 may be determined to be Symbol-3 804d based on the value of T_k and the third symbol 816 may be determined to be Symbol-4 804e based on the value of T_{k-1}.

[0071] At a receiver, the symbol-ordering circle 806 may be used to determine a transition number for each transition between consecutive symbols 812, 814, and/or 816. In one example, the receiver extracts a receive clock based on the occurrence of changes in signaling state between consecutive symbols 812, 814, and/or 816. The receiver may then capture the symbols 812, 814, 816 from the multi-wire interface and determine a transition number representing the transition between each pair of consecutive symbols 812, 814, and/or 816. In one example, the transition number may be determined by calculating the number of steps on the symbol-ordering circle 806 between the pair of consecutive symbols 812, 814.

Error Detection In A Transition Encoding Interface

[0072] According to certain aspects disclosed herein, reliable error detection may be implemented in a transition-encoded interface using an EDC added to data to be transmitted over the transition-encoded interface. The EDC may include a predefined number of bits with a predefined number of bits, where the EDC has a known, fixed value. In one example, the EDC has a zero value when transmitted. In some instances, the EDC is provided as the least significant bits (LSBs) of each word to be transmitted on the interface. The form and structure of the EDC word may be selected such that a single signaling state error affecting a word causes the EDC decoded at the receiver to have a value that is different from the fixed value (e.g., a non-zero value).

[0073] FIG. 9 illustrates an example 900 of the effect of a single error affecting a transition-encoded interface. In the example, a data word 912 is provided for transmission over the interface. An EDC 914 is appended to the data word 912 to produce a transmission word 902 that is input to and encoder. The transmission word 902 is transmitted in a sequence of symbols 910, where the sequence of symbols 910 includes 12 symbols. The sequence of symbols 910 is transmitted over a two-wire interface configured for CCIe operation and received at a receiver in a stream of symbols 904. In transmission, a signaling error occurs such that an originally-transmitted symbol 916 is modified and received as an erroneous symbol 918. A stream of transition numbers 906 corresponding to the received stream of symbols 904 includes transition numbers 920, 922 that include error offsets. A first transition number 920 represents the difference between the preceding symbol and the erroneous symbol 918, and a second transition number 922 represents the difference between the erroneous symbol 918 and the next symbol transmitted after the affected symbol.

[0074] The size, location, and structure of the EDC 914 may be selected such that the occurrence of a single symbol error produces an EDC 926 at the receiver that is different than the transmitted EDC 914. In one example, the EDC 914 includes multiple bits and may be set to a zero value. In the example of a CCIe interface, the EDC 914 may have three bits.

[0075] FIG. 10 is a diagram that illustrates an example in which a sequence of symbols 1002 transmitted over a multi-wire communication interface is affected by a single symbol error 1018 resulting in the capture of an erroneous symbol 1014 in the received sequence of symbols 1004. The transmitted sequence of symbols 1002 includes a first symbol 1008 (the A symbol), a second symbol (the X symbol 1010) and a third symbol 1012 (the B symbol). In the received sequence of symbols 1004, the first symbol 1008 and the third symbol 1012 are correctly received, while the second symbol 1014 is modified by the symbol error 1018 (displacement *e*) and is received as an erroneous symbol (the X′ symbol 1014).

[0076] The occurrences of a single symbol error 1018 results in two transition number errors. The first incorrect transition number 1020 represents the transition between the correctly received first symbol 1008 and the X′ Symbol 1014. The second incorrect transition number 1022 represents the transition between the X′ Symbol 1014 and the correctly received third symbol 1012. The first incorrect transition number 1020 may be expressed as $T_k + e_k$, where T_k is the first correct transition number 1016 corresponding to a transition between the first symbol 1008 and the X Symbol 1010, and e_k is the value of the error created in the first incorrect transition number 1020 relative to the first correct transition number 1016. The second incorrect transition number 1022 may be expressed as $T_{k-1} + e_{k-1}$, where T_{k-1} is the second correct transition number 1024 corresponding to the transition between the X Symbol 1010 and the third symbol 1012, and e_{k-1} is the value of the error created in the second incorrect transition number 1022 relative to the first correct transition number 1024.

[0077] The effect of the single symbol error 1018 is illustrated in the decoding transition circle 1006. The first symbol 1008, which corresponds to Symbol-1, is initially received from the multi-wire interface. The next symbol is incorrectly captured as the X′ Symbol 1014 due to error. The X′ Symbol 1014 may correspond to Symbol-0. The third symbol 1012, which corresponds to Symbol-4, is then received from the multi-wire interface. In this example, the most significant symbol is transmitted first, and:

$$e = 3, T_k = 2, T_{k-1} = 1$$

$$T_k + e = 2 + 3 = 5 = 0_{base5}, \text{ and } e_k = -2$$

$$T_{k-1} - e = 1 - 3 = -2 = 4_{base5}, \text{ and } e_{k-1} = -3$$

Each data word may be represented by a sequence of transition numbers: $\{T_0, T_1, \ldots, T_{m-1}\}$.

[0078] The displacement error *e* represents the difference between the transmitted X symbol 1010 and the received X′ Symbol 1014, which may correspond to a number of steps in the decoding transition circle 1006. The value of *e* is not necessarily equal in value to e_k due to roll over in the number system used to express transition numbers. For example, a transition number with a value of 3 may represent the difference between the transmitted X symbol 1014 and the received X′ Symbol 1014 the first correct transition number 1016 on the decoding transition circle 1006 caused by the displacement error *e*, while the value of e_k has a value of -2.

[0079] For two consecutive symbol transitions:

$$Bits = T_k r^k + T_{k-1} r^{k-1}$$

The result of a single error affecting two consecutive symbols may be expressed as:

$$\begin{aligned} Bits' &= (T_k + e_k) r^k + (T_{k-1} - e_{k-1}) r^{k-1} \\ &= (T_k r^k + T_{k-1} r^{k-1}) + (e_k r - e_{k-1}) r^{k-1} \end{aligned}$$

where:

$(e_k r - e_{k-1}) r^{k-1}$ may be referred to as the error effect,

$(e_k r - e_{k-1})$ may be referred to as the error coefficient, and

r^{k-1} may be referred to as the base power.

[0080] According to certain aspects, a transition-encoded interface may be configured such that *r* is an odd number. When *r* is an odd number, it follows that r^{k-1} is also an odd number (LSB is non-zero). Accordingly, the value of ($e_k r$ - e_{k-1}) determines the number of LSBs required for an EDC. FIG. 11 provides a listing of r^n (where n lies in the range 0 to 15) when *r*=3 and 5. The first table 1100 may relate to a CCIe interface, where *r*=3 transitions are available at each symbol interval. In each instance, the LSB 1104 of the base power is set to ‘1.’ The second example 1102 may relate to a 3-wire 3! interface, where *r*=5 transitions are available at each symbol interval (6 possible symbols). In each instance, the LSB 1206 of the base power is set to ‘1.’

[0081] FIG. 12 is a table 1200 that tabulates error coefficients and illustrates error coefficient when a symbol error does not involve repetition of a symbol in consecutive symbol intervals, which would cause a clock miss. $|e_k|$ is always smaller than *r*. That is:

$$1 \leq |e_k| \leq r\text{-}1,$$

$$1 \leq |e_{k-1}| \leq r\text{-}1.$$

Since the least value of $|e_k|$ is 1, the least value for $|e_k r|$ is *r*. The largest value of $|e_{k-1}|$ is *r*-1. The error coefficient ($e_k r$ - e_{k-1}) is never zero when a single symbol error is present.

[0082] FIG. 13 illustrates an example 1300 of calculation and tabulation of the longest non-zero LSB portion in an error coefficient. Here, the power of 2 LSBs of ($e_k r$ – e_{k-1}) is the longest when both $|e_k|$ and $|e_{k-1}|$ are longest power of 2 (2^n), and $e_k = e_{k-1}$. The Longest power of 2 LSBs of error coefficient determines the size of the “error detection constant LSBs.”

[0083] FIG. 14 illustrates two examples 1400, 1420 of cases in which a single symbol error results in an error in a single transition number 1408, 1426. In the first example 1400, a signaling error affects the last transmitted symbol 1402 in a preceding sequence of symbols. The signaling error causes a receiver to detect a modified symbol 1404 as the last-received symbol in the preceding sequence of symbols. The error may introduce an offset in the transition number 1406 that represents the difference between the last transmitted symbol 1402 in a preceding sequence of symbols and the first symbol of a current sequence of symbols. In the first example 1400, the effect of the error may be expressed as: $e_{m-1}r^{m-1}$, where the error coefficient is e_{m-1} and the base power is r^{m-1}.

[0084] In the second example 1420, a signaling error affects the last transmitted symbol 1422 in a current sequence of symbols. The signaling error causes a receiver to detect a modified symbol 1424 as the last-received symbol in the current sequence of symbols. The error may introduce an offset in the transition number 1426 that represents the difference between the last transmitted symbol 1422 in the current sequence of symbols and the first symbol of a next sequence of symbols. In the first example 1400, the effect of the error may be expressed as e_0.

[0085] Table 1 lists the number of LSBs in an EDC that can detect a single symbol error in a multi-wire interface that uses transition encoding.

r	EDC length (bits)	Example
3	3	2-wire single-ended (e.g. CCIe)
5	5	3-wire multi-level differential (3!)
7	6	3-wire single ended
9	7	
11	5	
13	6	
15	8	4-wire single-ended
17	9	
19	8	
21	7	
23	6	4-wire multi-level differential (4!)

Table 1

[0086] The cases illustrated in FIG. 14 do not affect the maximum number of LSBs required in an EDC to permit detection of a single symbol error.

Detection Of Multiple Symbol Errors Per Word

[0087] FIG. 15 is a timing diagram 1500 that illustrates a first example of signaling errors that affect two symbols 1504, 1506 in a sequence of symbols 1502 that encodes a single data word. FIG. 15 relates to an example in which signaling errors affect two non-consecutive symbols. The errors in symbols 1504, 1506 result in corresponding pairs of transition errors 1508, 1510. These transition errors result in erroneous transition numbers 1512, 1514, 1516, 1518. The error effect attributable to the first affected symbol 1504 may be stated as $(e_k r - e_{k-1})r^{k-1}$, while the error effect attributable to the first affected symbol 1504 may be stated as $\left(e_j r - e_{j-1}\right)r^{j-1}$. Multiple symbol errors can be detected provided if the total effect of the error

$$(e_k r - e_{k-1})r^{k-1} + (e_k r - e_{k-1})r^{k-1}$$

always modifies an EDC that has a predetermined length and value.

[0088] FIG. 16 is a timing diagram 1600 that illustrates a second example of signaling errors that affect two consecutive symbols 1604, 1606 in a sequence of symbols 1602 that encodes a single word. The errors in the consecutive symbols 1604, 1606 result in transition errors 1608 that cause the generation of three erroneous transition numbers 1610, 1612, 1614. The error effect attributable to the affected symbols 1504, 1506 may be stated as $(e_k r^2 + e_{k-1} r + e_{k-2})r^{k-2}$. The error effect attributable to errors affecting consecutive symbols 1604, 1606 can be detected with a shorter EDC than errors in non-consecutive symbols 1504, 1506 in receivers adapted in accordance with certain aspects disclosed herein.

[0089] FIG. 17 is a table 1700 that illustrates the number of bits of an EDC used for various values of *r* (available transitions per symbol boundary) and *m* (number of symbols used to encode a data element). The size of an EDC used for detecting two symbol errors varies with the value of m. The first row (shaded) of the table 1700 corresponds to an EDC used to detect a single symbol error.

[0090] According to certain aspects disclosed herein, a receiver can be configured to detect two symbol errors in a sequence of symbols representing a data word, when an EDC of sufficient length is transmitted with the data word. The length of the EDC may be determined based on the number of symbols used to encode a data word and the number of transitions available at the boundary between a pair of consecutively transmitted symbols.

[0091] Symbol slip error caused by clock miss or extra clock may not be detected by an error detection constant. However, the majority of these types of errors can be detected by higher protocol layers, at the next word, and/or using a state machine at the receiver device.

Correction of a Single Symbol Error per Word

[0092] FIG. 18 is a diagram 1800 illustrating the operation of certain circuits that can be used to correct a single symbol error per word in accordance with certain aspects disclosed herein. An error correction circuit may correct a single symbol error in a sequence of symbols 1802 that encodes a word, when an EDC transmitted in the sequence of symbols 1802 enables detection of two symbol errors in the sequence of symbols 1802. The error correction circuit may be configured to generate and decode different permutations of the symbols in a sequence of symbols 1802 received from a communication link after an error has been detected in the sequence of symbols 1802. Each permutation may include a modification to one symbol in the sequence of symbols 1802. When a single symbol error is introduced during transmission, changing a correctly received symbol in the sequence of symbols 1802 causes a decoder to decode an EDC that indicates presence of one or more errors in the sequence of symbols 1802. Changing a symbol in the sequence of symbols 1802 that has an error to another incorrect symbol results in an EDC that indicates that the single error remains in the sequence of symbols 1802. When a modified symbol in a permutation of the symbols reverses the error introduced during transmission, the decoder decodes an EDC that matches the expected, originally transmitted EDC value and the permutation of the symbols and/or data decoded from the permutation of the symbols may be used by the receiver in place of the received sequence of symbols 1802 and/or data decoded therefrom.

[0093] In the context of FIG. 18, a transmitter may have encoded a combined data word and an EDC in *m* symbols. In the example illustrated in FIG. 9, a transmission word 902 may include a data word 912 and an EDC 914. A receiver may determine that at least one symbol has been affected by a transmission error if, after decoding the *m* symbols in the sequence of symbols 1802, the decoded EDC has an unexpected value. The received EDC may have an unexpected value when it does not match a predefined value of the EDC 914 used for transmissions over the communication link and appended to the data word 912 by the transmitter. In one example, the transmitter may append a number of zero-value bits as the least significant bits of the transmission word 902, and a non-zero decoded EDC may indicate that at least one symbol error has occurred in transmission the sequence of symbols 1802.

[0094] An error correction circuit may correct a single symbol error when the EDC can be used to detect the occurrence of two symbol errors in one sequence of symbols 1802. In one example, an EDC appended as a sufficient number of least significant bits to a data word may permit two symbol errors to be detected, and a single symbol error to be corrected by an error correction circuit.

[0095] The error correction circuit may be enabled when an unexpected EDC value is decoded from the sequence of symbols 1802. The error correction circuit may correct a single symbol error by decoding each permutation of symbols obtained by modifying a single symbol in the sequence of symbols 1802. A permutation of symbols that produces a decoded EDC with the expected EDC value may be considered to have corrected the transmission error. That is, the permutation of symbols includes a symbol that has been modified in a manner that reverses the prior modification caused by the transmission error.

[0096] In the example illustrated in the diagram 1800 of FIG. 18, each symbol in the sequence of symbols 1802 can take one of a plurality of r+1 possible signaling states, where r represents the number of possible transitions per symbol. That is, in an encoding scheme that embeds clock information, each consecutively transmitted pair of symbols include two different symbols to ensure a transition occurs at each symbol boundary, and one of the r+1 defined symbols is not available at each symbol boundary. In a first example, a two-wire interface may be operated such that r+1=4 signaling states are available. In a second example, a three-wire 3! interface may be operated such that r+1=6 signaling states are available. A transmission error may produce a symbol S_i that is assigned a value V_{err} by a receiver that is different from the value $V_{correct}$ that would have been assigned to the originally transmitted symbol. A decoder may produce a received EDC that does not have an expected value after decoding the sequence of symbols 1802 that includes a symbol affected by error (i.e., $S_i = V_{err}$). The expected EDC value 1818 may have a predefined EDC value.

[0097] According to certain aspects, the value of S_i may be iteratively changed and provided in a permutation of the sequence of symbols 1802 that is then decoded to determine if the permutation of symbols produces an EDC that matches the expected

EDC value 1818. When the symbol affected by error is replaced in a permutation of symbols with the originally transmitted symbol value (i.e., $S_i = V_{correct}$), the decoded EDC has the expected EDC value 1818, and the permutation of symbols may be substituted for the received sequence of symbols 1802, and/or a data word decoded from the permutation of symbols may be substituted for the data word decoded from the received sequence of symbols 1802.

[0098] In some implementations, the values of all symbols 1812 in the sequence of symbols 1802 may be provided to a set of decoders 1808 that includes *r* (i.e., (*r*+1) - 1) symbols-to-data decoders. Each decoder in the set of decoders 1808 receives the value of each symbol in the sequence of symbols 1802, except for the value of S_i. In place of the value of S_i, each decoder in the set of decoders 1808 receives a modified version of S_i, and accordingly each decoder in the set of decoders 1808 decodes a permutation of symbols derived from the sequence of symbols 1802. A set of adders 1806 may be configured to provide different versions of S_i to the set of decoders 1808. The set of adders 1806 may include *modulo* (*r*) adders that each provide a different symbol value to substitute for S_i in a permutation of symbols.

[0099] In the example depicted in FIG. 18, all permutations involving changes to a single symbol can be processed and decoded in parallel. The EDC values 1816 decoded from multiple permutations of symbols can be compared using multi-bit comparators 1810, each of which may produce a logic 1 if a match exists between the expected EDC value 1818 and the EDC value 1816 decoded from a corresponding permutation of symbols. If one of the multi-bit comparators 1810 indicates a match, then the transmission error affecting the sequence of symbols 1802 may be determined to have been corrected.

[00100] In some implementations, an error correction block may be provided for each symbol in the sequence of symbols 1802, and sets of permutations may be generated and decoded concurrently for each symbol, where each set of permutations includes modifications of a single symbol in the sequence of symbols 1802. Each error correction block may include a set of adders 1806, a set of decoders 1808, and multi-bit comparators 1810. In these implementations, error correction is accomplished when an error correction block provides a match signal from one of the comparators 1810 in the error correction block, and the permutation of symbols and/or corresponding data word may be provided as outputs of the error-correction circuit.

[00101] In other implementations, an error correction circuit may use a state machine, sequencing logic and/or one or more processing circuits to perform error correction in multiple iterations. In some of these implementations, up to *m* error correction cycles may be executed to test the effect of permutations to each symbol (S_0, S_1,... S_i, ...S_m) in turn. In a first error correction cycle, the *r* permutations related to the first symbol (S_0) in the sequence of symbols 1802 are decoded to determine if an expected EDC value can be decoded. In the second error correction cycle, the *r* permutations related to the first symbol (S_1) in the sequence of symbols 1802 are decoded to determine if an expected EDC value can be decoded. This process may continue until a permutation is discovered that matches the expected EDC value 1818 when decoded. The error correction process may terminate when an EDC match is obtained.

Examples of Error Correction Techniques According to Certain Aspects

[00102] The hardware used to implement an error correction circuit may be reduced by sharing the components used to generate and decode permutations of symbols. For example, the set of adders 1806 may be replaced with a *modulo* (*r*) counter that is preloaded with the value of S_i in the received sequence of symbols 1802. For each of *r* iterations, the counter may be incremented and its output may be provided to a decoder as a substitute for the value of S_i in the received sequence of symbols 1802. The EDC produced by the decoder may be compared with the expected EDC value 1818. In some examples, a counter and decoder may be provided for each symbol in the sequence of symbols 1802 such that *m* permutations are generated and decoded in each cycle of the error detection process, which may extend to *r* cycles. In other examples, a single counter may be used and the maximum number of error detection cycles may be calculated as $m(r\text{-}1)$ cycles.

[00103] Various combinations of hardware may be employed to correct errors according to certain aspects disclosed herein. In one hardware-intensive example, counters, adders and decoders may be provided such that every possible permutation of symbols may be tested concurrently. In the latter example, the error correction circuit may operate with low latency. In another example, a single encoder may be used to test every permutation sequentially, such that error correction may require significant time to complete the error correction process. According to certain aspects, a tradeoff between hardware and error correction process time may be made

based on application requirements. In some instances, permutation generation, decoding, and/or error correction decisions may be implemented in a sequencer or processing device that is controlled by software instructions.

[00104] FIG. 19 is diagram 1900 that illustrates a first example of a configuration of decoders that may be used to correct symbol errors in accordance with certain aspects disclosed herein. In this example, sequencing logic, a state machine, or a controller may be used to perform error correction in a number of iterations where, in each iteration, permutations created by modifying a single symbol may be tested. In the example, a data word may be encoded in a sequence of symbols that includes 10 symbols $\{S_0, S_1, \ldots S_9\}$, and each symbol can have one of 4 values. When a symbol error has been determined to be present in a received sequence of symbols 1902, three permutations 1904, 1906, 1908 of the sequence of symbols 1902 may be concurrently generated for each of a plurality of test iterations. In each permutation 1904, 1906, 1908 generated in a test iteration, only one symbol location (the target symbol 1910) in the sequence of symbols 1902 is changed. In the example, the first iteration is in progress and only S_0 in the sequence of symbols 1902 is changed in the three permutations 1904, 1906, 1908. In one example, the value of S_0 may be modified by simple modulo 3 addition. The three permutations 1904, 1906, 1908 may then be tested using decoder circuits and/or logic 1912 that includes 3 decoders and comparators such that the EDC generated for all three permutations 1904, 1906, 1908 can be compared to the expected EDC value 1914. If correction is not accomplished, another iteration is performed for the next symbol (here, S_1) until all 10 symbols have been tested. This approach uses a maximum of 10 test iterations and 3 decoder paths.

[00105] FIG. 20 is diagram 2000 that illustrates a second example of a configuration of decoders that may be used to correct symbol errors in accordance with certain aspects disclosed herein. In this example, sequencing logic, a state machine, or other controller may be used to perform error correction in a number of iterations where, in each iteration, each symbol in a sequence of symbols 2002 is modified in one of a set of permutations 2004 generated per iteration. In the example, a data word may be encoded in a sequence of symbols that includes 10 symbols $\{S_0, S_1, \ldots S_9\}$, and each symbol can have one of 4 values. When a symbol error has been determined to be present in the received sequence of symbols 2002, 10 permutations 2004 of the sequence of symbols 2002 may be concurrently generated for each of a plurality of

test iterations. In each of the 10 permutations 2004 generated per test iteration, a different one of the symbol locations (S_0, S_1, … S_9} in the sequence of symbols 2002 is changed. In the example, the first iteration is in progress and a single symbol in each of the 10 permutations 2004 differs from the corresponding symbol in the sequence of symbols 2002 by a value of one. A symbol is changed by using modulo 3 addition to perform $S_j + 1$, where S_j represents the value of the corresponding symbol in the sequence of symbols 2002. The 10 permutations 2004 may then be tested using decoder circuits and/or logic 2006 that includes 10 decoders and comparators such that the EDC generated for all 10 permutations 2004 can be compared to the expected EDC value 2008. If correction is not accomplished, another iteration is performed using the next symbol value increment ($S_j + 2$, then $S_j + 3$) until all 3 possible values for all symbols in the sequence of symbols 2002 have been tested. This approach uses a maximum of 3 test iterations and 10 decoder paths.

[00106] FIG. 21 is a diagram 2100 that illustrates certain aspects of decoders used to correct symbol errors that may be exploited to reduce decoding overhead in accordance with certain aspects disclosed herein. The illustrated example is based on the configuration of decoders illustrated in FIG. 19, where a data word may be encoded in a sequence of symbols that includes 10 symbols {S_0, S_1, … S_9} and each symbol can have one of 4 values, and where decoder circuits and/or logic 1912 include 3 decoders and comparators such that the EDC generated for all three permutations 1904, 1906, 1908 can be compared to the expected EDC value 1914. The processing overhead, and number of decoders may be reduced by considering the effect of changing a symbol in a received sequence of symbols 2102. In this example, error correction is in the fifth iteration, and S_4 is modified in each permutation. The 4 possible states for symbols may be represented as {A, B, C, D}, where B = A + 1, C = B + 1, D = C +1, and A = D + 1. In the received sequence of symbols 2102, S_3 = A, and S_4 = B. It can be seen that the third permutation 2108 changes the value of S_4 such that $S_4 = S_3$ = A. In transition encoding schemes, consecutively transmitted pairs of symbols cannot have the same value. In some embodiments, error correction circuits and modules may be adapted to recognize permutations 2108 that introduce an error condition when a pair of consecutive symbols 2112 are identical, and such permutations 2108 need not be tested. In some instances, permutations 2104, 2106, 2108 may be dynamically assigned to decoding paths in decoder circuits and/or

logic 1912 that provide a reduced number of decoders and comparators reflecting the ability to eliminate at least one permutation 2108 due to creation of invalid pairs of consecutive symbols 2112.

[00107] The examples illustrated in FIGs. 18-21 can be scaled. For example, in FIG. 19, the decoder circuits and/or logic 1912 may be adapted to include a larger number of decoders and comparators such that sets of permutations corresponding to two or more symbols in the sequence of symbols 1902 may be concurrently tested (i.e., 6, 9, 12, etc. permutations can be tested per iteration). In another example, the decoder circuits and/or logic 2006 of FIG. 20 may be adapted to include fewer decoders and comparators such that an increased number of iterations is used to test all permutations of the sequence of symbols 1902.

Example of a Processing Circuit

[00108] FIG. 22 is a conceptual diagram 2200 illustrating a simplified example of a hardware implementation for an apparatus employing a processing circuit 2202 that may be configured to perform one or more functions disclosed herein. In accordance with various aspects of the disclosure, an element, or any portion of an element, or any combination of elements as disclosed herein may be implemented using the processing circuit 2202. The processing circuit 2202 may include one or more processors 2204 that are controlled by some combination of hardware and software modules. Examples of processors 2204 include microprocessors, microcontrollers, digital signal processors (DSPs), field programmable gate arrays (FPGAs), programmable logic devices (PLDs), state machines, sequencers, gated logic, discrete hardware circuits, and other suitable hardware configured to perform the various functionality described throughout this disclosure. The one or more processors 2204 may include specialized processors that perform specific functions, and that may be configured, augmented or controlled by one of the software modules 2216. The one or more processors 2204 may be configured through a combination of software modules 2216 loaded during initialization, and further configured by loading or unloading one or more software modules 2216 during operation.

[00109] In the illustrated example, the processing circuit 2202 may be implemented with a bus architecture, represented generally by the bus 2210. The bus 2210 may include any number of interconnecting buses and bridges depending on the specific

application of the processing circuit 2202 and the overall design constraints. The bus 2210 links together various circuits including the one or more processors 2204, and storage 2206. Storage 2206 may include memory devices and mass storage devices, and may be referred to herein as computer-readable media and/or processor-readable media. The bus 2210 may also link various other circuits such as timing sources, timers, peripherals, voltage regulators, and power management circuits. A bus interface 2208 may provide an interface between the bus 2210 and one or more transceivers 2212. A transceiver 2212 may be provided for each networking technology supported by the processing circuit. In some instances, multiple networking technologies may share some or all of the circuitry or processing modules found in a transceiver 2212. Each transceiver 2212 provides a means for communicating with various other apparatus over a transmission medium. Depending upon the nature of the apparatus, a user interface 2218 (e.g., keypad, display, touch interface, speaker, microphone, joystick) may also be provided, and may be communicatively coupled to the bus 2210 directly or through the bus interface 2208.

[00110] A processor 2204 may be responsible for managing the bus 2210 and for general processing that may include the execution of software stored in a computer-readable medium that may include the storage 2206. In this respect, the processing circuit 2202, including the processor 2204, may be used to implement any of the methods, functions and techniques disclosed herein. The storage 2206 may be used for storing data that is manipulated by the processor 2204 when executing software, and the software may be configured to implement any one of the methods disclosed herein.

[00111] One or more processors 2204 in the processing circuit 2202 may execute software. Software shall be construed broadly to mean instructions, instruction sets, code, code segments, program code, programs, subprograms, software modules, applications, software applications, software packages, routines, subroutines, objects, executables, threads of execution, procedures, functions, algorithms, etc., whether referred to as software, firmware, middleware, microcode, hardware description language, or otherwise. The software may reside in computer-readable form in the storage 2206 or in an external computer readable medium. The external computer-readable medium and/or storage 2206 may include a non-transitory computer-readable medium. A non-transitory computer-readable medium includes, by way of example, a magnetic storage device (e.g., hard disk, floppy disk, magnetic strip), an optical disk (e.g., a compact disc (CD) or a digital versatile disc (DVD)), a smart card, a flash memory device (e.g., a "flash drive," a card, a stick, or a key drive), a random access memory (RAM), a read only memory (ROM), a programmable ROM (PROM), an erasable PROM (EPROM), an electrically erasable PROM (EEPROM), a register, a removable disk, and any other suitable medium for storing software and/or instructions that may be accessed and read by a computer. The computer-readable medium and/or storage 2206 may also include, by way of example, a carrier wave, a transmission line, and any other suitable medium for transmitting software and/or instructions that may be accessed and read by a computer. Computer-readable medium and/or the storage 2206 may reside in the processing circuit 2202, in the processor 2204, external to the processing circuit 2202, or be distributed across multiple entities including the processing circuit 2202. The computer-readable medium and/or storage 2206 may be embodied in a computer program product. By way of example, a computer program product may include a computer-readable medium in packaging materials. Those skilled in the art will recognize how best to implement the described functionality presented throughout this disclosure depending on the particular application and the overall design constraints imposed on the overall system.

[00112] The storage 2206 may maintain software maintained and/or organized in loadable code segments, modules, applications, programs, etc., which may be referred to herein as software modules 2216. Each of the software modules 2216 may include instructions and data that, when installed or loaded on the processing circuit 2202 and executed by the one or more processors 2204, contribute to a run-time image 2214 that controls the operation of the one or more processors 2204. When executed, certain instructions may cause the processing circuit 2202 to perform functions in accordance with certain methods, algorithms and processes described herein.

[00113] Some of the software modules 2216 may be loaded during initialization of the processing circuit 2202, and these software modules 2216 may configure the processing circuit 2202 to enable performance of the various functions disclosed herein. For example, some software modules 2216 may configure internal devices and/or logic circuits 2222 of the processor 2204, and may manage access to external devices such as the transceiver 2212, the bus interface 2208, the user interface 2218, timers, mathematical coprocessors, and so on. The software modules 2216 may include a control program and/or an operating system that interacts with interrupt handlers and device drivers, and that controls access to various resources provided by the processing circuit 2202. The resources may include memory, processing time, access to the transceiver 2212, the user interface 2218, and so on.

[00114] One or more processors 2204 of the processing circuit 2202 may be multifunctional, whereby some of the software modules 2216 are loaded and configured to perform different functions or different instances of the same function. The one or more processors 2204 may additionally be adapted to manage background tasks initiated in response to inputs from the user interface 2218, the transceiver 2212, and device drivers, for example. To support the performance of multiple functions, the one or more processors 2204 may be configured to provide a multitasking environment, whereby each of a plurality of functions is implemented as a set of tasks serviced by the one or more processors 2204 as needed or desired. In one example, the multitasking environment may be implemented using a timesharing program 2220 that passes control of a processor 2204 between different tasks, whereby each task returns control of the one or more processors 2204 to the timesharing program 2220 upon completion of any outstanding operations and/or in response to an input such as an interrupt. When a task has control of the one or more processors 2204, the processing circuit is effectively specialized for the purposes addressed by the function associated with the controlling task. The timesharing program 2220 may include an operating system, a main loop that transfers control on a round-robin basis, a function that allocates control of the one or more processors 2204 in accordance with a prioritization of the functions, and/or an interrupt driven main loop that responds to external events by providing control of the one or more processors 2204 to a handling function.

[00115] FIGs. 23 and 24 are flowcharts 2300 and 2400 illustrating methods that may be used cooperatively to receive and correct transmission errors at a receiver in a multi-wire interface.

[00116] In the first flowchart 2300 (see FIG. 23), at block 2302 the receiver may receive a sequence of symbols representative of the signaling state of a plurality of connectors.

[00117] At block 2304, the receiver may decode the sequence of symbols. An example of symbol decoding is illustrated by the second flowchart 2400 (see FIG. 24).

[00118] At block 2304, the receiver may determine whether a symbol error is present in the sequence of symbols based on a value of an EDC in the received plurality of bits.

[00119] If no error is detected, then at block 2308 the receiver may proceed to block 2310 and skip error detection. If an error is detected, then at block 2308 the receiver may initiate error detection by continuing at block 2312.

[00120] At block 2312, the receiver may generate one or more permutations of the sequence of symbols. Each permutation may include one symbol that is different from corresponding symbols in the sequence of symbols and different from corresponding symbols in other permutations. The receiver may generate a permutation for each possible value of one symbol in the sequence of symbols. In one example, the receiver may generate a permutation for each possible change in value of one symbol in the sequence of symbols. In another example, the receiver may generate a permutation corresponding to each symbol in the sequence of symbols, where a symbol is modified by a same value in each permutation.

[00121] At block 2314, the receiver may decode the one or more permutations. An example of symbol decoding is illustrated by the second flowchart 2400 (see FIG. 24). The receiver may refrain from decoding a permutation that includes two consecutive symbols having the same value.

[00122] At block 2316, the receiver may identify a permutation that produces a decoded EDC value that matches an expected EDC value. The expected EDC value may correspond to a predefined value for EDCs transmitted over the multi-wire interface to enable detection of up to two symbol errors at the receiver. The signaling error may cause a decoded version of the EDC to have a value that is different from the predefined value.

[00123] In a first example, after determining that a symbol error is present in the sequence of symbols, the receiver may concurrently generate a set of permutations of the sequence of symbols in which all members of the set of permutations include modified versions of one symbol in the sequence of symbols. That is, one target symbol in the sequence of symbols may be tested using a set of permutations in which only the target symbol is modified from permutation to permutation. The receiver may decode all members of the set of permutations concurrently. If a correct EDC is not decoded, then a next target symbol is selected and a set of permutations generated to include different values of the next symbol. This process is repeated until all symbols in the sequence of symbols have been tested.

[00124] In a second example, after determining that a symbol error is present in the sequence of symbols, the receiver may concurrently generate and decode all possible permutations of the sequence of symbols, where each permutation includes only one symbol that is modified with respect to the original sequence of symbols, and each permutation is different from all other permutations. The receiver may concurrently compare an EDC value decoded from each possible permutation of the sequence of symbols with the expected EDC value. In this example, a maximum number of decoders may be required. In this example, error correction can be performed with minimized delay.

[00125] In a third example, after determining that a symbol error is present in the sequence of symbols, the receiver may generate a set of permutations of the sequence of symbols in which the members of the set of permutations include modified versions of different symbols in the sequence of symbols. That is, each permutation in a first set of permutations, for example, includes one symbol that is modified by a fixed value x (using modulo(r) addition), each permutation in a second set of permutations includes one symbol that is modified by a fixed value x+1, and so on. Each set permutations may be tested before the next set of permutations is generated. The receiver may decode all members of the set of permutations concurrently.

[00126] In some instances, the predefined value for EDCs transmitted over the multi-wire interface has a fixed length that is determined based on a total number of states per symbol defined for encoding data transmissions on the plurality of connectors. The EDC may be provided as a predefined number of least significant bits in the plurality of bits. The predefined value for EDCs transmitted over the multi-wire interface may be determined based on a total number of symbols used to encode the plurality of bits.

[00127] A clock may be embedded in transitions between symbols in the sequence of symbols.

[00128] The plurality of connectors may have a number (N) of single-ended connectors. In one example, the total number of states per symbol available for encoding data transmissions is 2^N - x, where x is an integer greater than zero. In another example, the total number of states per symbol available for encoding data transmissions is $N!$- x, where x is an integer greater than zero.

[00129] The second flowchart 2400 (see FIG. 24) illustrates an example of a method for decoding symbols. At block 2404 the receiver may convert the sequence of symbols

into a transition number. Each digit of the transition number may represent a transition between two consecutive symbols transmitted on the plurality of connectors.

[00130] At block 2406 the receiver may convert the transition number into a received plurality of bits.

[00131] FIG. 25 is a conceptual diagram illustrating an example of a hardware implementation for an apparatus 2500 employing a processing circuit 2502. In this example, the processing circuit 2502 may be implemented with a bus architecture, represented generally by the bus 2516. The bus 2516 may include any number of interconnecting buses and bridges depending on the specific application of the processing circuit 2502 and the overall design constraints. The bus 2516 links together various circuits including one or more processors, represented generally by the processor 2512, and computer-readable media, represented generally by the processor-readable storage medium 2514. The bus 2516 may also link various other circuits such as timing sources, timers, peripherals, voltage regulators, and power management circuits. A transceiver or communications interface 2518 provides a means for communicating with various other apparatus over a multi-wire interface 2520. Depending upon the nature of the apparatus, a user interface (e.g., keypad, display, speaker, microphone, joystick) may also be provided. One or more clock generation circuits may be provided within the processing circuit 2502 or controlled by the processing circuit 2502 and/or one or more processors 2512. In one example, the clock generation circuits may include one or more crystal oscillators, one or more phase-locked loop devices, and/or one or more configurable clock trees.

[00132] The processor 2512 is responsible for managing the bus 2516 and general processing, including the execution of software stored on the processor-readable storage medium 2514. The software may include code and/or instructions that, when executed by the processor 2512, cause the processing circuit 2502 to perform the various functions described *supra* for any particular apparatus. The processor-readable storage medium 2514 may be used for storing data that is manipulated by the processor 2512 when executing software.

[00133] In one configuration, the processing circuit may include one or more modules and/or circuits 2504 configured for receiving a sequence of symbols representative of the signaling state of a plurality of connectors of the multi-wire interface 2520, and one or more decoding modules and/or circuits 2506 configured for decoding the

sequence of symbols. For example, the decoding modules and/or circuits 2506 may be configured to convert the sequence of symbols into a transition number, where each digit of the transition number represents a transition between two consecutive symbols transmitted on the plurality of connectors, and to convert the transition number into a received plurality of bits. The processing circuit may include one or more modules and/or circuits 2504 configured for The processing circuit may include one or more error detection modules and/or circuits 2508 configured for determining whether a symbol error is present in the sequence of symbols based on a value of an EDC in the received plurality of bits. The processing circuit may include one or more error correction modules and/or circuits 2510 configured for correcting symbol errors. In one example, the error correction modules and/or circuits 2510 may be enabled when the symbol error is determined to be present in the sequence of symbols. The error correction modules and/or circuits 2510 may be configured to generate one or more permutations of the sequence of symbols, where each permutation including one symbol that is different from corresponding symbols in the sequence of symbols and different from corresponding symbols in other permutations, decode the one or more permutations, and identify a permutation that produces a decoded EDC value that matches an expected EDC value. The expected EDC value may correspond to a predefined value for EDCs transmitted over the multi-wire interface to enable detection of up to two symbol errors at the receiver.

[00134] Those of skill in the art would appreciate that the various illustrative logical blocks, modules, circuits, and algorithm steps described in connection with the embodiments disclosed herein may be implemented as electronic hardware, computer software, or combinations of both. To clearly illustrate this interchangeability of hardware and software, various illustrative components, blocks, modules, circuits, and steps have been described above generally in terms of their functionality. Whether such functionality is implemented as hardware or software depends upon the particular application and design constraints imposed on the overall system.

[00135] The various features of the invention described herein can be implemented in different systems without departing from the invention. It should be noted that the foregoing embodiments are merely examples and are not to be construed as limiting the invention. The description of the embodiments is intended to be illustrative, and not to limit the scope of the claims. As such, the present teachings can be readily

applied to other types of apparatuses and many alternatives, modifications, and variations will be apparent to those skilled in the art.

CLAIMS

1. A method of correcting transmission errors at a receiver in a multi-wire interface, comprising:

receiving a sequence of symbols representative of signaling state of a plurality of connectors;

decoding the sequence of symbols, including:

converting the sequence of symbols into a transition number, each digit of the transition number representing a transition between two consecutive symbols transmitted on the plurality of connectors; and

converting the transition number into a received plurality of bits;

determining whether a symbol error is present in the sequence of symbols based on a value of an error detection code (EDC) in the received plurality of bits; and

when the symbol error is determined to be present in the sequence of symbols:

generating one or more permutations of the sequence of symbols, each permutation including one symbol that is different from corresponding symbols in the sequence of symbols and different from corresponding symbols in other permutations;

decoding the one or more permutations; and

identifying a first permutation as a corrected sequence of symbols when it produces a decoded EDC value that matches an expected EDC value,

wherein the expected EDC value corresponds to a predefined value for EDCs transmitted over the multi-wire interface to enable detection of up to two symbol errors at the receiver.

2. The method of claim 1, further comprising:

substituting the first permutation for the sequence of symbols or substituting corrected data decoded from the first permutation for error data decoded from the sequence of symbols.

3. The method of claim 1, further comprising:

concurrently generating a set of permutations of the sequence of symbols in which all members of the set of permutations include modified versions of a same symbol in the sequence of symbols when the symbol error is determined to be present in the sequence of symbols; and

concurrently decoding all members of the set of permutations.

4. The method of claim 1, further comprising:

concurrently generating and decoding all possible permutations of the sequence of symbols that include a single symbol that is modified with respect to a corresponding symbol in the sequence of symbols when the symbol error is determined to be present in the sequence of symbols; and

concurrently comparing an EDC value decoded from each possible permutation of the sequence of symbols with the expected EDC value.

5. The method of claim 1, further comprising:

concurrently generating a set of permutations of the sequence of symbols in which members of the set of permutations include modified versions of different symbols in the sequence of symbols when the symbol error is determined to be present in the sequence of symbols; and

concurrently decoding all members of the set of permutations.

6. The method of claim 1, wherein the predefined value for EDCs transmitted over the multi-wire interface has a fixed length that is determined based on a total number of states per symbol defined for encoding data transmissions on the plurality of connectors.

7. The method of claim 6, wherein the EDC is a predefined number of least significant bits in the plurality of bits, and wherein the predefined value for EDCs transmitted over the multi-wire interface is determined based on a total number of symbols used to encode the plurality of bits.

8. The method of claim 1, wherein the symbol error causes a decoded version of the EDC to have a value that is different from the predefined value.

9. The method of claim 1, wherein the plurality of connectors comprises a number (N) of single-ended connectors, and a total number of states per symbol available for encoding data transmissions is $2^N - x$, wherein x is at least 1.

10. The method of claim 1, wherein the plurality of connectors comprises a number (N) of single-ended connectors, and a total number of states per symbol available for encoding data transmissions is $N!- x$, wherein x is at least 1, and wherein the one or more permutations of the sequence of symbols includes r permutations per symbol where r represents a number of possible transitions per symbol.

11. The method of claim 1, wherein the plurality of connectors comprises a number (N) of single-ended connectors, and a total number of states per symbol available for encoding data transmissions is $N!- x$, wherein x is at least 1, and wherein the one or more permutations of the sequence of symbols includes r-1 permutations per symbol where r represents a number of possible transitions per symbol.

12. The method of claim 1, wherein decoding the one or more permutations comprises:

refraining from decoding a permutation that includes two consecutive symbols having a same value.

13. An apparatus comprising:

means for receiving a sequence of symbols representative of signaling state of a plurality of connectors;

means for decoding the sequence of symbols and configured to:

convert the sequence of symbols into a transition number, each digit of the transition number representing a transition between two consecutive symbols transmitted on the plurality of connectors; and

convert the transition number into a received plurality of bits;

means for determining whether a symbol error is present in the sequence of symbols based on a value of an error detection code (EDC) in the received plurality of bits; and

means for correcting symbol errors wherein, when the symbol error is determined to be present in the sequence of symbols, the means for correcting symbol errors is configured to:

generate one or more permutations of the sequence of symbols, each permutation including one symbol that is different from corresponding symbols in the sequence of symbols and different from corresponding symbols in other permutations;

decode the one or more permutations; and

identify a permutation as a corrected sequence of symbols when it produces a decoded EDC value that matches an expected EDC value,

wherein the expected EDC value corresponds to a predefined value for EDCs transmitted over the plurality of connectors to enable detection of up to two symbol errors at the receiver.

14. The apparatus of claim 13, wherein the means for correcting the symbol error is configured to:

concurrently generate a set of permutations of the sequence of symbols in which all members of the set of permutations include modified versions of a same symbol in the sequence of symbols when the symbol error is determined to be present in the sequence of symbols; and

concurrently decode all members of the set of permutations.

15. The apparatus of claim 13, wherein the means for correcting the symbol error is configured to:

concurrently generate and decode all possible permutations of the sequence of symbols that include a single symbol that is modified with respect to a corresponding symbol in the sequence of symbols when the symbol error is determined to be present in the sequence of symbols; and

concurrently compare an EDC value decoded from each possible permutation of the sequence of symbols with the expected EDC value.

16. The apparatus of claim 13, wherein the means for correcting the symbol error is configured to:

concurrently generate a set of permutations of the sequence of symbols in which members of the set of permutations include modified versions of different symbols in the sequence of symbols when the symbol error is determined to be present in the sequence of symbols; and

concurrently decode all members of the set of permutations.

17. An apparatus comprising:

a communications transceiver configured to provide a sequence of symbols representative of signaling state of a plurality of connectors;

a decoder circuit configured to convert a transition number representative of transitions between consecutive symbols in the sequence of symbols into a plurality of bits;

an error detection circuit configured to determine whether symbol errors are present in the sequence of symbols based on a value of an error detection code (EDC) in the plurality of bits; and

an error correction circuit responsive to a determination that a symbol error is present in the sequence of symbols and configured to:

generate one or more permutations of the sequence of symbols, each permutation including one symbol that is different from corresponding symbols in the sequence of symbols and different from corresponding symbols in other permutations;

decode the one or more permutations; and

identify a first permutation as a corrected sequence of symbols when it produces a decoded EDC value that matches an expected EDC value,

wherein the expected EDC value corresponds to a predefined value for EDCs transmitted over the plurality of connectors to enable detection of up to two symbol errors, and

wherein the symbol error causes a decoded version of the EDC to have a value that is different from the predefined value.

18. The apparatus of claim 17, wherein the error correction circuit is configured to:

substitute the first permutation for the sequence of symbols or substitute corrected data decoded from the first permutation for error data decoded from the sequence of symbols; and

generate a permutation for each possible value of one symbol in the sequence of symbols.

19. The apparatus of claim 17, wherein the error correction circuit is configured to:

concurrently generate a set of permutations of the sequence of symbols in which all members of the set of permutations include modified versions of a same symbol in the sequence of symbols when the symbol error is determined to be present in the sequence of symbols; and

concurrently decode all members of the set of permutations.

20. The apparatus of claim 17, wherein the error correction circuit is configured to:

concurrently generate and decode all possible permutations of the sequence of symbols that include a single symbol that is modified with respect to a corresponding symbol in the sequence of symbols when the symbol error is determined to be present in the sequence of symbols; and

identify the permutation that produces the decoded EDC value that matches the expected EDC value by concurrently comparing an EDC value decoded from each possible permutation of the sequence of symbols with the expected EDC value.

21. The apparatus of claim 17, wherein the error correction circuit is configured to:

concurrently generate a set of permutations of the sequence of symbols in which members of the set of permutations include modified versions of different symbols in the sequence of symbols different when the symbol error is determined to be present in the sequence of symbols; and

concurrently decode all members of the set of permutations.

22. The apparatus of claim 17, wherein the predefined value for EDCs transmitted over the plurality of connectors has a fixed length that is determined based on a total number of states per symbol defined for encoding data transmissions on the plurality of connectors.

23. The apparatus of claim 17, wherein the EDC is a predefined number of least significant bits in the plurality of bits, and wherein the predefined value for EDCs transmitted over the plurality of connectors is determined based on a total number of symbols used to encode the plurality of bits.

24. The apparatus of claim 17, wherein the error correction circuit is configured to generate one or more permutations of the sequence of symbols by:

generating a permutation for each possible change in value of one symbol in the sequence of symbols.

25. The apparatus of claim 17, wherein the error correction circuit is configured to generate one or more permutations of the sequence of symbols by:

generating a permutation corresponding to each symbol in the sequence of symbols, wherein a symbol is modified by a same value in each permutation.

26. The apparatus of claim 17, wherein the plurality of connectors comprises a number (*N*) of single-ended connectors, and a total number of states per symbol available for encoding data transmissions is $2^N - x$ where x is at least 1, or a total number of states per symbol available for encoding data transmissions is *N*!- x where x is at least 1.

27. A processor readable storage medium having code executable by the processor stored thereon, the code comprising instructions for:

receiving a sequence of symbols representative of signaling state of a plurality of connectors;

decoding the sequence of symbols, including:

converting the sequence of symbols into a transition number, each digit of the transition number representing a transition between two consecutive symbols transmitted on the plurality of connectors; and

converting the transition number into a received plurality of bits;

determining whether a symbol error is present in the sequence of symbols based on a value of an error detection code (EDC) in the received plurality of bits; and

when the symbol error is determined to be present in the sequence of symbols:

generating one or more permutations of the sequence of symbols, each permutation including one symbol that is different from corresponding symbols in the sequence of symbols and different from corresponding symbols in other permutations;

decoding the one or more permutations; and

identifying a permutation as a corrected sequence of symbols when it produces a decoded EDC value that matches an expected EDC value,

wherein the expected EDC value corresponds to a predefined value for EDCs transmitted over the plurality of connectors to enable detection of up to two symbol errors.

28. The storage medium of claim 27, wherein the code includes instructions for:

concurrently generating a set of permutations of the sequence of symbols in which all members of the set of permutations include modified versions of a same symbol in the sequence of symbols when the symbol error is determined to be present in the sequence of symbols; and

concurrently decoding all members of the set of permutations.

29. The storage medium of claim 27, wherein the code includes instructions for:

concurrently generating and decoding all possible permutations of the sequence of symbols that include a single symbol that is modified with respect to a corresponding symbol in the sequence of symbols when the symbol error is determined to be present in the sequence of symbols; and

concurrently comparing an EDC value decoded from each possible permutation of the sequence of symbols with the expected EDC value.

30. The storage medium of claim 27, wherein the code includes instructions for:

concurrently generating a set of permutations of the sequence of symbols in which members of the set of permutations include modified versions of different symbols in the sequence of symbols different when the symbol error is determined to be present in the sequence of symbols; and

concurrently decoding all members of the set of permutations.

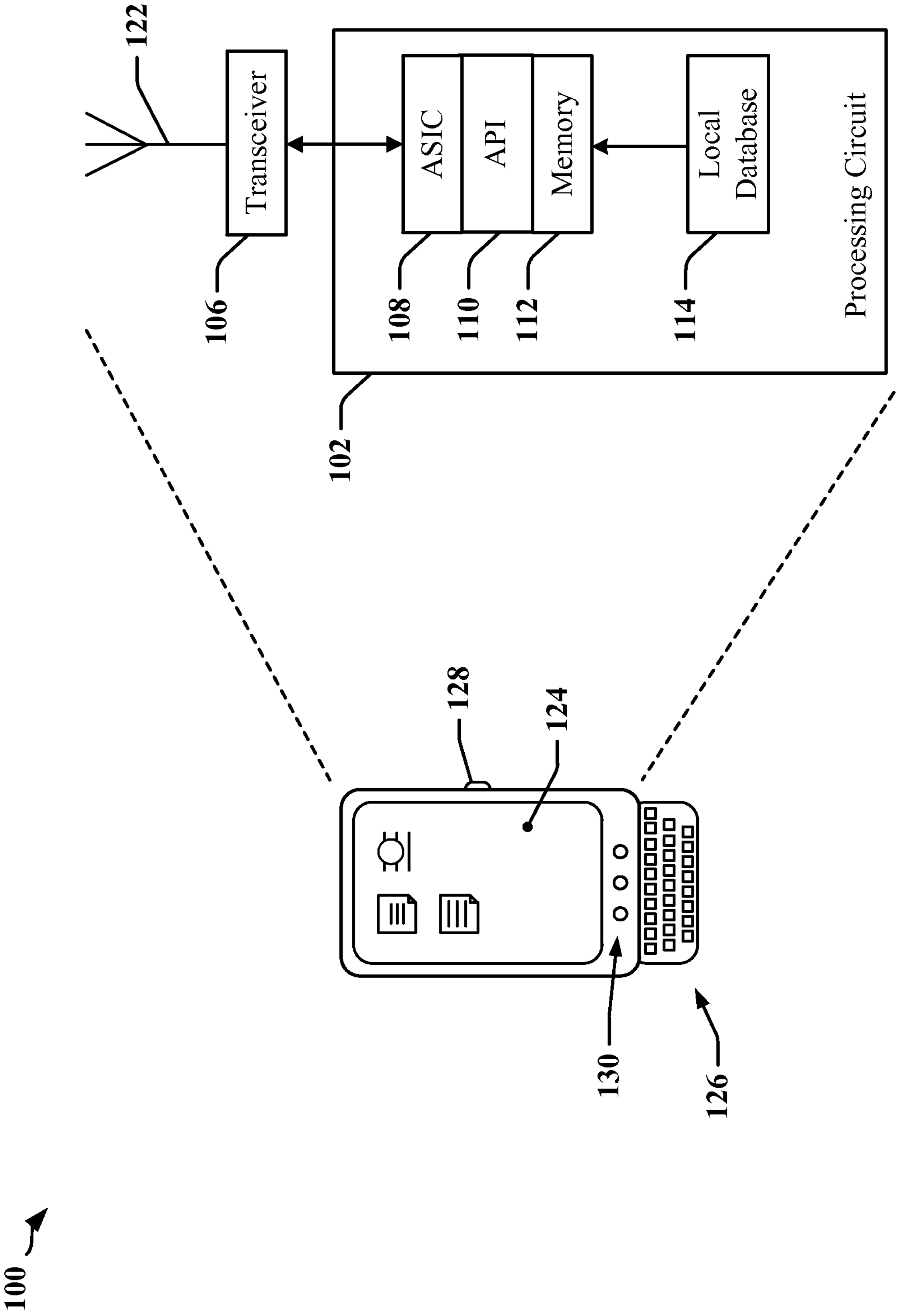

100
122
Transceiver
106
ASIC
API
Memory
108
110
112
Local Database
114
Processing Circuit
102
128
124
130
126

FIG. 1

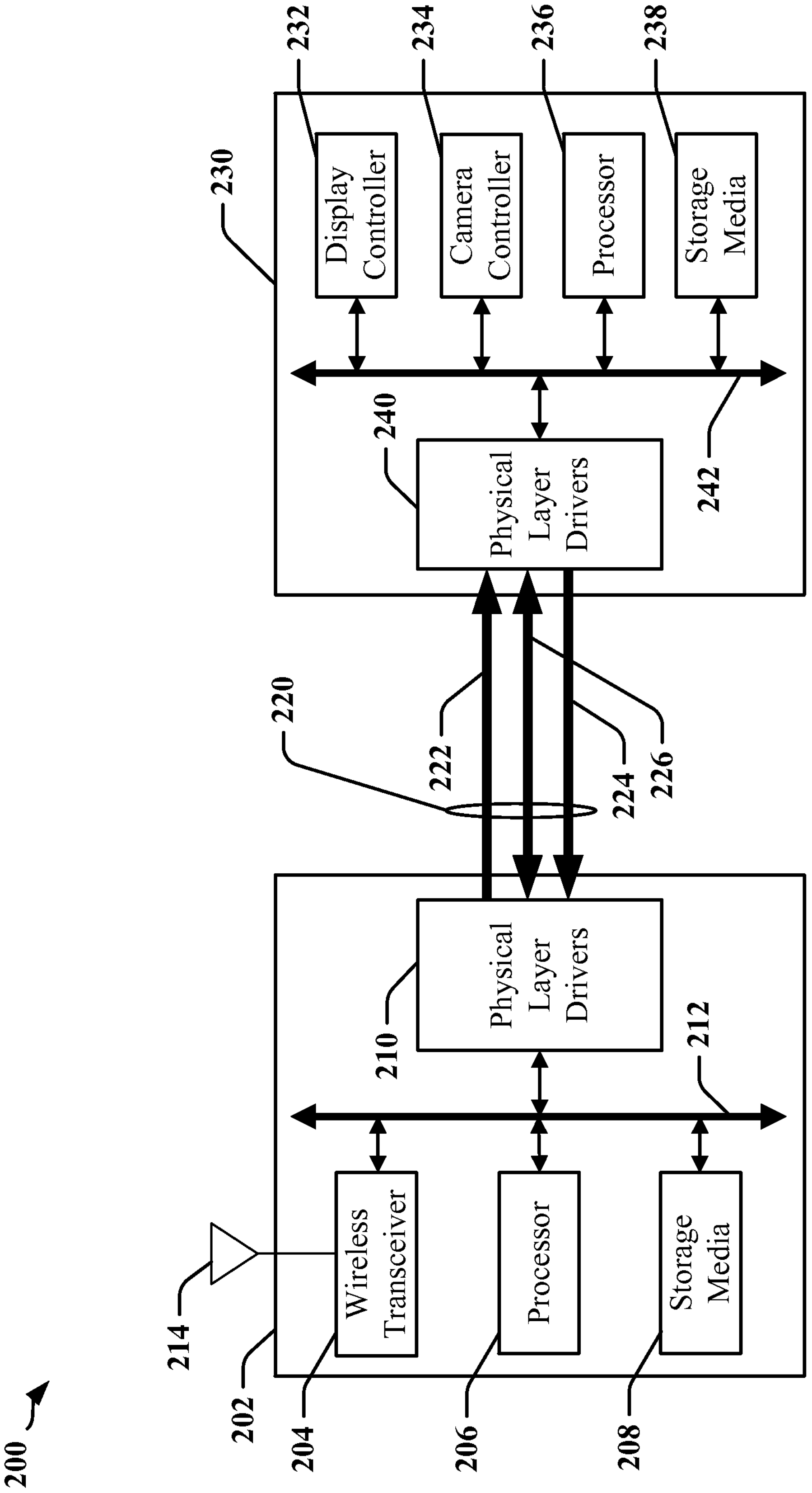

200
202
214
204
Wireless Transceiver
206
Processor
208
Storage Media
212
210
Physical Layer Drivers
220
222
224
226
230
240
Physical Layer Drivers
242
232
Display Controller
234
Camera Controller
236
Processor
238
Storage Media

FIG. 2

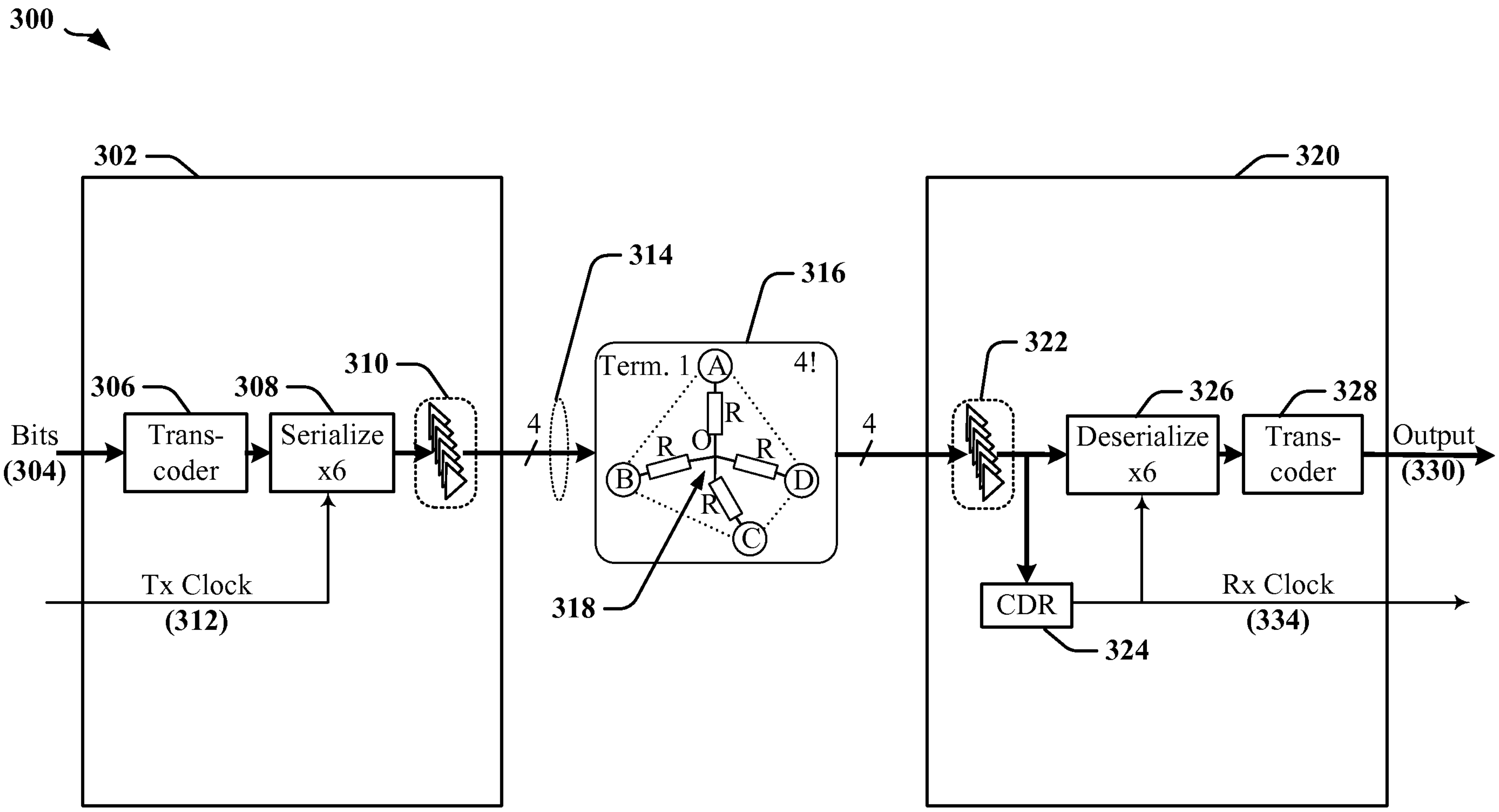

300
302
320
314
316
322
306
308
310
326
328
Bits
(304)
Trans-
coder
Serialize
x6
4
Term. 1
A
4!
R
O
R
R
B
D
R
C
4
Deserialize
x6
Trans-
coder
Output
(330)
Tx Clock
(312)
318
CDR
Rx Clock
(334)
324

FIG. 3

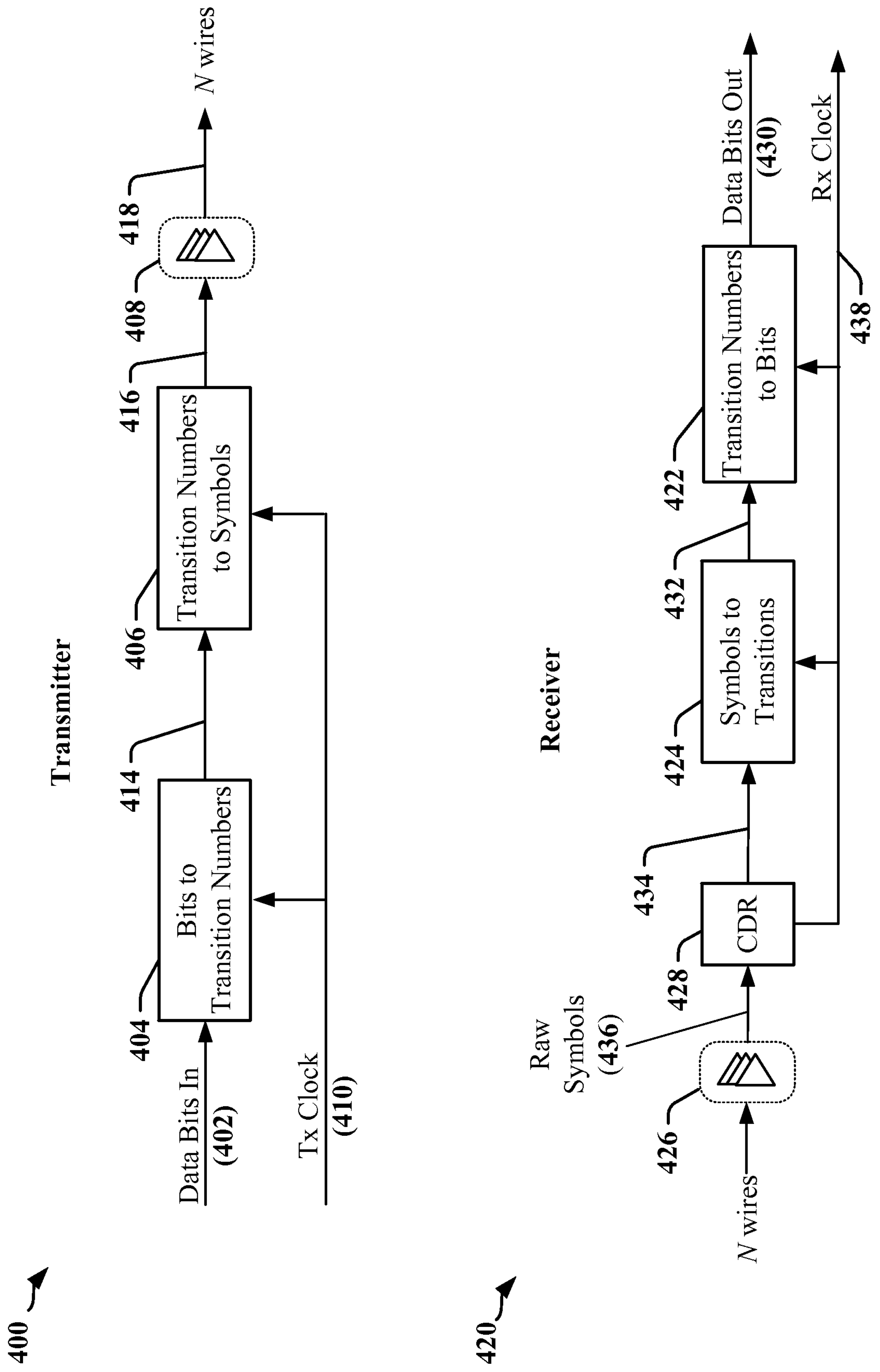
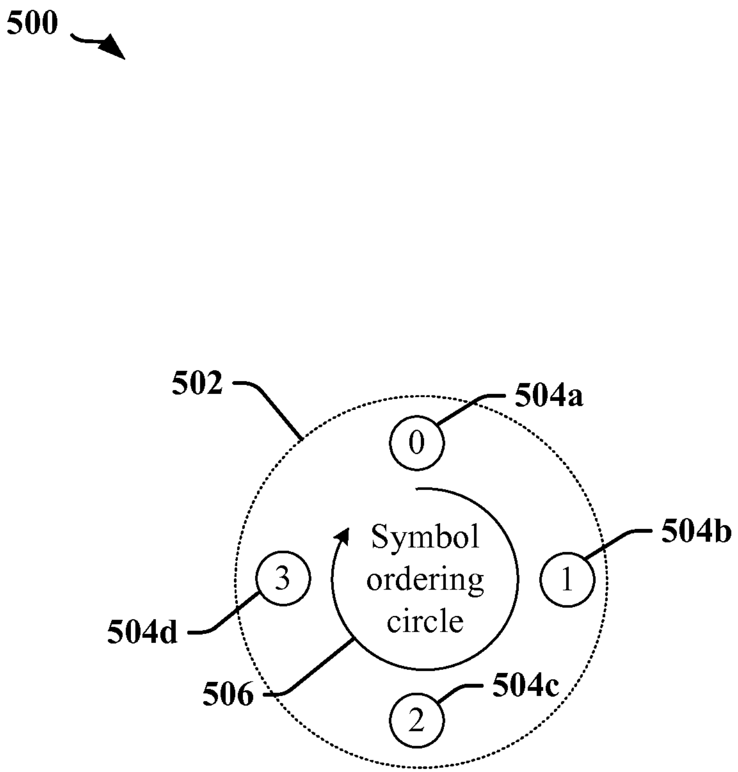

400
Transmitter
Data Bits In
(402)
404
Bits to
Transition Numbers
414
406
Transition Numbers
to Symbols
416
408
418
N wires
Tx Clock
(410)
420
Receiver
N wires
426
Raw
Symbols
(436)
428
CDR
434
424
Symbols to
Transitions
432
422
Transition Numbers
to Bits
Data Bits Out
(430)
Rx Clock
438

FIG. 4



520

	Previous Symbol Ps (522)	Current Symbol Cs (524)	Transition Number T (526)
530	0	1	1
		2	2
		3	0
532	1	2	1
		3	2
		0	0
534	2	3	1
		0	2
		1	0
536	3	0	1
		1	2
		2	0

FIG. 5

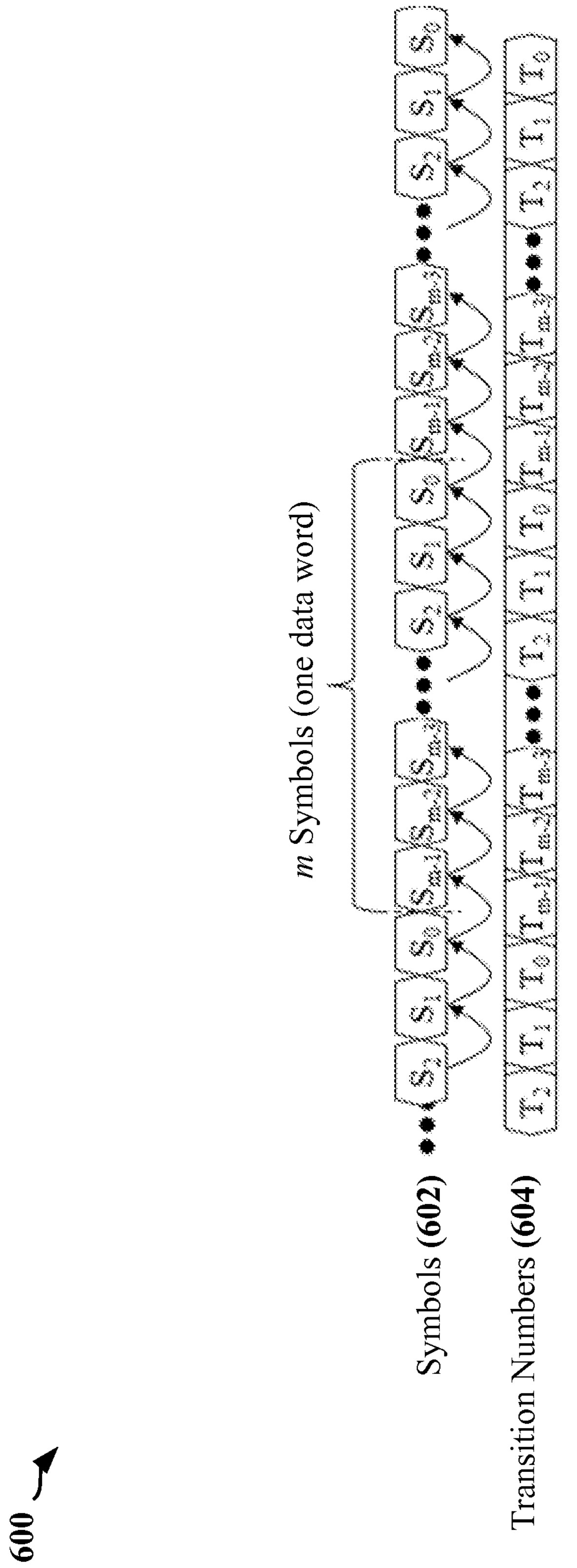

600
Symbols (602)
Transition Numbers (604)
m Symbols (one data word)
S2
S1
S0
Sm-1
Sm-2
Sm-3
T2
T1
T0
Tm-1
Tm-2
Tm-3

FIG. 6

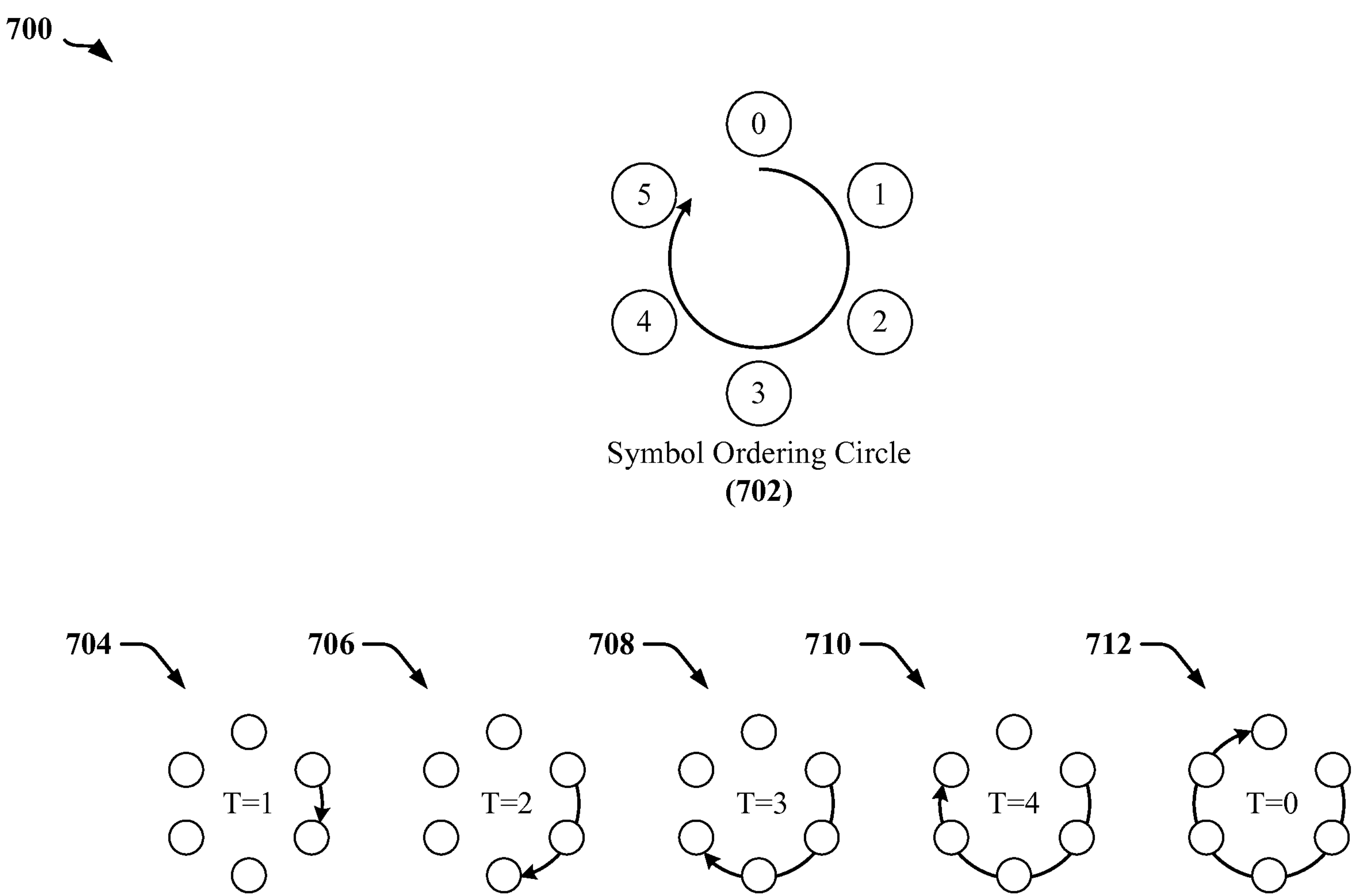

700
0
1
2
3
4
5
Symbol Ordering Circle
(702)
704
706
708
710
712
T=1
T=2
T=3
T=4
T=0

FIG. 7

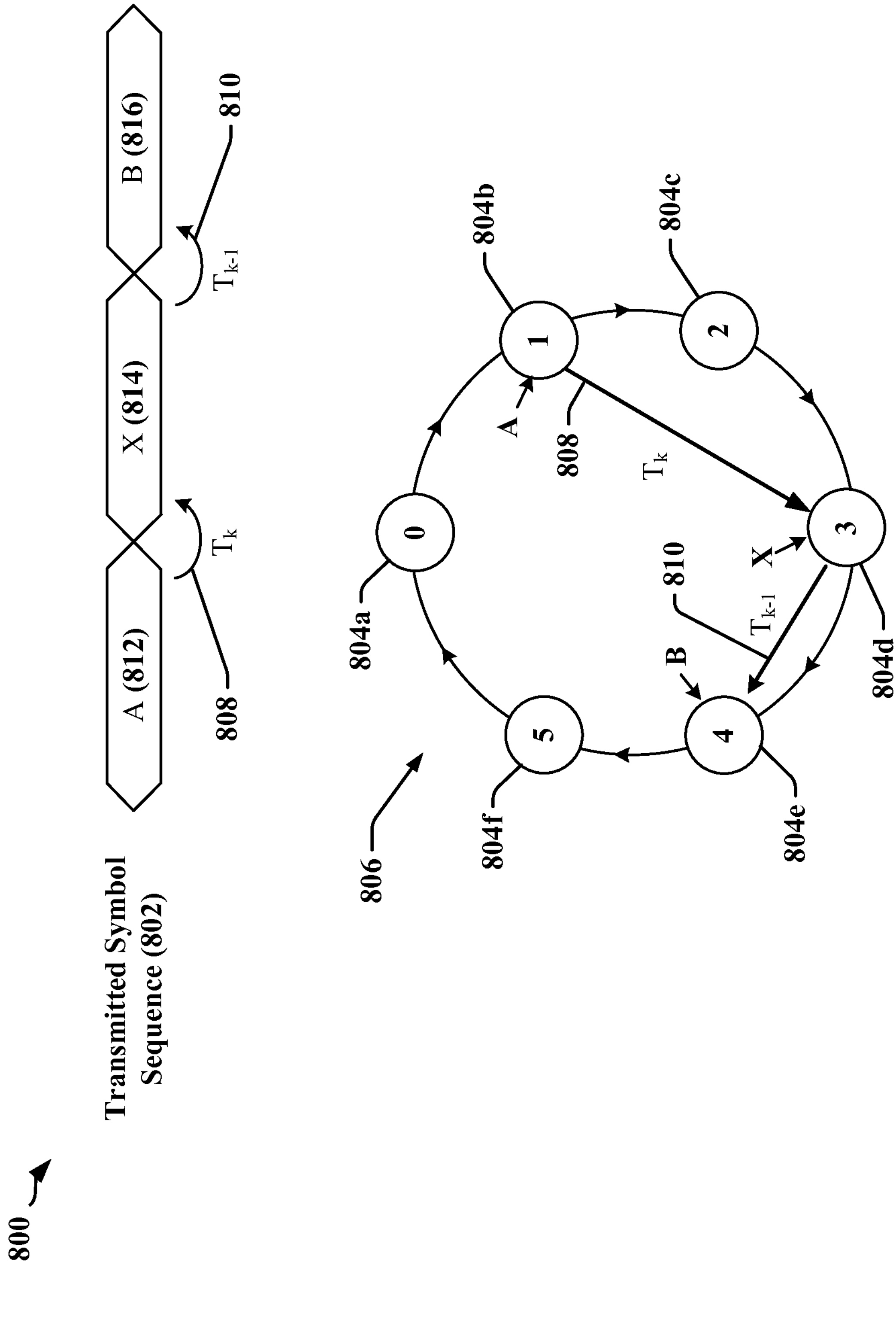

800
Transmitted Symbol Sequence (802)
A (812)
X (814)
B (816)
808
810
T_k
T_{k-1}
806
0
1
2
3
4
5
804a
804b
804c
804d
804e
804f
A
X
B
808
810
T_k
T_{k-1}

FIG. 8

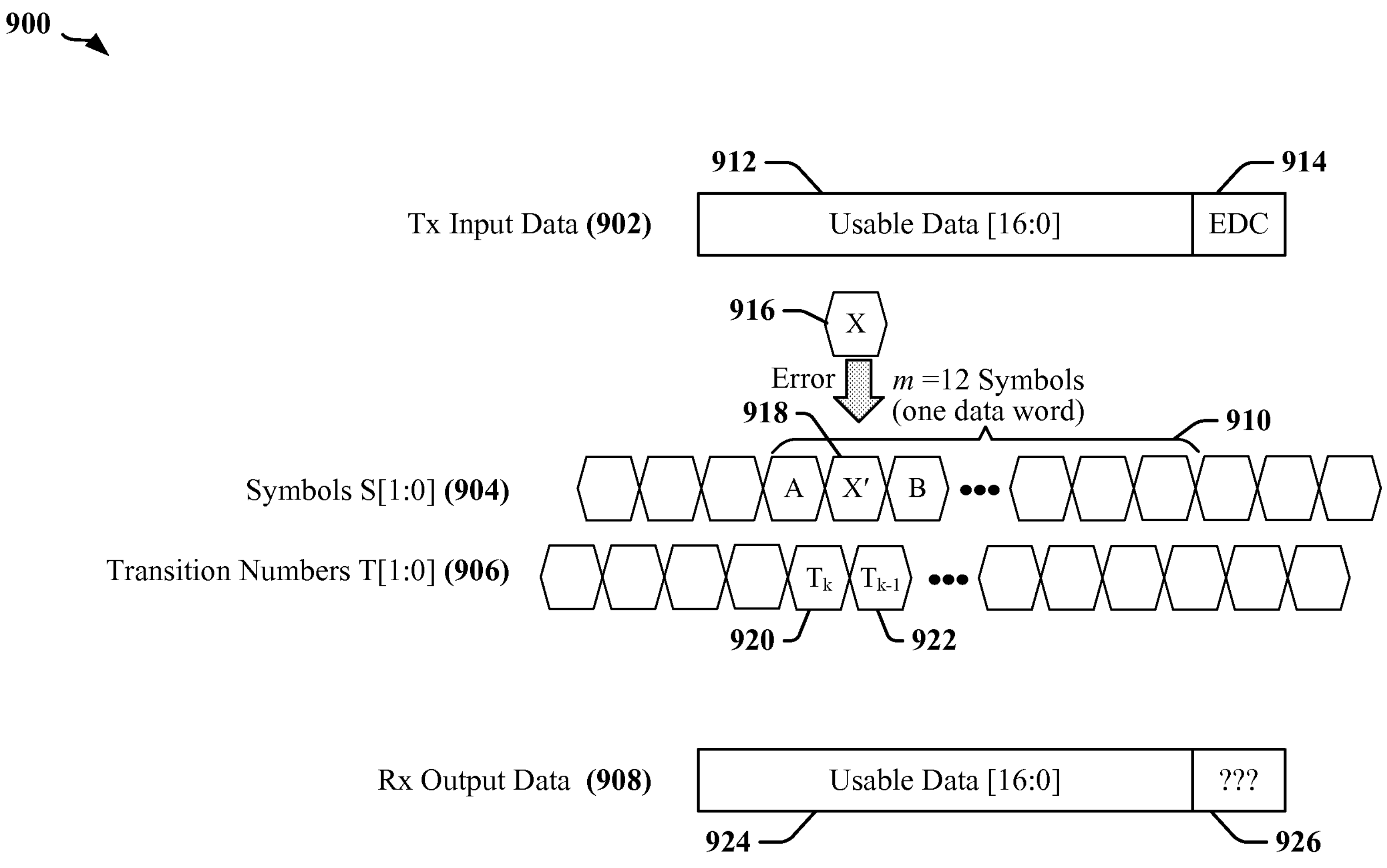

900
912
914
Tx Input Data (902)
Usable Data [16:0]
EDC
916
X
Error
m =12 Symbols
(one data word)
918
910
Symbols S[1:0] (904)
A
X′
B
Transition Numbers T[1:0] (906)
Tk
Tk-1
920
922
Rx Output Data (908)
Usable Data [16:0]
???
924
926

FIG. 9

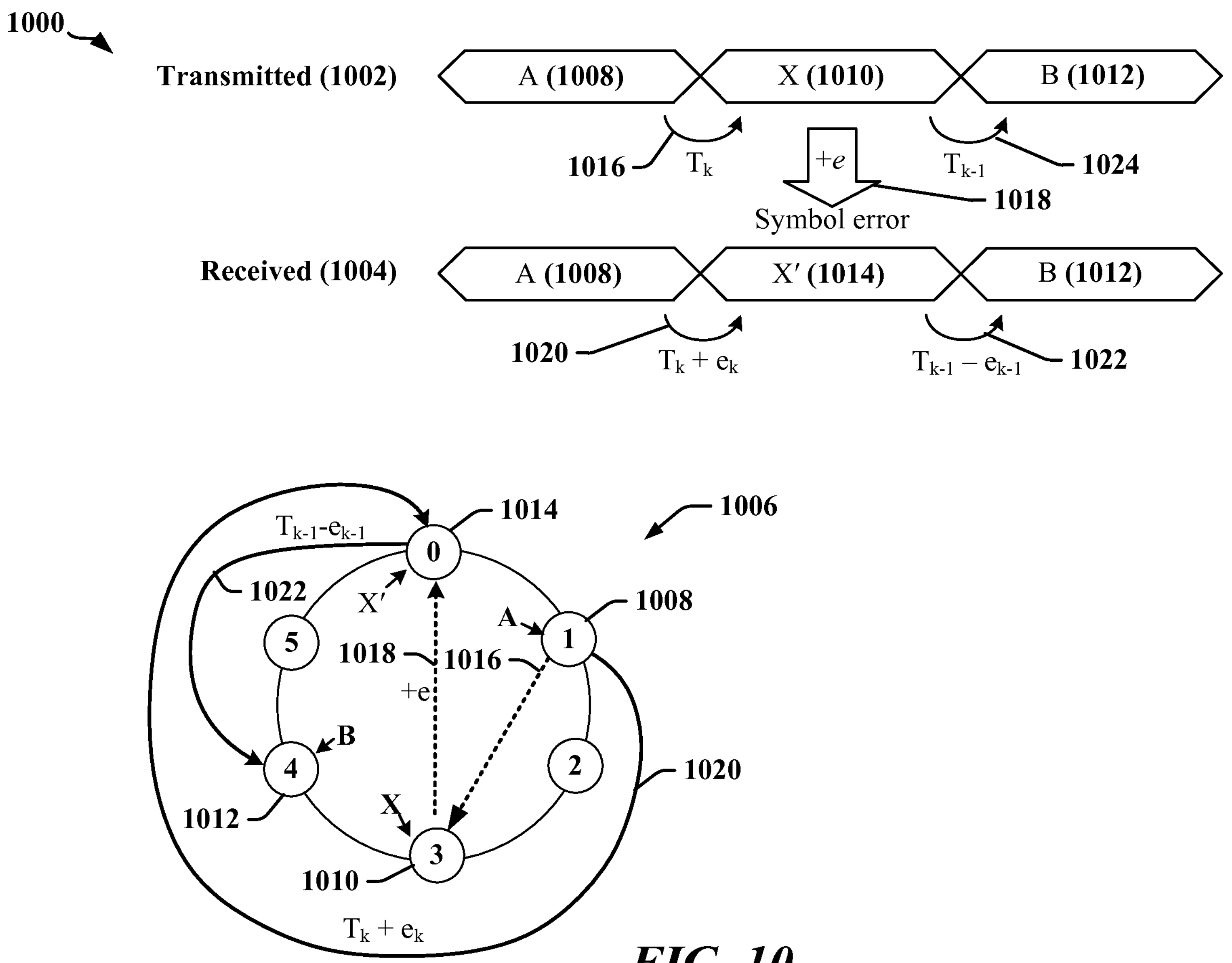

1000
Transmitted (1002)
A (1008)
X (1010)
B (1012)
1016
T_k
+e
T_{k-1}
1024
1018
Symbol error
Received (1004)
A (1008)
X′ (1014)
B (1012)
1020
$T_k + e_k$
$T_{k-1} - e_{k-1}$
1022
1006
1014
T_{k-1}-e_{k-1}
0
1022
X′
1008
A
1
5
1018
1016
+e
B
4
2
1020
1012
X
3
1010
$T_k + e_k$

FIG. 10

1100

	r = 3	
n	r^n hex	r^n bin
0	000001	000000000000000000000001
1	000003	000000000000000000000011
2	000009	000000000000000000001001
3	00001B	000000000000000000011011
4	000051	000000000000000001010001
5	0000F3	000000000000000011110011
6	0002D9	000000000000001011011001
7	00088B	000000000000100010001011
8	0019A1	000000000001100110100001
9	004CE3	000000000100110011100011
10	00E6A9	000000001110011010101001
11	02B3FB	000000101011001111111011
12	081BF1	000010000001101111110001
13	1853D3	000110000101001111010011
14	48FB79	010010001111101101111001
15	DAF26B	110110101111001001101011

1104

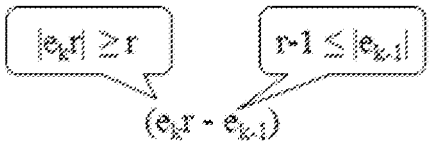
1102

	r = 5	
n	r^n hex	r^n bin
0	000000001	000000000000000000000000000000000001
1	000000005	000000000000000000000000000000000101
2	000000019	000000000000000000000000000000011001
3	00000007D	000000000000000000000000000001111101
4	000000271	000000000000000000000000001001110001
5	000000C35	000000000000000000000000110000110101
6	000003D09	000000000000000000000011110100001001
7	00001312D	000000000000000000010011000100101101
8	00005F5E1	000000000000000001011111010111100001
9	0001DCD65	000000000000000111011100110101100101
10	0009502F9	000000000000100101010000001011111001
11	002E90EDD	000000000010111010010000111011011101
12	00E8D4A51	000000001110100011010100101001010001
13	048C27395	000001001000110000100111001110010101
14	16BCC41E9	000101101011110011000100000111101001
15	71AFD498D	011100011010111111010100100110001101

1106

FIG. 11

1200



Error coefficient $(e_k r - e_{k-1})$ never becomes 0

Dec	Hex	Bin	$\{e^k, e^{k-1}\}$ list	
-21	EB	11101011	-4, 1	
-18	EE	11101110	-3, 3	
-17	EF	11101111	-3, 2	
-16	F0	11110000	-4, -4	-3, 1
-13	F3	11110011	-2, 3	-3, -2
-12	F4	11110100	-3, -3	-2, 2
-9	F7	11110111	-1, 4	-2, -1
-8	F8	11111000	-2, -2	-1, 3
-7	F9	11111001	-2, -3	
-4	FC	11111100	-1, -1	
-3	FD	11111101	-1, -2	
3	03	00000011	1, 2	
4	04	00000100	1, 1	
7	07	00000111	2, 3	
8	08	00001000	2, 2	1, -3
9	09	00001001	1, -4	2, 1
13	0D	00001101	2, -3	3, 2
16	10	00010000	4, 4	3, -1
17	11	00010001	3, -2	
18	12	00010010	3, -3	
21	15	00010101	4, -1	

FIG. 12

1300

$(2^n r - 2^n) = 2^n (r-1)$

Shift r-1 left by n-1 digits

Replace LSB of r with 0
e.g.)
$r=3=00011_2 \rightarrow r\text{-}1=00010_2$
$r=5=00101_2 \rightarrow r\text{-}1=00100_2$
$r=7=00111_2 \rightarrow r\text{-}1=00110_2$
$r=15=01111_2 \rightarrow r\text{-}1=01110_2$
$r=23=10111_2 \rightarrow r\text{-}1=10110_2$

r		r-1	Longest 2^n for e_k or e_{k-1}		Longest non-zero LSB of $(e_k r - e_{k-1})$
3	00011_2	00010_2	2	00010_2	000100_2
5	00101_2	00100_2	4	00100_2	0010000_2
7	00111_2	00110_2	4	00100_2	0011000_2
9	01001_2	01000_2	8	01000_2	01000000_2
11	01011_2	01010_2	8	01000_2	01010000_2
15	01111_2	01110_2	8	01000_2	01110000_2
23	10111_2	10110_2	16	10000_2	101100000_2

Dec	Hex	Bin	$\{e^k, e^{k-1}\}$ list	
-21	EB	11101011	-4, 1	
-18	EE	11101110	-3, 3	
-17	EF	11101111	-3, 2	
-16	F0	11110000	-4, -4	-3, 1
-13	F3	11110011	-2, 3	-3, -2
-12	F4	11110100	-3, -3	-2, 2
-9	F7	11110111	-1, 4	-2, -1
-8	F8	11111000	-2, -2	-1, 3
-7	F9	11111001	-2, -3	
-4	FC	11111100	-1, -1	
-3	FD	11111101	-1, -2	
3	03	00000011	1, 2	
4	04	00000100	1, 1	
7	07	00000111	2, 3	
8	08	00001000	2, 2	1, -3
9	09	00001001	1, -4	2, 1
13	0D	00001101	2, -3	3, 2
16	10	00010000	4, 4	3, -1
17	11	00010001	3, -2	
18	12	00010010	3, -3	
21	15	00010101	4, -1	

FIG. 13

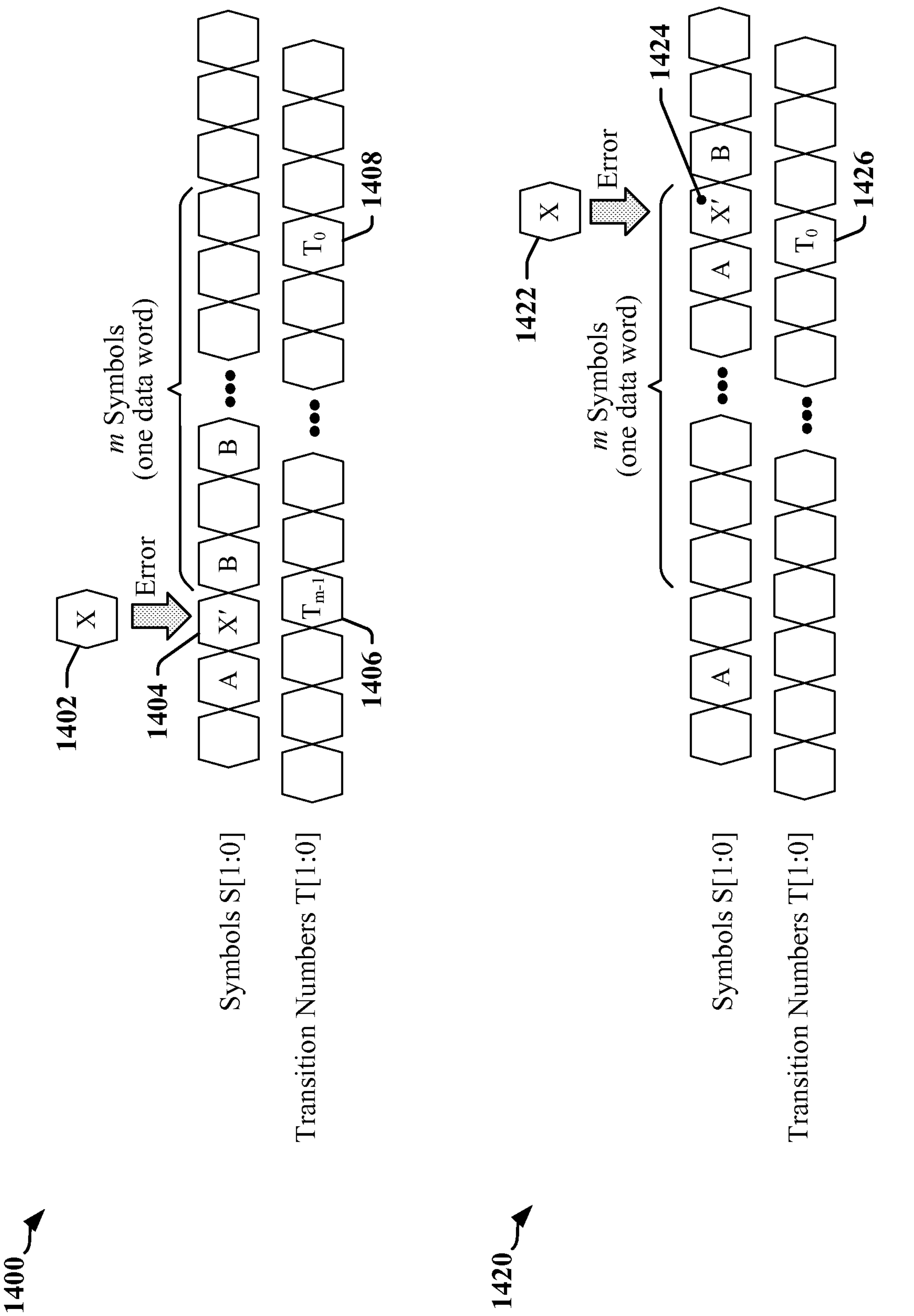

1400
Symbols S[1:0]
Transition Numbers T[1:0]
1402
X
1404
Error
A
X'
B
B
T_{m-1}
1406
T_0
1408
m Symbols
(one data word)
1420
Symbols S[1:0]
Transition Numbers T[1:0]
1422
X
Error
1424
A
A
X'
B
T_0
1426
m Symbols
(one data word)

FIG. 14

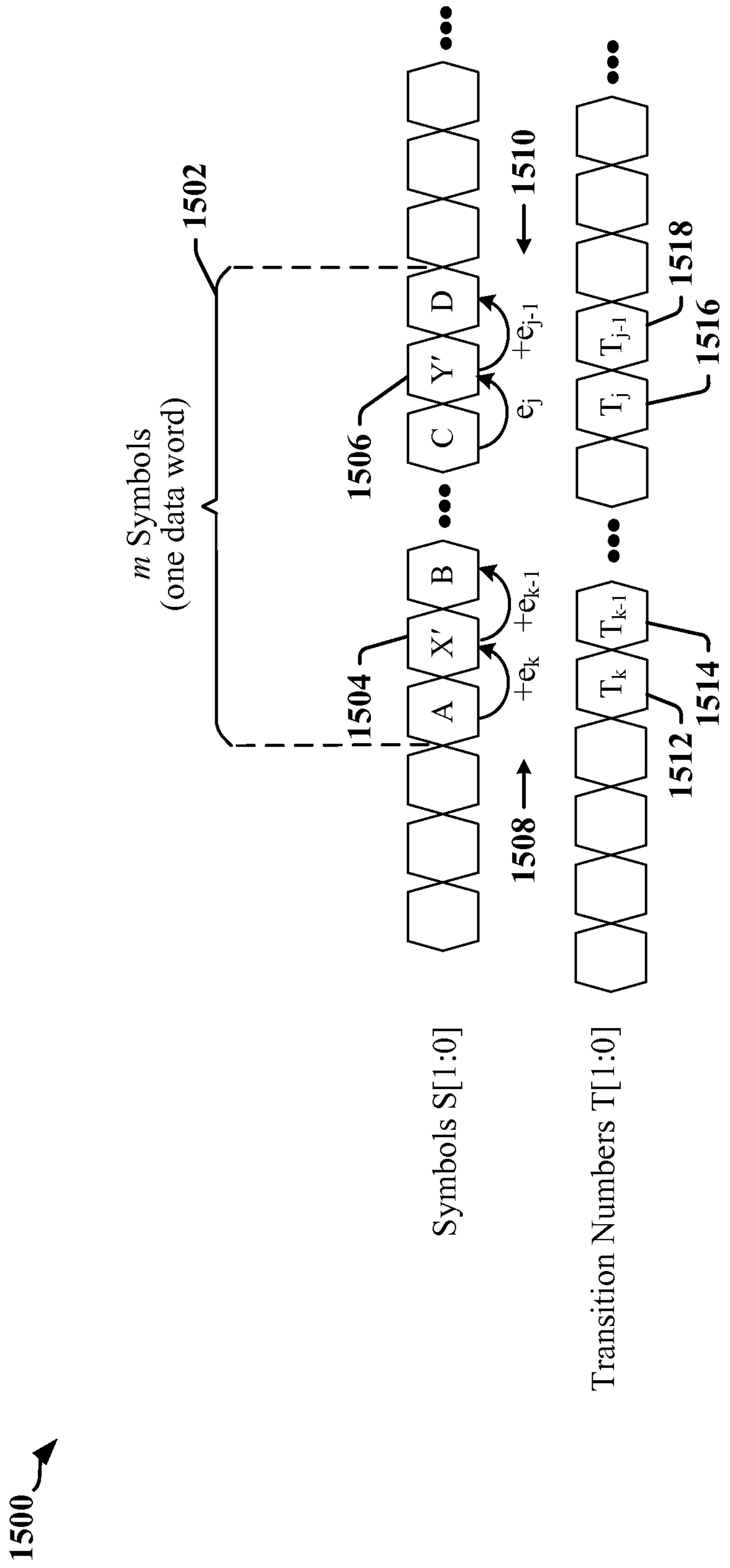

1500
Symbols S[1:0]
Transition Numbers T[1:0]
m Symbols
(one data word)
1502
1504
1506
1508
1510
1512
1514
1516
1518
A
X'
B
C
Y'
D
+e_k
+e_k-1
e_j
+e_j-1
T_k
T_k-1
T_j
T_j-1

FIG. 15

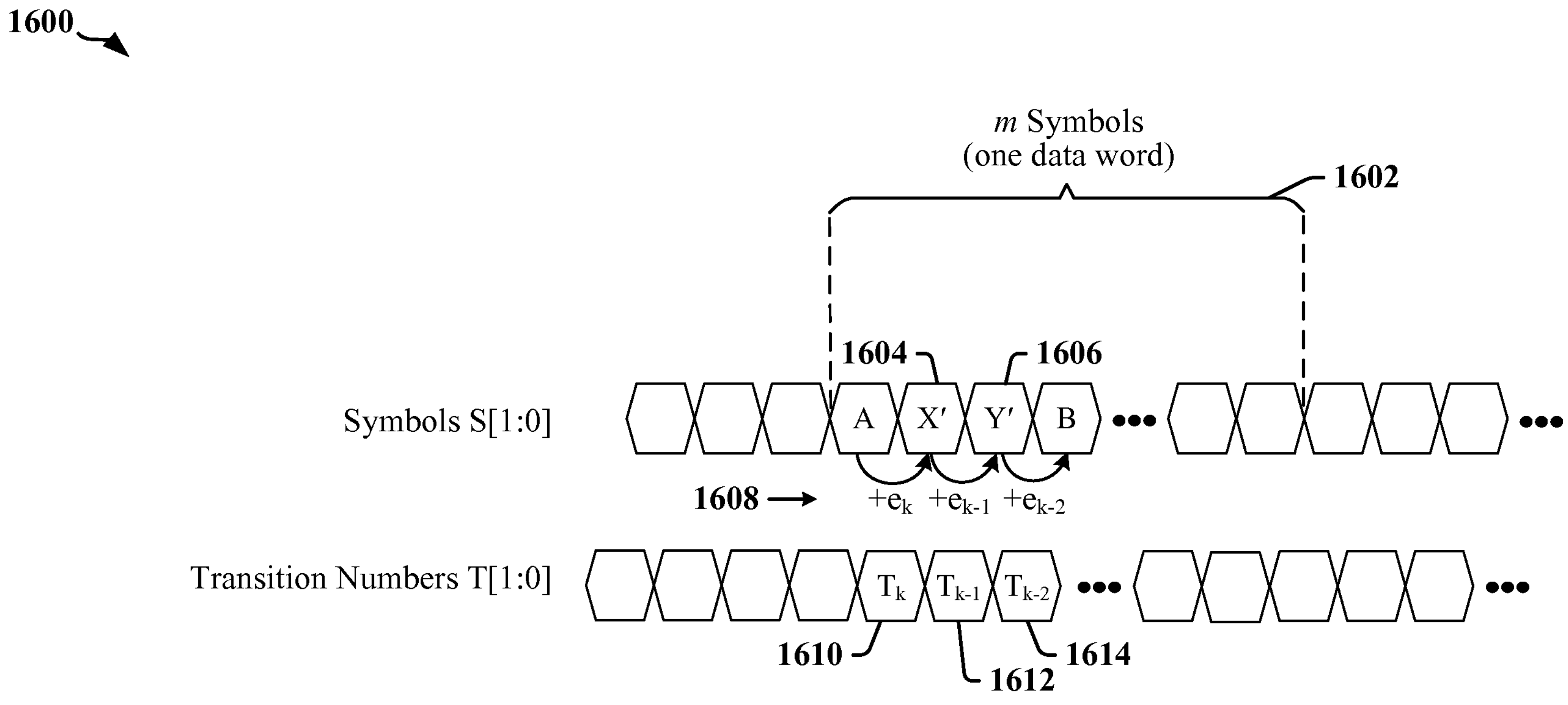
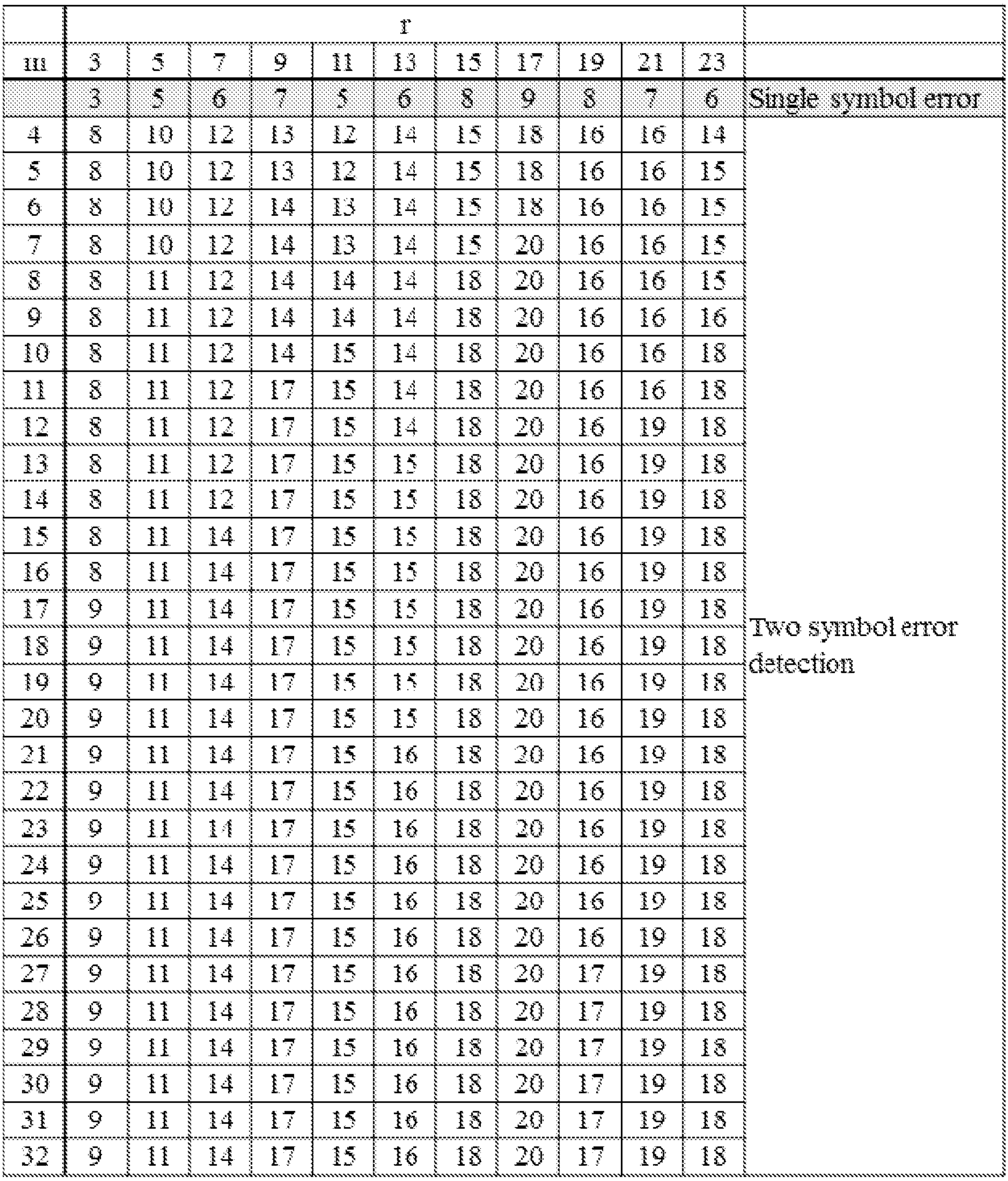

1600
m Symbols
(one data word)
1602
1604
1606
Symbols S[1:0]
A
X′
Y′
B
1608
+e_k
+e_k-1
+e_k-2
Transition Numbers T[1:0]
T_k
T_k-1
T_k-2
1610
1612
1614

FIG. 16

1700



	r											
m	3	5	7	9	11	13	15	17	19	21	23	
	3	5	6	7	5	6	8	9	8	7	6	Single symbol error
4	8	10	12	13	12	14	15	18	16	16	14	Two symbol error detection
5	8	10	12	13	12	14	15	18	16	16	15	
6	8	10	12	14	13	14	15	18	16	16	15	
7	8	10	12	14	13	14	15	20	16	16	15	
8	8	11	12	14	14	14	18	20	16	16	15	
9	8	11	12	14	14	14	18	20	16	16	16	
10	8	11	12	14	15	14	18	20	16	16	18	
11	8	11	12	17	15	14	18	20	16	16	18	
12	8	11	12	17	15	14	18	20	16	19	18	
13	8	11	12	17	15	15	18	20	16	19	18	
14	8	11	12	17	15	15	18	20	16	19	18	
15	8	11	14	17	15	15	18	20	16	19	18	
16	8	11	14	17	15	15	18	20	16	19	18	
17	9	11	14	17	15	15	18	20	16	19	18	
18	9	11	14	17	15	15	18	20	16	19	18	
19	9	11	14	17	15	15	18	20	16	19	18	
20	9	11	14	17	15	15	18	20	16	19	18	
21	9	11	14	17	15	16	18	20	16	19	18	
22	9	11	14	17	15	16	18	20	16	19	18	
23	9	11	14	17	15	16	18	20	16	19	18	
24	9	11	14	17	15	16	18	20	16	19	18	
25	9	11	14	17	15	16	18	20	16	19	18	
26	9	11	14	17	15	16	18	20	16	19	18	
27	9	11	14	17	15	16	18	20	17	19	18	
28	9	11	14	17	15	16	18	20	17	19	18	
29	9	11	14	17	15	16	18	20	17	19	18	
30	9	11	14	17	15	16	18	20	17	19	18	
31	9	11	14	17	15	16	18	20	17	19	18	
32	9	11	14	17	15	16	18	20	17	19	18	

FIG. 17

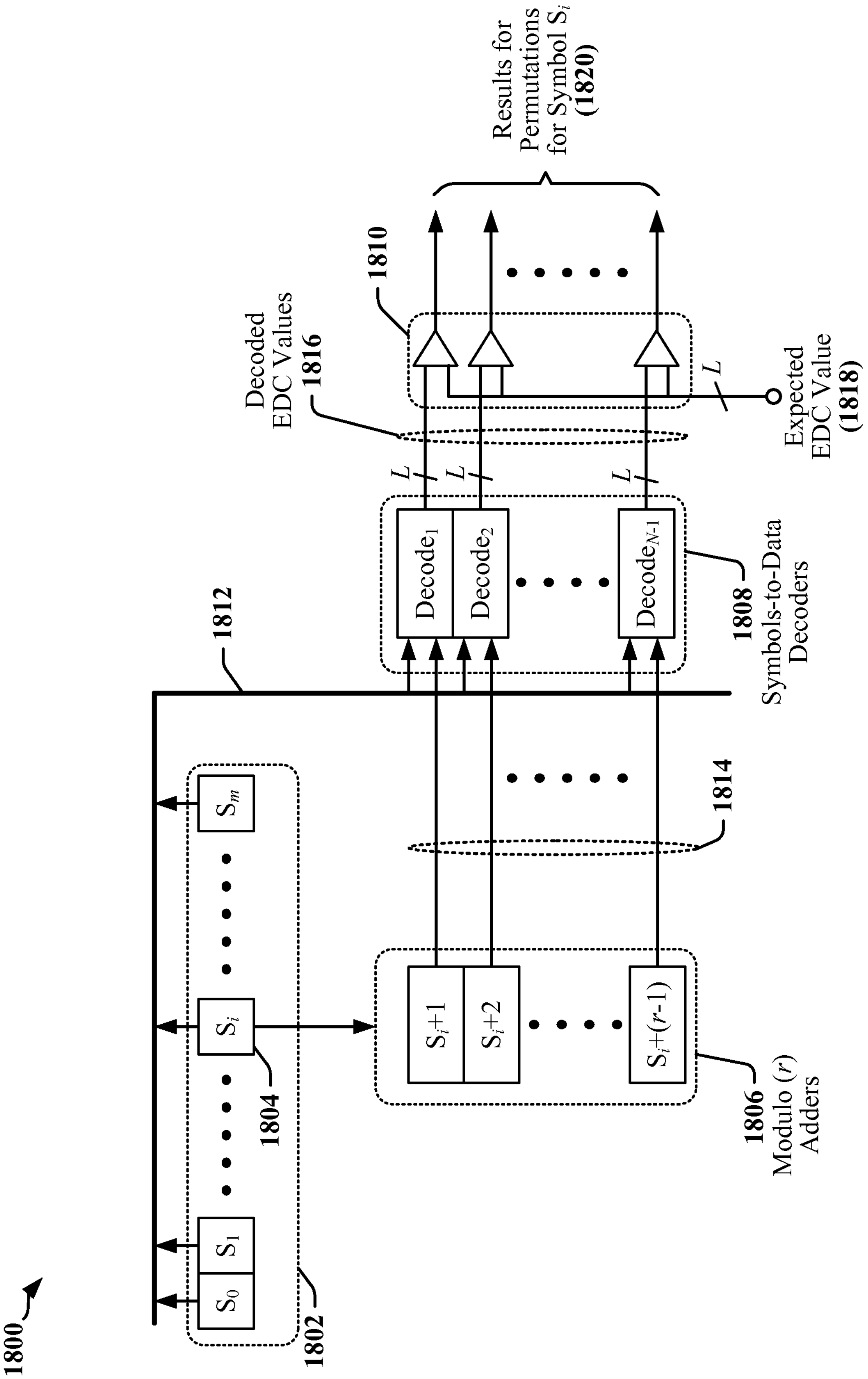

1800
1802
S0
S1
Si
Sm
1804
1812
Si+1
Si+2
Si+(r-1)
1806
Modulo (r) Adders
1814
Decode1
Decode2
DecodeN-1
1808
Symbols-to-Data Decoders
L
L
L
Decoded EDC Values
1816
1810
L
Expected EDC Value
(1818)
Results for Permutations for Symbol Si
(1820)

FIG. 18

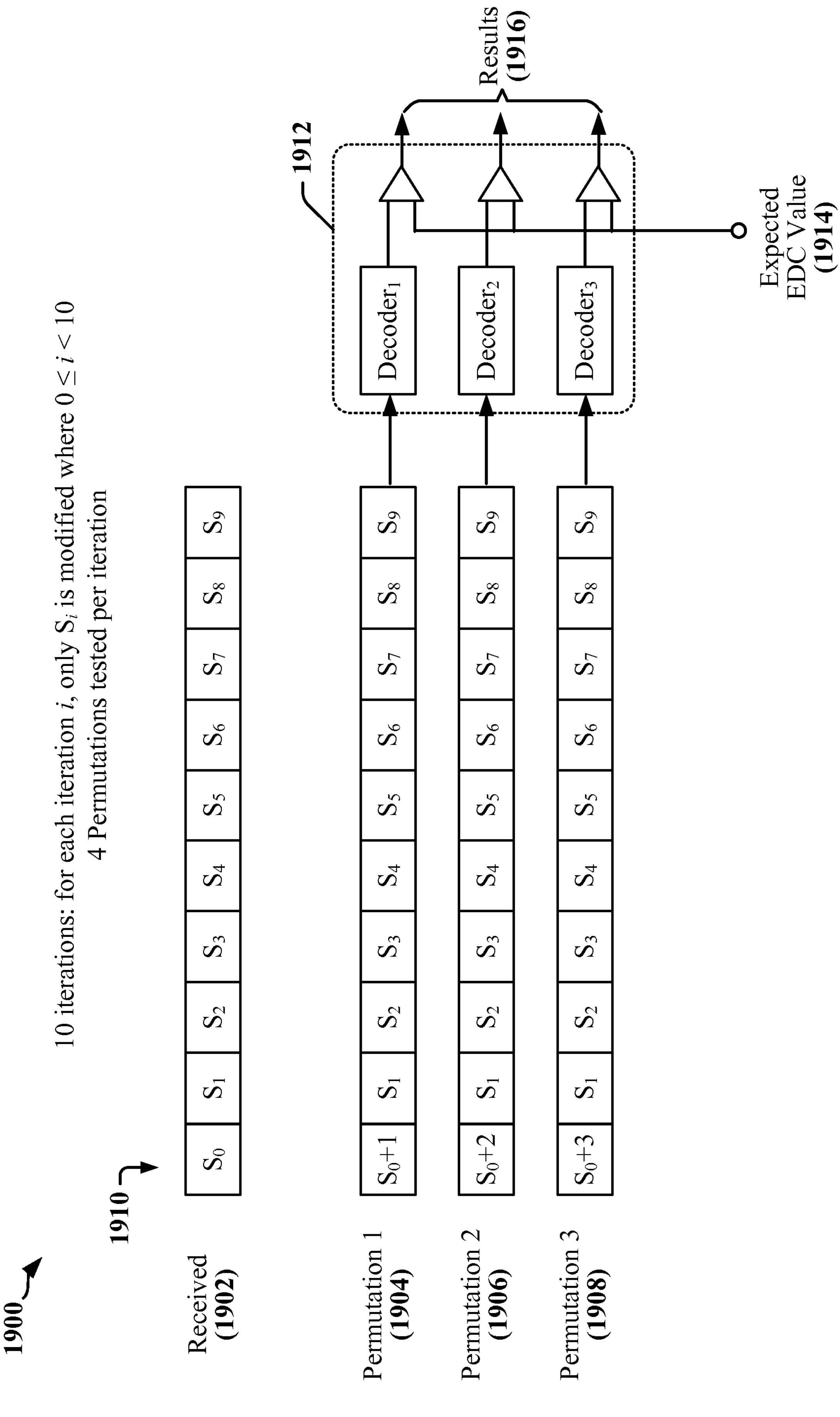
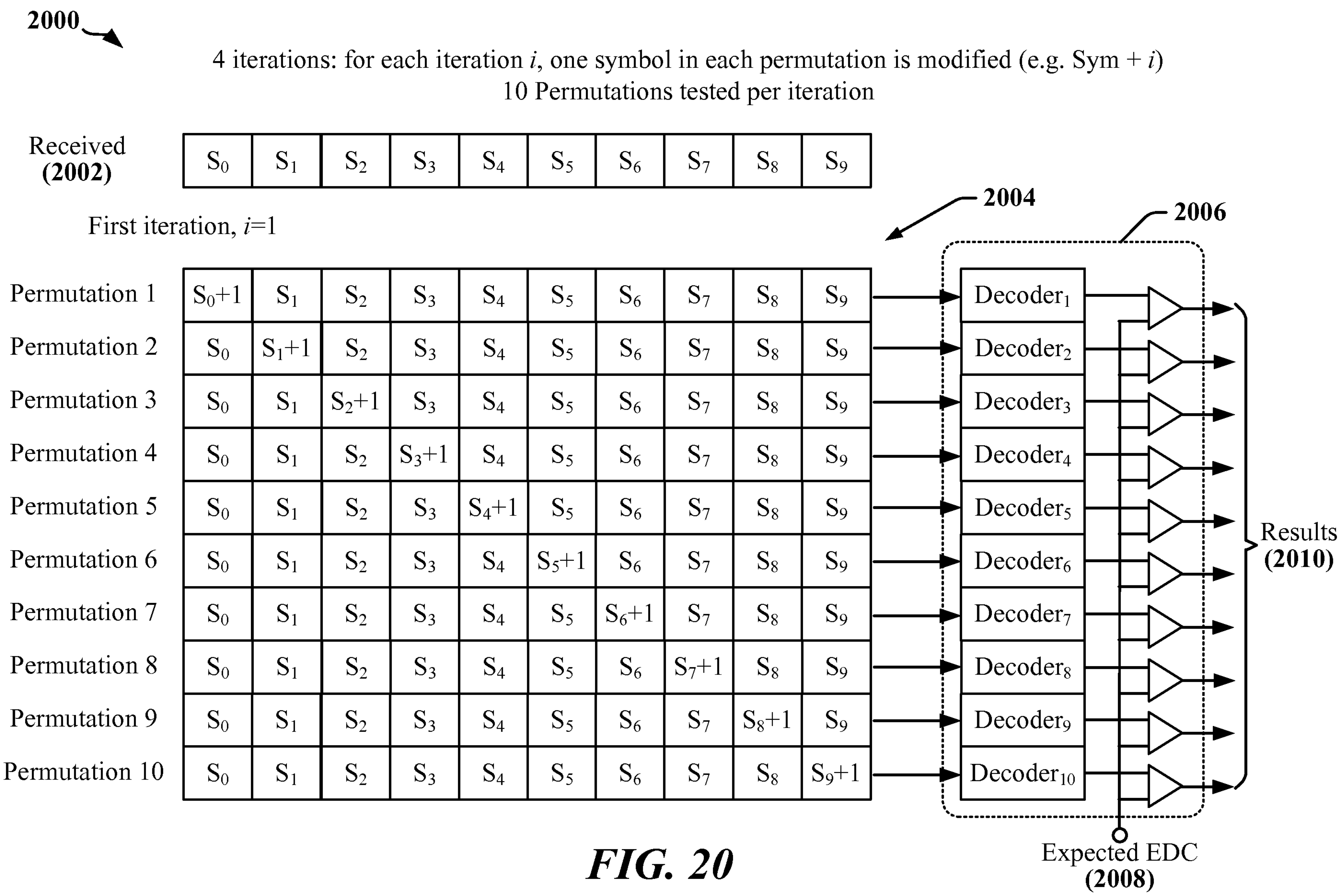
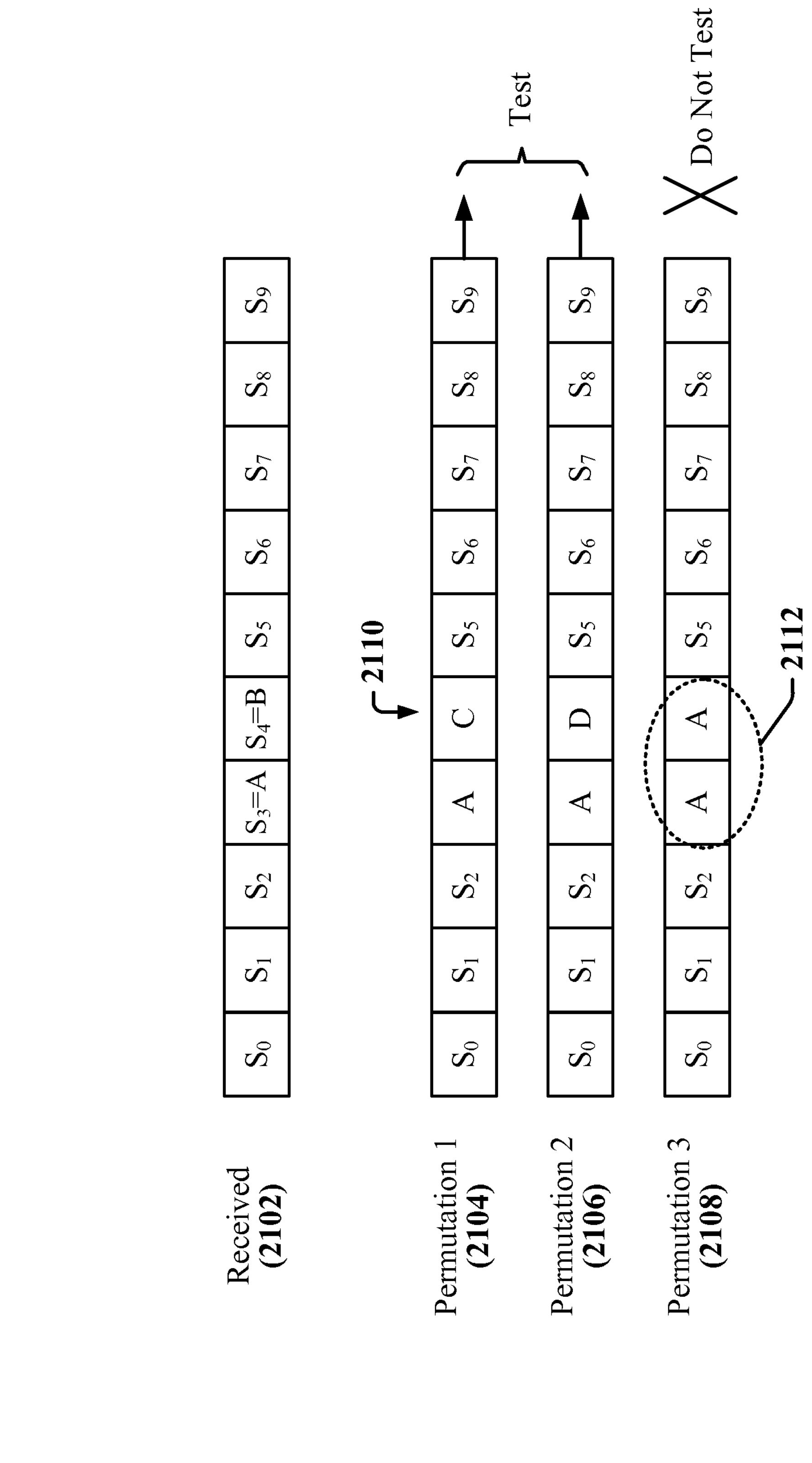

1900
10 iterations: for each iteration i, only S_i is modified where $0 \leq i < 10$
4 Permutations tested per iteration
1910
Received (1902)
S_0 S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_8 S_9
Permutation 1 (1904)
S_0+1 S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_8 S_9
Permutation 2 (1906)
S_0+2 S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_8 S_9
Permutation 3 (1908)
S_0+3 S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_8 S_9
1912
Decoder_1
Decoder_2
Decoder_3
Expected EDC Value (1914)
Results (1916)

FIG. 19



Received (2002)	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9
Permutation 1	S_0+1	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9
Permutation 2	S_0	S_1+1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9
Permutation 3	S_0	S_1	S_2+1	S_3	S_4	S_5	S_6	S_7	S_8	S_9
Permutation 4	S_0	S_1	S_2	S_3+1	S_4	S_5	S_6	S_7	S_8	S_9
Permutation 5	S_0	S_1	S_2	S_3	S_4+1	S_5	S_6	S_7	S_8	S_9
Permutation 6	S_0	S_1	S_2	S_3	S_4	S_5+1	S_6	S_7	S_8	S_9
Permutation 7	S_0	S_1	S_2	S_3	S_4	S_5	S_6+1	S_7	S_8	S_9
Permutation 8	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7+1	S_8	S_9
Permutation 9	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8+1	S_9
Permutation 10	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9+1

FIG. 20



2100
Received
(2102)
S_0
S_1
S_2
S_3=A
S_4=B
S_5
S_6
S_7
S_8
S_9
2110
Permutation 1
(2104)
S_0
S_1
S_2
A
C
S_5
S_6
S_7
S_8
S_9
Permutation 2
(2106)
S_0
S_1
S_2
A
D
S_5
S_6
S_7
S_8
S_9
Test
Permutation 3
(2108)
S_0
S_1
S_2
A
A
S_5
S_6
S_7
S_8
S_9
2112
Do Not Test

FIG. 21

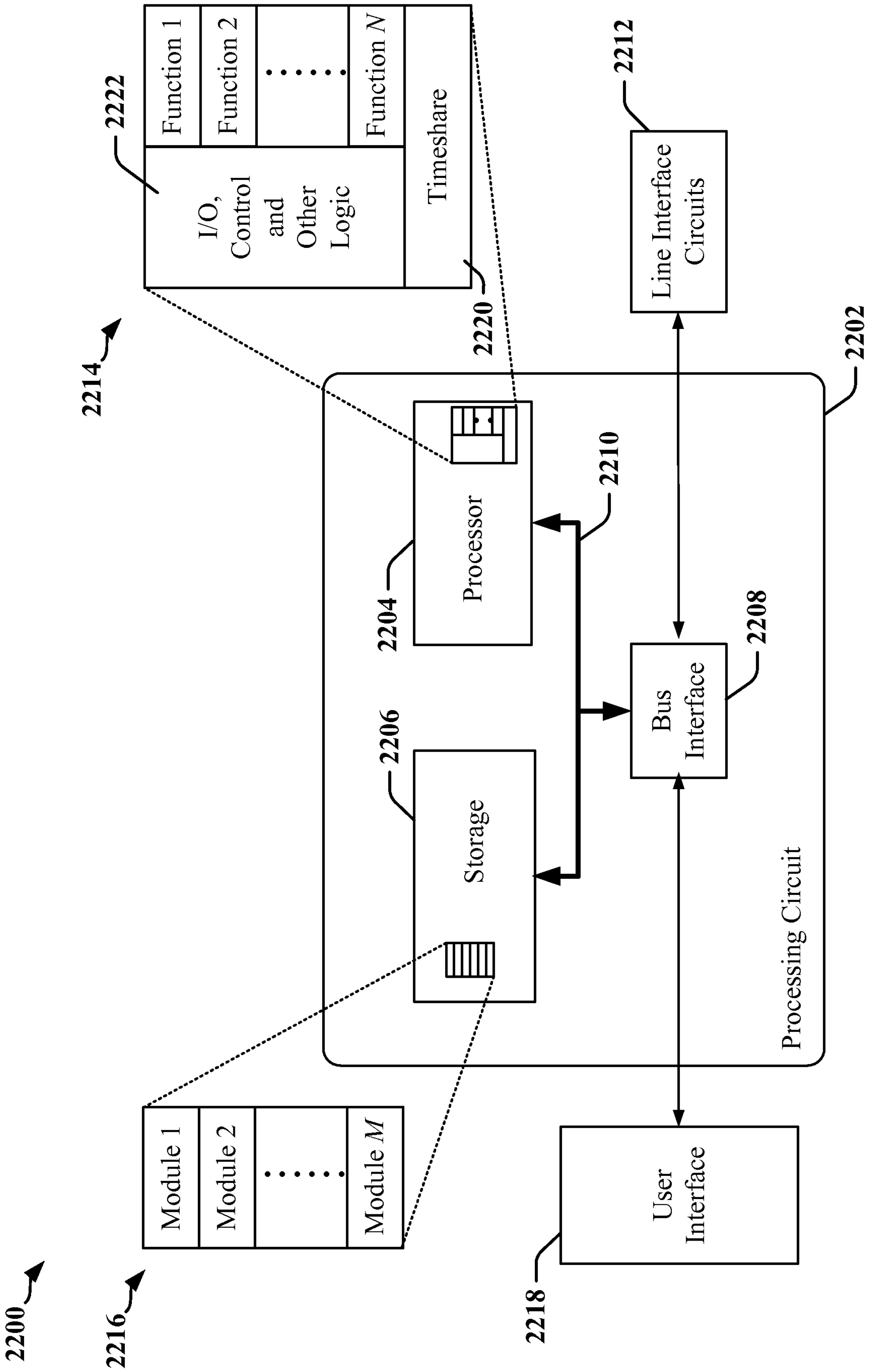

2200
2216
Module 1
Module 2
Module M
2218
User Interface
Storage
2206
Processing Circuit
Bus Interface
2208
2204
Processor
2210
2202
Line Interface Circuits
2212
2214
2220
I/O, Control and Other Logic
2222
Function 1
Function 2
Function N
Timeshare

FIG. 22

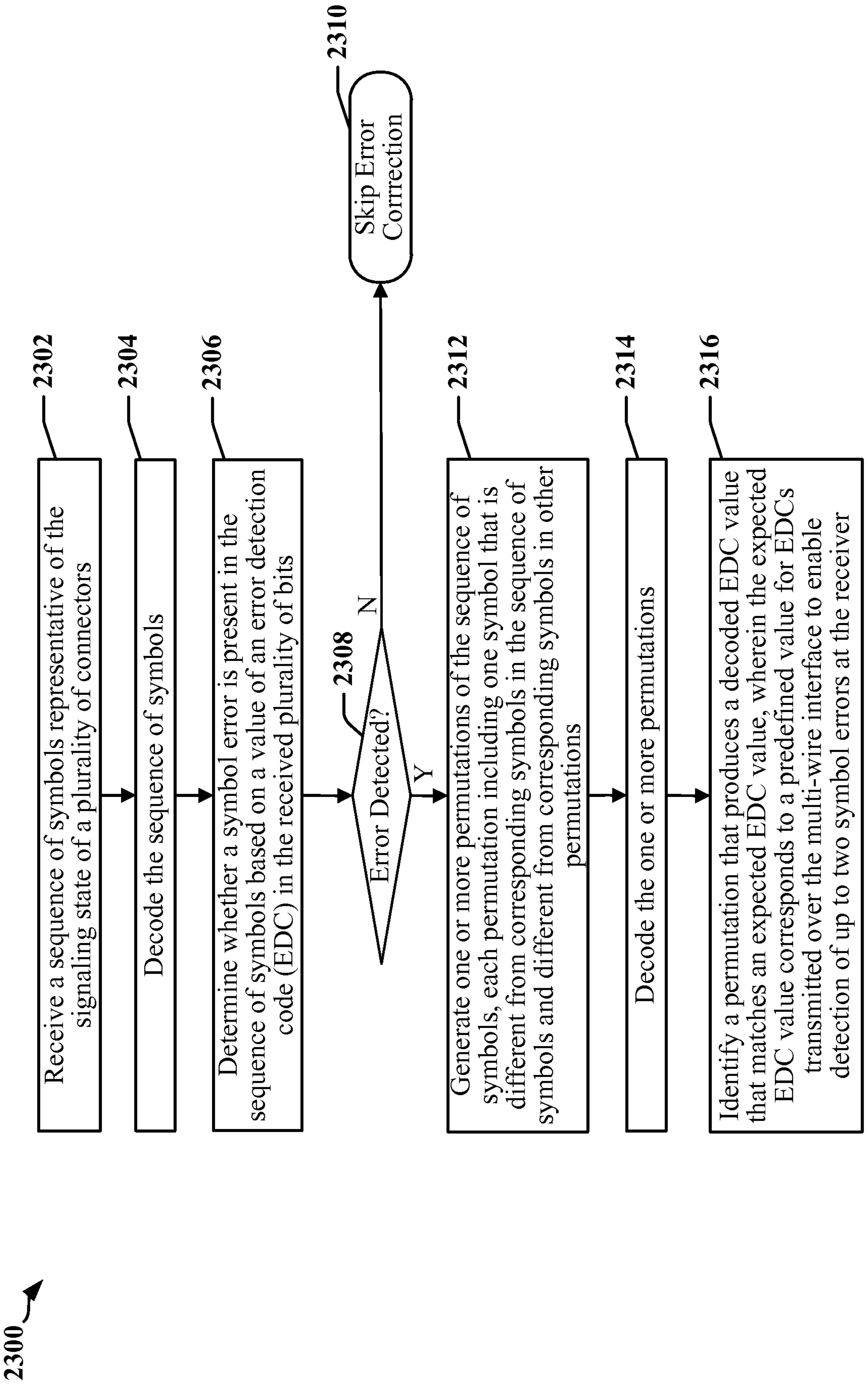

2300
2302
Receive a sequence of symbols representative of the signaling state of a plurality of connectors
2304
Decode the sequence of symbols
2306
Determine whether a symbol error is present in the sequence of symbols based on a value of an error detection code (EDC) in the received plurality of bits
2308
Error Detected?
N
Y
2310
Skip Error Correction
2312
Generate one or more permutations of the sequence of symbols, each permutation including one symbol that is different from corresponding symbols in the sequence of symbols and different from corresponding symbols in other permutations
2314
Decode the one or more permutations
2316
Identify a permutation that produces a decoded EDC value that matches an expected EDC value, wherein the expected EDC value corresponds to a predefined value for EDCs transmitted over the multi-wire interface to enable detection of up to two symbol errors at the receiver

FIG. 23

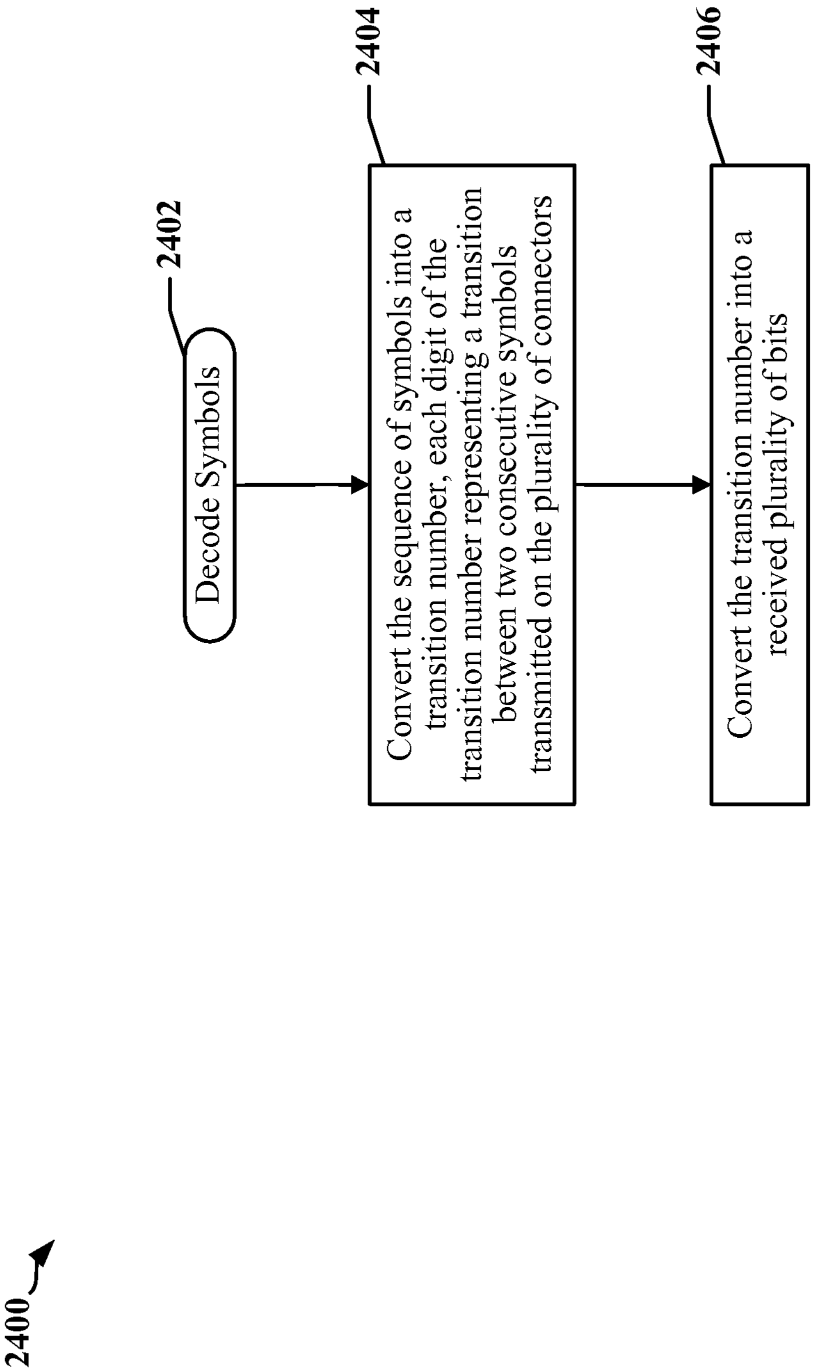

2400
2402
Decode Symbols
2404
Convert the sequence of symbols into a transition number, each digit of the transition number representing a transition between two consecutive symbols transmitted on the plurality of connectors
2406
Convert the transition number into a received plurality of bits

FIG. 24

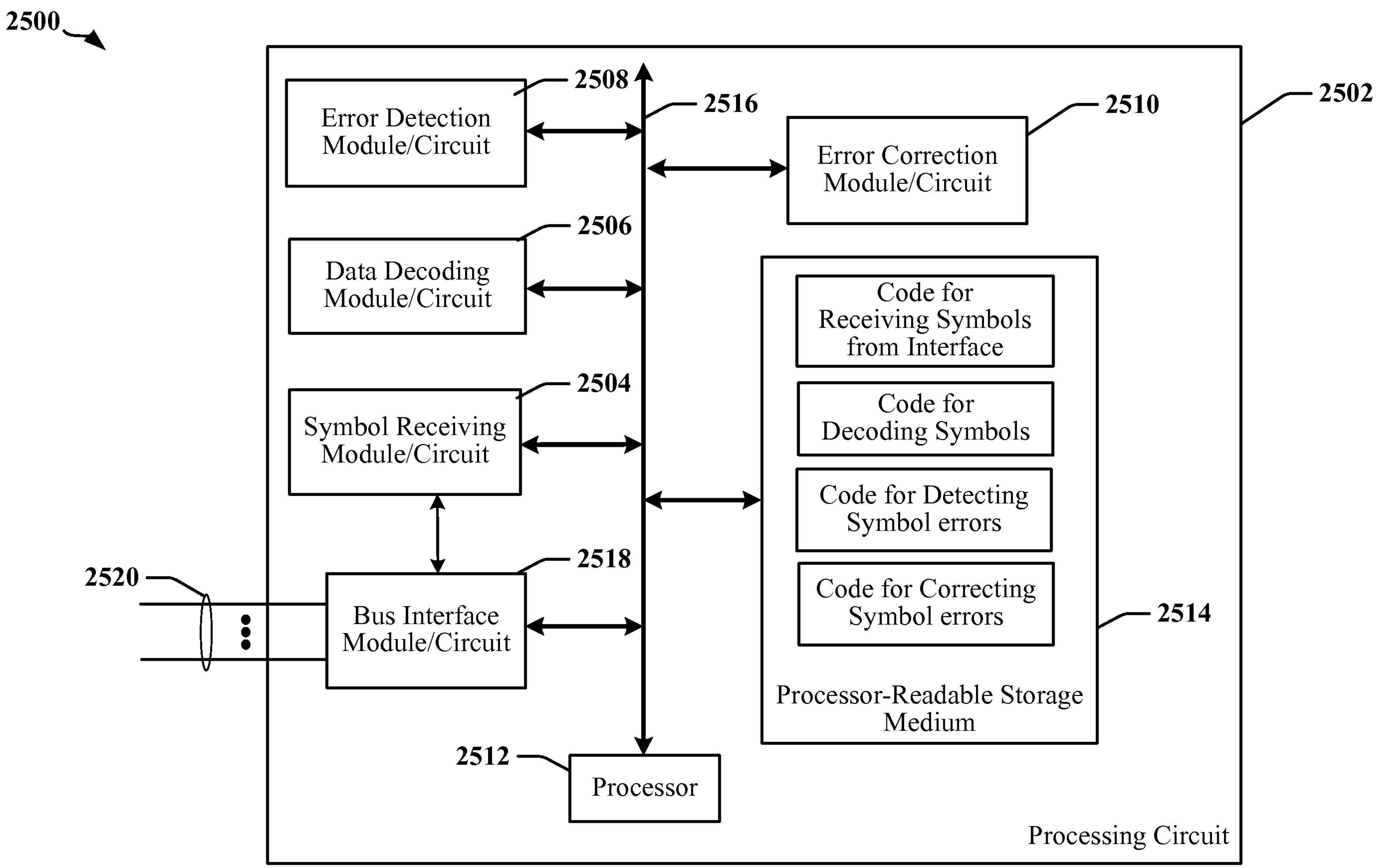

2500
2508
Error Detection Module/Circuit
2516
2510
Error Correction Module/Circuit
2502
2506
Data Decoding Module/Circuit
2504
Symbol Receiving Module/Circuit
Code for Receiving Symbols from Interface
Code for Decoding Symbols
Code for Detecting Symbol errors
Code for Correcting Symbol errors
2514
Processor-Readable Storage Medium
2520
2518
Bus Interface Module/Circuit
2512
Processor
Processing Circuit

FIG. 25

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2015/062511

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04L1/00 H03M13/03 H04L7/033 H04L25/02 H04L25/14
H04L25/493 H04L1/24
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H04L H03M

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2014/270026 A1 (SENGOKU SHOICHIRO [US] ET AL) 18 September 2014 (2014-09-18) paragraph [0003] - paragraph [0010] paragraph [0058] - paragraph [0061] paragraph [0092] - paragraph [0096] -----	1-30
A	US 2003/152154 A1 (JOHNSON RYAN C [US]) 14 August 2003 (2003-08-14) abstract paragraph [0002] paragraph [0022] paragraph [0047] - paragraph [0052] -----	1-30
A	CN 101 493 804 A (IBM [US]) 29 July 2009 (2009-07-29) figures 2,5,6 -----	1-30

□ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search 17 February 2016	Date of mailing of the international search report 25/02/2016
Name and mailing address of the ISA/ European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Authorized officer Palacián Lisa, Marta

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2015/062511

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2014270026 A1	18-09-2014	CN 105210047 A	30-12-2015
		EP 2972928 A1	20-01-2016
		KR 20150132270 A	25-11-2015
		TW 201441823 A	01-11-2014
		US 2014270026 A1	18-09-2014
		US 2015365226 A1	17-12-2015
		WO 2014150984 A1	25-09-2014
US 2003152154 A1	14-08-2003	NONE	
CN 101493804 A	29-07-2009	NONE	