



(51) International Patent Classification:

G06F 13/14 (2006.01) G06F 11/08 (2006.01)  
G06F 13/16 (2006.01) G06F 12/00 (2006.01)

(21) International Application Number:

PCT/US2012/028658

(22) International Filing Date:

10 March 2012 (10.03.2012)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

61/481,151 30 April 2011 (30.04.2011) US

(71) Applicant (for all designated States except US): **RAM-BUS INC.** [US/US]; 1050 Enterprise Way, Suite 700, Sunnyvale, CA 94089 (US).

(72) Inventor; and

(75) Inventor/Applicant (for US only): **PEREGO, Richard, E.** [US/US]; 2902 E. 148th Place, Thornton, CO 80602 (US).

(74) Agent: **SHEM WELL, Charles, E.**; P.O.Box 70307, Sunnyvale, CA 94086 (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, 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, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, 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, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, 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, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— of inventorship (Rule 4.17(iv))

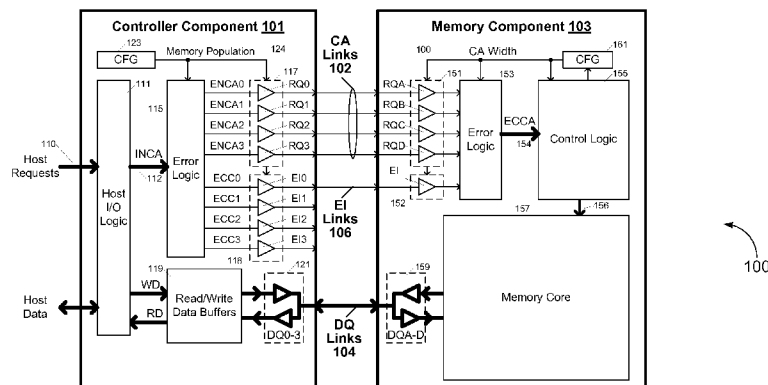
Published:

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

— before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))

(54) Title: CONFIGURABLE, ERROR-TOLERANT MEMORY CONTROL

FIG. 1



(57) Abstract: Configurable, error-tolerant communication of memory control information between components of a memory system. A controller component and memory component each have a variable-width command/address (CA) interface that operates in conjunction with an error detection/correction (EDC) channel to enable a variable level of error detection and correction with respect to command/address information conveyed between the two components as the widths of the CA interfaces are adjusted.

WO 2012/151001 A1

**CONFIGURABLE, ERROR-TOLERANT MEMORY CONTROL**

## TECHNICAL FIELD

[0001] The present invention relates generally to the field of integrated circuits, and more particularly to error-tolerant communication of operational information between components of a memory system.

## BRIEF DESCRIPTION OF THE DRAWINGS

[0002] The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

[0003] Figure 1 illustrates an embodiment of a memory system that enables configurable, error-tolerant communication of memory access commands and addresses;

[0004] Figure 2 presents an exemplary high-level view of the memory system of Figure 1, illustrating the increasing storage capacity and command/address communication error-tolerance as additional memory components are added;

[0005] Figure 3 illustrates more detailed embodiments of the command/address interface, error interface, error detection/correction logic, and control logic and configuration circuit of the memory component depicted in Figure 1;

[0006] Figure 4A illustrates more detailed embodiments of the command/address interface, error interface, error detection/correction logic, and configuration circuit of the controller component shown in Figure 1;

[0007] Figure 4B illustrates exemplary control port assignments within the controller component logic of Figure 4A;

[0008] Figure 4C illustrates a table of operating modes that may be selected within the controller component logic of Figure 4A;

[0009] Figures 5A-5C illustrate exemplary arrangements of command/address bits and error-checking bits within incoming bit streams for respective modes of operation within the memory component and controller component of Figure 1; and

[0010] Figure 6 illustrates a sequence of configuration operations that may be executed by a controller component (and/or another component that controls the controller component) to configure the command/address error detection/correction operation within the memory system of Figure 1.

## DETAILED DESCRIPTION

**[0011]** Configurable, error-tolerant communication of memory control information between components of a memory system is disclosed in various embodiments. In a number of embodiments, for example, a controller component and memory component each have a variable-width command/address (CA) interface that operates in conjunction with an error information (EI) channel to enable a variable level of error detection and/or correction (“EDC”) with respect to command/address information (i.e., information bearing command and/or address values, also referred to herein as “request information”) conveyed between the two components as the widths of the CA interfaces are adjusted. In one implementation (or configuration), the error information channel is implemented by respective dedicated signaling interfaces within the controller component and memory component(s), and one or more chip-to-chip signaling links coupled between those interfaces. In another implementation, the error information channel is implemented by time multiplexed transmission of error information over one or more CA links, with error information being, for example, appended to (or pre-pended to or embedded within) a corresponding command/address transmission. In a number of embodiments described below, the error information includes error-check bits that may be used to detect and in some instances correct one or more bit errors within corresponding information transmitted over the CA links. Accordingly, the error information is occasionally referred to herein as error detect/correct information and the error-related operations as error detection/correction (EDC) operations. More generally, the error information includes any information that can be used or associated with detecting occurrence of an error. The corrective action, if any, may be independent of the error detection operation. Accordingly, all references to “error detection/correction” information or operations refer to error detection information or error detection operations that may facilitate but do not require error correction.

**[0012]** Figure 1 illustrates an embodiment of a memory system 100 that enables configurable, error-tolerant communication of memory access commands and addresses. The memory system includes a controller component 101 and memory component 103 coupled to one another via command/address (CA) signaling links 102 and write/read data (DQ) signaling links 104. In the particular embodiment shown, and other embodiments described below, the control and memory components are also coupled to one another via an error information (EI) signaling link 106, though in all such cases the information communicated over the error information signaling link may be partially or entirely time-multiplexed onto the CA signaling links and/or DQ signaling links.

**[0013]** In general, the controller component 101 and memory component 103 operate as master and slave, with the controller component issuing memory access commands (e.g., row and column access commands, program/erase commands, etc.), maintenance commands (e.g., memory core refresh, timing calibration, signal driver/receiver calibration, equalization control, etc.) and configuration commands (e.g., register programming commands) to the memory component via CA signaling links 102, and the memory component responding by carrying out the commanded operation, such as storing write data received from the controller component or returning read data to the controller component via DQ signaling links 104.

**[0014]** Memory access commands may vary according to the core technology (e.g., row activation commands and column read/column write commands within a DRAM (dynamic random access memory) device; program and erase commands within a flash memory device, etc.), but are generally accompanied by address values that specify memory core locations to be accessed in various memory core operations. In the embodiment of Figure 1 and those below, a DRAM memory core is assumed within memory component 103 (and thus memory access commands that specify row activation operations, column read operations and column write operations), though in all cases a different type of memory core may be present and a different set of memory access commands may apply.

**[0015]** In the embodiment of Figure 1, the control and memory components (101, 103) are implemented by distinct (separate) IC dies with the interconnecting CA, DQ and EI signaling links formed by printed circuit board traces, cables (e.g., flex-tape cables or the like), bond wires, through-silicon vias, or any other structures for conveying signals between the two dies. In other embodiments, the control and memory components may be implemented by separate functional circuit blocks within the same IC die and the interconnecting signaling links formed within one or more metal layers or other on-die conductive structures. Conversely, the individual components shown in Figure 1 and other embodiments below may each be representative of multiple integrated circuit dies. For example, memory component 103 may be representative of multiple memory components disposed on a socketed (i.e., removably inserted into a connector socket) memory module, coupled in common to CA signaling links 102 and coupled to respective, mutually exclusive subsets of DQ links 104, thereby forming a “rank” of memory components that may be read from and written to in parallel, as a unit.

**[0016]** While shown in isolation, memory system 100 generally forms part of a subsystem within a larger host system (e.g., mobile phone, notebook computer, tablet computer, desktop computer, video/audio playback/recording device, gaming device, global positioning system or any other device requiring data storage) having one or more memory access requestors, such as a

processor or application-specific IC (ASIC). Though generally described herein as being separate from controller component 101, any or all of the memory access requestors or “host requestors” may be implemented on the same or different die as the controller component 101, as in the case of a processor core (or multiple processor cores) implemented on a die also having a memory controller function. In all cases, each host requestor may issue memory access requests to (or within) controller component 101, requesting retrieval or storage of data at specified logical or physical addresses within memory component 101. As discussed below, controller component 101 includes circuitry to organize such memory access requests, referred to herein (and shown in Figure 1) as “host requests,” 110 for eventual execution, for example by converting each host request into one or more command/address values and queuing the command/address values for execution.

**[0017]** Internally, controller component 101 includes host input/output (I/O) logic 111, error logic 115, CA output interface 117, error information (EI) output interface 118, read/write data buffers 119, data I/O interface 121, and configuration circuit 123, all of which cooperate to service host requests 110. More specifically, host I/O logic 111 converts/organizes host memory access requests (“host requests”) into a stream of incoming command/address values (INCA) 112 and transfers read and write data corresponding to the command/address values between read/write data buffers 119 and a host data interface (“Host Data”). The incoming command/address values 112 are supplied in sequence to error logic 115 which outputs, in turn, a configurable number of encoded command/address values (ENCA0-ENCA3) and corresponding error correction code (ECC) values (ECC0-ECC3) to respective serializing output drivers within CA interface 117 and EI output interface 118. More specifically, CA output drivers RQ0-RQ3 constitute CA output interface 117 (or request port) and are selectively enabled to transmit the configurable number of command/address values as serial bit streams over respective CA links 102. Similarly, error information output drivers, EI0-EI3, constitute EI output interface 118 and are selectively enabled to transmit the error correction code values to respective memory components 103. In the particular example shown, only one memory component 103 is coupled to controller component 101, so only a single error information output driver (EI0) is enabled to transmit the serialized ECC value over EI signaling link 106.

**[0018]** As discussed in greater detail below, configuration circuit 123 is programmed (e.g., by a host component or by operation of the controller component itself) with one or more values that indicate the ratio of active (i.e., enabled) CA output drivers for each active EI output driver and thus, in effect, the population size (or number) of independently controlled memory components 103 attached to controller component 101. This “memory population” information

124 is provided to error logic 115 which responsively encodes the incoming CA values into the outgoing encoded CA values (i.e., in accordance with the number of independently controlled memory components 103), and also to the CA and EI output driver banks 117 and 118, thereby selectively enabling the error information output drivers EI0-EI3 according to the number of independently controlled memory components present in memory system 100.

**[0019]** Turning to memory component 103, a bank of input receivers, RQA-RQD, forms a variable-width CA interface 151 that receives and deserializes the incoming serialized, encoded command/address values transmitted by controller component 101. The deserialized command/address values are supplied, together with error information received and deserialized within error information interface 152 (populated by a solitary deserializing receiver, "EI"), to an error logic circuit 153 which performs an error detection and correction functions to yield a sequence of error-corrected command/address values 154 (ECCA). The error-corrected command/address values 154 are supplied to control logic 155 which, in turn, issues addresses and control signals 156 to memory core ("core array and access control logic") 157 to carry out the commanded memory operations. In the case of a row activation (or sense) operation, for example, control logic 155 outputs bank and row addresses (conveyed in an error-corrected CA value) and row-sense control signals to memory core 157 to open a selected a page of memory within the core array (i.e., loading an address-specified row of data within address-specified storage bank into a sense amplifier array for subsequent column read/column write access). In the case of column read or write operations, control logic 155 outputs bank and column addresses and column control signals to memory core 157 to enable incoming write data (received via DQ signaling links 104 deserializing data receivers within data I/O transceiver 159) to be loaded into the address-specified column of an open memory page, or to enable the address-specified column of read data to be output to the controller component 101 via serializing output drivers within data I/O transceiver 159 and DQ signaling links 104.

**[0020]** The memory component also includes a configuration circuit 161 that configures the logical width of CA interface 151. In the embodiment shown, for example, configuration circuit 161 provides a CA width value 100 (or signals derived in response thereto) to enable only those serializing receivers (RQA-RQD) within CA interface 151 employed for CA signal reception in a given memory system configuration, and also provides the CA width value (or derived signals) to error logic 153 and control logic 155. As explained below, error logic 153 may perform one of a number of different error information decoding operations based on the width of the CA interface (and thus the ratio of error information to CA information received over a given interval). Control logic 155 may also respond to incoming error-corrected command/address

values in accordance with the width configuration of the CA interface. For example, in an embodiment in which each enabled CA receiver (RQA-RQD) receive a respective stream of memory access commands, the control logic may apply the different streams of memory access commands to respective regions of memory core 157 in parallel, in effect dividing the memory core into a number of separately accessed logical memories according to the interface width configuration. Also, while embodiments herein are generally described in terms of error detection and error correction functions, in all cases, the error correction function may be performed by other logic circuits within a given integrated circuit die, by another integrated circuit die (e.g., an error-detecting memory component may communicate an error detection to a controller component, inviting a re-transmission of the memory command), or even dispensed with altogether.

**[0021]** Still referring to Figure 1, configuration circuit 161 may be programmed in response to one or more register-write commands delivered via all or a subset of CA signaling links 102 (e.g., delivered over the CA signaling link coupled to CA receiver RQA, which is enabled even in the minimum CA interface width configuration) together with configuration information (i.e., interface width information and optionally other information to be stored within configuration circuit 161) transmitted, for example, in bit-fields otherwise used to convey address information. The register-write commands and/or configuration information may additionally or alternatively be delivered to one or more attached memory components via a sideband control interface or serial interface (not shown). However communicated, control logic 155 responds to the register-write command(s) by storing the configuration information within configuration circuit 161, thus establishing the width of the CA interface for subsequent transfers via CA signaling links 102. As explained below, a similar programming operation may be carried out within the controller component, but in response to host commands or by operation of the controller component itself upon determining the nature and quantity of attached memory components.

**[0022]** Figure 2 presents a high-level view of memory system 100, illustrating the increasing storage capacity and command/address communication error-tolerance as additional memory components 103 are added. That is, in the dual-memory arrangement (“x2”), a second memory component 103 is added (relative to the single-memory arrangement, “x1”) to double the storage capacity of the memory system, and in the quad arrangement (“x4”), third and fourth memory components are further added to double the storage capacity again (i.e., yielding four times the capacity of the system with a single memory component 103). The width of the data interface of memory component 103 (i.e., interface 159 in Figure 1) is configurable to accommodate the full data bandwidth of controller component 101 in the single memory arrangement, half the data

bandwidth of the controller component in the dual memory arrangement, and one quarter of the data bandwidth in the quad memory arrangement. That is, in the single-memory arrangement, controller-component data I/O circuits DQ0-DQ3, each assumed to be 9-bits wide for purposes of example (other widths may be supported in alternative embodiments), are coupled to counterpart memory-component data I/O circuits DQA-DQD, respectively, to establish a 36-bit wide data interface within memory component 103. In the dual-memory arrangement, each memory component 103 is operated with an 18-bit wide data interface with controller-component I/O circuits DQ0 and DQ1 coupled to I/O circuits DQA and DQB of a first memory component 103, and controller-component I/O circuits DQ2 and DQ3 coupled to I/O circuits DQA and DQB of another memory component 103. Subdividing further, in the quad-memory arrangement each memory component 103 operates with a 9-bit wide data interface, with controller-component I/O circuits DQ0-DQ3 coupled to respective I/O circuits DQA within four memory components 103.

**[0023]** Still referring to Figure 2, the configurable width of the CA interface of memory component 103 enables the CA bandwidth of controller component 101 to be divisibly allocated, providing further flexibility within the memory system. That is, a given memory component 103 may receive command/address information from all, half or one quarter of the CA output drivers within the CA interface of controller component 101 in the single, dual and quad memory arrangements, respectively, and thus receive command/address information at the full, half or quarter command/address signaling bandwidth of controller component 101.

**[0024]** As Figure 2 illustrates, each memory component 103 is coupled to receive error information via a respective error information link (which may be more than one link in alternative embodiments) so that the error information signaling bandwidth allocated to a given memory component remains constant even as the number of memory components rises and the CA bandwidth per component falls. Said another way, the ratio of error information to CA information provided in conjunction with a block of command/address information rises as the CA interface of the memory component narrows. Accordingly, as shown by the graphic at 170, the error correction capability within a given memory component and thus the error tolerance of the memory system as a whole rises as the memory component population (and storage capacity of the memory system) increases.

**[0025]** Figure 3 illustrates more detailed embodiments of the CA interface 151, EI interface 152, error logic 153, and control logic 155 and configuration circuit 161 of the memory component depicted in Figure 1. In the embodiment shown, configuration circuit 161 includes a configuration register 203 having bypass and CA-width fields (“Byp” and “CA\_Width”), and



decode logic 205 that generates a 2-bit mode-select signal (“Mode\_Sel[1:0]”) and a set of receiver-enable signals (EN\_EI, and EN\_RQA - EN\_RQD) in accordance with the control values programmed within the bypass and CA-width fields. More specifically, when the bypass value (a single bit in this example) is a ‘1’, error detection/correction circuitry within error logic 153 is bypassed, with CA values received via one or more of the CA link receivers (RQA-RQD) within CA interface 151 being delivered without error detection or correction to request decode and control modules 251, 253, 255, 257 within the control logic 155. By contrast, when the bypass value is a ‘0’, EDC logic 153 operates on command/address values received via CA interface 151, detecting and correcting errors therein using one of multiple different detect/correct circuit blocks in accordance with the width of the CA interface specified in the CA-width field. In the particular embodiment shown, three different CA interface width configurations are supported and are referred to herein as the “x1,” “x2,” and “x4” CA interface widths according to the number of CA signaling links used to convey CA information to the memory component in each configuration.

**[0026]** In the unbypassed, x4 CA interface configuration (i.e., Byp=0 and CA\_Width=‘10’ within configuration register 203), decode circuit 205 asserts all four CA receiver-enable signals (EN\_RQA - EN\_RQD) and the EI receiver-enable signal (EN\_EI), thereby setting CA interface to its maximum logical width (i.e., logical width equal to physical width) and enabling reception of error information. Each deserializing receiver, RQA-RQD and EDC, includes a serial signal receiver 211 that samples an incoming serial bit stream from a respective CA or EI signaling link in response to rising and falling edges (i.e., “even” and “odd” edges) of a high-speed clock signal (not specifically shown) and outputs a corresponding logic-level bit stream to a respective 1:16 deserializing circuit 215. The deserializing circuit 215 converts each incoming sequence of 32 serial bits into a pair of 16-bit input words (i.e., an even input word formed by bits captured at 16 successive rising edges of the high-speed clock, and an odd input word formed by bits captured at 16 successive falling edges of the high-speed clock), with the input words being latched in response to a lower-speed logic clock (“CLK”). Thus, at peak CA bandwidth, CA interface 151 is capable of delivering four pairs of 16-bit CA input words (CA\_EVEN[15:0]/CA\_ODD[15:0], CA\_EVEN[31:16]/CA\_ODD[31:16], CA\_EVEN[47:32]/CA\_ODD[47:32] and CA\_EVEN[63:48]/CA\_ODD[63:48]) to EDC logic 153 per cycle of the logic clock (and over 16 cycles of the high-speed clock). The EI receiver is similarly capable of delivering a pair of 16-bit EI input words, but, because at most 15 bits of each input word constitute bits within an error correction code (ECC) in the exemplary

embodiment shown, the EI receiver output is designated “ECC\_EVEN[14:0]” and “ECC\_ODD[14:0].”

**[0027]** Continuing with the x4, unbypassed configuration, the reception of at least 15 EDC bits per 64 CA bits provides sufficient check-bit redundancy (e.g., using for example and without limitation Hamming codes, BCH (Bose, Ray-Chaudhuri, Hocquenghem) codes, Reed-Solomon codes, Turbo codes, Low-Density Parity Check (LDPC) codes or any number of other error-check coding schemes) to enable detection of as many as three bit errors (“triple error detect,” or “TED”) and to enable correction of as many as two bit errors (i.e., “double error correction” or “DEC”). Herein, EDC circuits are described using the nomenclature “(n, k),” where “k” represents the total number of bits supplied to the EDC circuit, and “n” represents the number of “message bits” and the difference between n and k (n minus k) represents the number of error-checking bits (also referred to herein as “redundancy bits,” parity bits, ECC bits, EDC bits, syndrome bits, error information bits, etc.) Accordingly, in the embodiment of Figure 3, the four 16-bit even CA input words are supplied, together with 15 even ECC bits, to a double-error-correct/triple-error-detect (DEC-TED) error detection/correction circuit 221e within error logic 153. EDC circuit 221e, depicted as a “(79,64)” DEC-TED EDC circuit in Figure 3, performs the error detection/correction function, latching a “corrected” 64-bit even CA output value 222e in response to each rising edge of the logic clock (note that more than one cycle of the logic clock may be allocated for the EDC operation in alternative embodiments), together with one or more optional error-detect signals (not specifically shown). The four 16-bit odd CA input words are similarly supplied, together with the 15 odd ECC bits, to (79,64) DEC-TED EDC circuit 221o, which operates concurrently with circuit 221e, to latch a corrected 64-bit odd CA output 222o at each rising edge of the logic clock, together with one or more optional error-detect signals.

**[0028]** Figure 5A illustrates the arrangement of CA bits and error-checking bits (i.e., EDC information) within the incoming CA and error information bit streams over 32 successive bit intervals (e.g., 16 cycles of the high-speed clock in an embodiment that employs falling and rising edge clocking, or rising edge clocking using two complementary clocks) in the x4 CA width, unbypassed operating mode. The lightly shaded check bits and CA bits received during even numbered bit intervals are referred to collectively as an even ECC block and correspond to the CA bits and error information bits (or error checking bits or EI bits) operated upon by EDC circuit 221e during a given cycle of the logic clock. By contrast, the more darkly shaded check bits and CA bits received during the odd numbered bit intervals, referred to collectively herein as an “odd ECC block,” correspond to the CA and error information bits operated upon by EDC

circuit 221o during that same logic clock cycle. As Figure 5A shows, the two ECC logic blocks are effectively interleaved due to the separation of their respective inputs within the deserializing circuits 215 of the CA interface 151 and EI interface 152. While this “bit-interleaving” is not required (the two ECC blocks could alternatively be formed from CA/EI bits received in respective bursts of 16 bit intervals), bit-interleaving may be advantageous in systems that exhibit bursty bit errors. For example, when bit-interleaved as shown in Figure 5A, a burst of four bit errors (i.e., bit errors in four successive bit intervals) will yield only two bit errors in each ECC block, and thus a number of bit errors below the bit-correction limit for each of the even and odd EDC circuits 221e, 221o. By contrast, if the same 4-bit error burst occurred within a single ECC block (i.e., where the ECC blocks are formed from information received in successive groups of bit intervals, rather than individually interleaved bit intervals), the number of bit errors would exceed the error detection and error correction capabilities of the DEC-TED EDC circuits provided.

**[0029]** Still referring to Figure 5A, it should be noted that the interleaved even and odd ECC blocks do not necessarily correspond to the organization of commands and addresses within those blocks. For example, the 32 bits received via RQA may form a single command/address value, despite the fact that the constituent bits of the command/address value are split between even and odd ECC blocks (and thus routed to separate even and odd EDC circuits 221e, 221o). Accordingly, after undergoing error detection/correction within EDC circuits 221e and 221o, the resulting corrected 64-bit ECC values 222e, 222o are restored to their original, interleaved arrangement within x4 interleave circuit 231 (i.e., within interleave logic 230 of Figure 3), before being supplied as a set of four 32-bit values, ECCAx4[127:0] to control logic 155. Note that, because the even and odd EDC circuits output respective corrected 64-bit CA values in parallel (i.e., 222e, 222o), the interleaving of the constituent bits of those two 64-bit values may be effected without active circuitry and instead by the manner of interconnection between the outputs of EDC circuits 221e, 221o and the inputs to control logic 155. In such an embodiment, interleave circuit 231 (and the other interleave circuits shown within interleave logic 230) may be viewed as a pass-through element in which the outputs of EDC circuits 221e, 221o are overlapped to achieve the desired bit ordering at the input to control logic 155.

**[0030]** Referring again to configuration circuit 161 and more specifically to mode table 207, the unbypassed, x4 CA width configuration yields a mode-select signal “10” that selects, within front-end multiplexers 241, 243, 245 and 247 of control logic 155, the output of interleave circuit 231 (i.e., CAX4 with DEC-TED) to be passed in respective 32-bit portions to request and

decode modules 251, 253, 255 and 257. The request and decode modules, in response, output respective sets of address and control signals to the memory core.

**[0031]** Still referring to Figure 3, in the unbypassed, x2 CA width configuration, CA receivers RQC and RQD are disabled (i.e., by deassertion of signals EN\_RQC and EN\_RQD), thus narrowing the CA interface to a two-port logical width and therefore to half the four-port physical width. In an embodiment in which the CA signaling rate remains constant across changes in the CA interface width, this halving of the CA interface width results in a halving of the CA bandwidth, and thus reception of 64 CA bits during each logic clock cycle (i.e., each sequence of 32 bit intervals) instead of 128 CA bits. Because the EDC function is engaged (i.e., not bypassed), the deserializing receiver within EI interface 152 continues to receive as many as 32 error checking bits per logic cycle, thereby doubling the EI to CA bit ratio relative to the unbypassed, x4 mode. In the particular implementation shown, the increased EI bit density (i.e., increased ratio of EI bits to CA bits) is sufficient to enable double error correction and triple error detection on even and odd blocks of 32 CA bits. More specifically, the even pair of 16-bit CA words output by CA receivers RQA and RQB are supplied to DEC-TED Even EDC (45,32) circuit 223e together with thirteen even ECC bits (e.g., ECC\_EVEN[12:0]), and the odd pair of 16-bit CA words are supplied to DEC-TED Odd EDC (45,32) circuit 223o together with thirteen odd ECC bits (e.g., ECC\_ODD[12:0]) to effect the x2 ECC mode shown in Figure 5B. The two EDC logic circuits 223e, 223o output respective 32-bit corrected CA values to x2 interleave circuit 233 which, in turn, restores the original interleaving of even and odd bits in the 64-bit CA value ECCAx2[63:0] supplied in respective 32-bit halves to multiplexers 241 and 243. As shown in mode table 207, the unbypassed, x2 CA width configuration yields a mode-select signal '01' that selects, within multiplexers 241 and 243, the output of interleave circuit 233 (i.e., CAx2 with DEC-TED) to be passed to request/decode modules 251 and 253. The remaining request/decode modules (255 and 257) may be disabled.

**[0032]** In the unbypassed, x1 CA width configuration, CA receivers RQB, RQC and RQD are disabled (i.e., by deassertion of signals EN\_RQB, EN\_RQC and EN\_RQD), thus narrowing the CA interface to a single-port logical width, one-quarter of the four-port physical width. In an embodiment in which the CA signaling rate remains constant across changes in the CA interface width, this quartering of the CA interface width results in a quartering of the CA bandwidth, and thus reception of 32 CA bits during each logic clock cycle (i.e., each sequence of 32 bit intervals) instead of 128 CA bits. Because the EDC function is engaged (i.e., not bypassed), as many as 32 error checking bits per logic clock cycle continue to be received via EI interface 152, thus quadrupling the EI-to-CA bit ratio relative to the unbypassed, x4 mode. In the particular

implementation shown, the increased EI bit density (i.e., increased ratio of EI bits to CA bits) enables triple error detection and triple error correction (TEC) on even and odd CA words. More specifically, the even 16-bit CA word output by CA receiver RQA (CA\_EVEN[15:0]) is supplied to TEC Even EDC (31,16) circuit 225e together with fifteen even ECC bits (e.g., ECC\_EVEN[14:0]), and the odd pair 16-bit CA word from CA receiver RQA (CA\_ODD[15:0]) is supplied to TEC Odd EDC (31,16) circuit 225o together with fifteen odd ECC bits (i.e., ECC\_ODD[15:0]) to effect the x1 ECC mode shown in Figure 5C. The two EDC circuits 225e and 225o output respective 16-bit corrected CA values to x1 interleave circuit 235 which, in turn, restores the original interleaving of even and odd bits in the 32-bit CA value ECCAx1[31:0] supplied to multiplexer 241. As shown in mode table 207, the unbypassed, x1 CA width configuration yields a mode-select signal '00' that selects, within multiplexer 241, the output of interleave circuit 234 (i.e., CAx1 with TEC) to be passed to request/decode module 251. The remaining request/decode modules (253, 255 and 257) may be disabled.

**[0033]** Still referring to Figure 3, if the bypass bit is set within configuration register 203, the EI receiver is disabled (i.e., EN\_EI deasserted) and the RQA-RQD receivers are enabled according to the CA interface width setting. For simplicity, only the x4 bypass path is shown, with the outputs of all four deserializing receivers (RQA-RQD) being interleaved within interleave circuit 237 to yield a 128-bit uncorrected command/address value (UCCAx4[127:0]) which is delivered in four 32-bit values to multiplexers 241, 243, 245, 247, respectively. As shown in mode table 207, the bypassed configuration yields a mode select signal '11' that selects, within multiplexers 241-247, the four 32-bit uncorrected command/address values (CAu[31:0], CAu[63:32], CAu[95:64] and CAu[127:96]) to be passed to request/decode modules 251-257. In the embodiment shown, bypassing the error detection/correction logic 200 reduces memory access latency by a clock cycle (i.e., consumption of a clock cycle in the EDC stage of the command pipeline is avoided). In alternative embodiments, an additional register, clocked by the logic clock, may be placed in the uncorrected CA path to equalize (or levelize) the CA latency across all CA pipeline configurations.

**[0034]** Reflecting on the memory component operation described in reference to Figure 3, it can be seen that the command/address error correction capability of the memory component, and thus the CA error-tolerance of the memory system as a whole increases as the width of the memory component CA interface is narrowed. Accordingly, as shown in Figure 2, the command/address error detection capability and error correction capability increase as command/address bandwidth of the controller component is dispersed among an increasing number of attached memory components. Though x1, x2 and x4 interface widths are shown,

intermediate interface widths (e.g., x3,) may also be supported, and a physically wider CA interface may be varied between a larger number of logical width configurations.

**[0035]** Figure 4A illustrates more detailed embodiments of the CA interface 117, EI interface 118, error logic 115, and configuration circuit 123 of the controller component shown in Figure 1. The configuration circuit 123 includes a configuration register 281 having fields to store (i) a control-port count value (“CP\_Count”) that indicates the number of activated (enabled) control ports within the controller component, (ii) control port width values (CP\_Width[3:0]) that indicate respective CA interface widths for the activated control ports, and (iii) bypass values (Byp[3:0]) that indicate, for each activated control port, whether the error detection/correction function is to be bypassed. In the embodiment shown, each control port (P0, P1, P2 and P3) serves as a logical port for routing CA information and EI information to a respective attached memory component. Thus, if only a single memory component is attached to the controller component, the control-port count field is set to 1, thereby activating control port P0 (and disabling control ports P1-P3) and enabling allocation of the full CA bandwidth of the controller component to a single memory component. By contrast, if two or more memory components are to be controlled by the controller component, the control-port field is set to 2, 3 or 4, thereby activating various combinations of control ports and splitting allocation of the CA bandwidth of the controller component among the two or more attached memory components.

**[0036]** As shown, configuration circuit 123 includes a decoder 283 coupled to receive the control-port count value, control-port width values and bypass values from the corresponding fields of configuration register 281 and to generate a number of 2-bit mode-select signals to be output to respective control ports (i.e., Mode\_Sel[3:0][1:0]), as there are four control ports in this example), a set of receiver-enable signals (EN\_EI, and EN-RQ0-EN\_RQ3), and a set of port-select signals PSel[2:0] in accordance with the control values programmed within configuration register 281.

**[0037]** The port-select signals are supplied to port multiplexers 330, 331 and 332 to pass the outputs of selected control ports to output drivers RQ0-3, and thus allocate portions of the controller component CA interface to the respective control ports. Figure 4B illustrates an exemplary control port assignment for various control-port count values and control-port width values that may be applied within the embodiment of Figure 4A. As shown, the control-port count value (CP Count) may be set to 1, 2, 3 or 4 (referred to herein as single, dual, triple and quad control port configurations, respectively), and the width of each control port (CP Width) set in accordance with the control port count. Thus, in the single control port configuration, the active control port, P0, may be set to widths of 1, 2 or 4, while the inactive control-port widths

are set to zero; in the dual control-port configuration, the active control ports P0 and P1 may each be set to widths of 1 or 2, and so forth as shown.

**[0038]** Still referring to the exemplary configuration options shown in Figure 4B, when in a single-port, x1-width configuration, output drivers RQ1-RQ3 are unused (disabled) and the port-selections within port multiplexers are shown as 'xx' (don't care). In the single-port, x2 width configuration, RQ1 is allocated to control port 0 (RQ2 and RQ3 remaining disabled), and in the single-port, x4 width configuration, all four output drivers are allocated to control port 0. In the dual-port configuration (i.e., P0 and P1 active), output drivers RQ1-RQ3 are allocated to control ports 0 and 1 according to the width configurations for those ports as shown. Similarly, in the triple-port configuration (i.e., P0, P1 and P2 active), output drivers RQ1-RQ3 are allocated to the active control ports according to their width configurations as shown. In the quad port configuration, an output driver is allocated to each of the active control ports and thus the port multiplexers select control ports 1, 2 and 3 to source CA information to output drivers RQ1, RQ2 and RQ3, respectively.

**[0039]** Returning to Figure 4A when the bypass value corresponding to a given control port is a '1', ECC generator bank 301 within error logic 115 is bypassed, and the incoming CA values (INCA) are delivered to without correction to serializing output drivers RQ0-RQ3 within the CA interface 117. By contrast, when the bypass value is a '0', error logic 115 operates on incoming command/address values (INCA) received via the host I/O logic (not shown in Figure 4A), generating error correction codes (ECCs) for respective blocks of command/address bits using one of multiple different detect/correct circuit ECC generators in accordance with the number of control ports specified by the CP\_Count value and the CA interface width to be allocated to the control port or control ports as indicated by the CP\_Width values. In the particular embodiment shown, three different CA interface width configurations are supported and are referred to herein as the "x1," "x2," and "x4" interface widths according to the number of CA signaling links used to convey CA information to an attached memory component. Additional and/or different CA interface width configurations may be supported in alternative embodiments.

**[0040]** In the unbypassed, x4 interface configuration within a given control port 'i' (i.e., Byp[i]=0 and CP\_Width[i]=4 within configuration register 281), even/odd splitter circuits 293, 294, 295 and 296 within buffer logic 291 convert respective incoming 32-bit CA values (INCA0[31:0]-INCA3[31:0]) into four respective sets of 16-bit even and odd input words (i.e., an even input CA word formed by bits 0, 2, 4, ..., 30 within the incoming 32-bit CA value, and an odd input CA word formed by bits 1, 3, 5, ..., 31) in response to each rising edge of logic

clock signal, CLK. The four 16-bit even CA input words are supplied to error correction code (ECC) generator 303e (i.e., within ECC generator bank 301) which includes logic circuitry to generate a sufficient number of error-check bits (15 in this example) to enable double-error correction and triple-error detection (DEC, TED) when provided, together with the original 64 CA bits, to counterpart EDC circuit 221e of Figure 3. ECC generator 303e is thus shown as a DEC-TED (79,64) ECC generator in Figure 4A and outputs a 15-bit ECC value 302 together with the original 64 even CA bits 304 to x4 interleave circuit 313 as frequently as once per clock cycle (i.e., at peak command throughput of the controller component). The four 16-bit odd input CA words are similarly supplied to ECC generator 303o (79,64, DEC-TED ECC generator) which, in turn, outputs a 15-bit ECC value and the original 64 odd input CA bits to x4 interleave circuit 313.

**[0041]** Referring to the detail view of control port 0 (P0), interleave circuit 313 recombines the originally split even and odd data words (recreating the input to splitter circuits 293) to produce a x4-mode encoded data value, ENCAx4[127:0], and similarly combines the even and odd ECC values to produce a consolidated x4-mode ECC word, ECCx4[31:0] (two bits of which are populated with don't-care information and thus may arbitrarily be set to '0' or '1'). As shown in mode table of Figure 4C, the unbypassed single-port (CP Count=1), x4 width (CP\_Width[0]=4) configuration yields a mode-select signal for control port 0 (Mode\_Sel[0][1:0]) that selects, within multiplexers 323, 325, 327 and 329, the output of interleave circuit 313 (i.e., CAx4 with DEC-TED) to be passed to the four 32:1 serializing transmitters of the CA interface 117 (i.e., each serializing transmitter including a 32:1 serializer 335 and an output driver 337) and that selects, within ECC multiplexer 321, the x4-mode ECC word, ECCx4[31:0], to be passed to a 32:1 serializing transmitter that forms error information interface 118 (EI0). The serialization operations within the CA interface and EI0 interface collectively yield respective bit streams corresponding to the x4 ECC mode shown in Figure 5A.

**[0042]** Still referring to the single-port mode shown in Figure 4C (i.e., CP\_Count=1), in the unbypassed x2 width configuration (i.e., CP\_Width=2 within configuration register 281), splitter circuits 293 and 294 convert respective incoming 32-bit CA values (INCA0[31:0] and INCA1[31:0]) into respective sets of 16-bit even and odd input words in response to each rising edge of controller clock signal, CLK. The two 16-bit even CA input words are supplied to ECC generator 305e which includes logic to generate a sufficient number of error-check bits (13 in this case) to enable double-error correction and triple-error detection when provided, together with the original 32 CA bits, to counterpart EDC circuit 223e of Figure 3. ECC generator 305e is thus shown as a DEC-TED (45,32) ECC generator in Figure 4A and provides a 13-bit ECC



output value together with the original 32 even CA bits to x2 interleave circuit 315 as frequently as once per clock cycle. The two 16-bit odd input CA words are similarly supplied to ECC generator 305o (45,32, DEC-TED ECC generator) which, in turn, outputs a 13-bit ECC value and the original 32 odd CA bits to interleave circuit 315.

**[0043]** Referring to the detail view of control port 0 shown in Figure 4A, interleave circuit 315 recombines the originally split even and odd data words to produce a x2-mode encoded CA value, ENCAx2[63:0], and similarly combines the even and odd ECC values to produce a consolidated x2-mode ECC word, ECCx2[31:0] (six bits of which are populated with don't-care information and thus may arbitrarily be set to '0' or '1'). As shown in Figure 4C, the unbypassed, single-port x2 interface width configuration yields a mode-select signal '01' that selects, within multiplexers 323 and 325, the output of interleave circuit 315 (i.e., CAx2 with DEC-TED) to be passed to a pair of 32:1 serializing transmitters and that selects, within ECC multiplexer 321, the x2-mode ECC word, ECCx2[31:0], to be passed to the 32:1 serializing transmitter included within EI interface 118. Decode circuit 283 responds to the x2 width configuration (and logic '0' state of the bypass bit) by deasserting EN\_RQ2 and EN\_RQ3 (thus disabling output drivers RQ2 and RQ3) and enabling the remaining output drivers. The serialization operations within the enabled output drivers of the x2 CA interface (i.e., RQ0 and RQ1) and the EI interface collectively yield respective bit streams corresponding to the x2 ECC mode shown in Figure 5B.

**[0044]** In the unbypassed, single-port, x1 interface configuration (i.e., CP\_Count=1, CP\_Width[0] = 1 and Byp[0]=0, within configuration register 281), splitter circuit 293 converts incoming 32-bit CA value INCA0[31:0] into even and odd input words (INCA\_EVEN[15:0] and INCA\_ODD[15:0]) in response to each rising edge of controller clock signal, CLK. The even CA input word is supplied to ECC generator 307e which generates a sufficient number of error-check bits (15 in this example) to enable triple error detection and triple error correction when provided, together with the original 16 even CA bits, to the counterpart EDC circuit 225e shown in Figure 3. ECC generator 307e is thus shown as a TEC (31,16) ECC generator in Figure 4A and provides a 15-bit ECC output value together with the original 16 even CA bits to x1 interleave circuit 317 within control port 0 as frequently as once per clock cycle. The 16-bit odd input CA word is similarly supplied to ECC generator 307o (31,16 TEC ECC generator) which, in turn, outputs a 15-bit ECC value and the original 16 odd CA bits to interleave circuit 317.

**[0045]** Within control port 0, interleave circuit 317 recombines the originally split even and odd data words to produce a x1-mode encoded CA value, ENCAx1[31:0], and similarly combines the even and odd ECC values to produce a consolidated x1-mode ECC word,

ECCx1[31:0] (two bits of which are populated with don't-care information and thus may arbitrarily be set to '0' or '1'). As shown in mode table 288, the unbypassed, single-port, x1 CA width configuration yields a mode-select signal (i.e., Mode\_Sel[0][1:0]) that selects, within multiplexer 323 the output of interleave circuit 317 (i.e., CAx1 with TEC) to be passed to serializing transmitter RQ0 and that selects, within ECC multiplexer 321, the x1-mode ECC word, ECCx1[31:0], to be passed to the serializing transmitter within EI interface 118. Decode circuit 283 responds to the single-port, x1 width configuration (and logic '0' state of the bypass bit) by deasserting EN\_RQ1, EN\_RQ2 and EN\_RQ3 (thus disabling output drivers RQ1, RQ2 and RQ3) and enabling the remaining output drivers. The serialization operations within the enabled output drivers of the x1 CA interface (i.e., RQ0) and the EDC interface collectively yield bit streams corresponding to the x1 ECC mode shown in Figure 5C.

**[0046]** In single port mode, control ports 1-3 are unused as shown in Figure 4C, so that corresponding error information interfaces EI1-EI3 are also unused and disabled to save power. When a given control port is unused, logic therein may be disabled (e.g., left in an unlocked state) to limit power consumption.

**[0047]** The overall interface bandwidth of the controller component may also be allocated entirely or fractionally to the single active control port in single port mode. By contrast, when two or more control ports are active (i.e., CP\_Count > 1), the overall CA interface width (and thus the CA bandwidth of the controller component) is split between the active control ports. Thus, in the dual control port configuration shown in Figure 4C, control ports 0 and 1 may each be operated in a x1 width configuration (enabling triple-error-correcting ECC generation within each control port and leaving half the controller component CA output drivers, RQ2 and RQ3, unused and disabled), in a x2 width configuration (enabling double-error-correcting/triple-error-detecting ECC generation within each control port and consuming the entire CA interface) or a combination of the two (one device in x1 width and the other in x2 width). In a triple control port configuration, control ports 0-2 may each be operated in a x1 width configuration (enabling triple-error-correcting ECC generation within each control port and leaving one of the controller component CA output drivers, RQ3, unused and disabled), or with one of the control ports in a x2 width configuration and the others in a x1 width configuration. Lastly, in a quad control port configuration, all four control ports are active and operated in a x1 width configuration, thereby enabling triple-error-correcting ECC generation within each control port as shown.

**[0048]** Reflecting on Figures 4A-4C, it can be seen that not all control ports need be operable in the same width configuration, and thus that interleave logic and multiplexer circuitry provided to support the full range of width configurations may be omitted in one or more of the

control ports. In an embodiment limited to the configurations shown in Figure 4C, for example, control ports 3 and 4 are operable only in the x1 width configuration and thus interleave circuits 313 and 315, and multiplexers 325, 327 and 329 may be omitted from that control port.

Similarly, control port 2 is operable only in the x1 and x2 width configurations so that interleave circuit 313 and multiplexers 327 and 329 may be omitted. In other embodiments, each of the control ports may be activated regardless of the activation state of other control ports (e.g., control port 4 activated, but not control port 0), so that numerous configurations other than those shown in Figure 4C may be supported and each control port populated, for example, by the interleave and multiplexing logic circuitry shown.

**[0049]** Still referring to Figures 4A and 4B, if the bypass bit for a given control port is set within configuration register 281 (i.e., Byp[i]=1), the EI interface for that control port is disabled (i.e., EN\_EI deasserted) and one or more of output drivers RQA-RQD are enabled and allocated to the control port according to the control port count and the interface widths for the active control ports. For simplicity, Figure 4A illustrates the signaling paths within control port 0 for the bypassed, single-port, x4-width configuration only, with the outputs of all four serializing output drivers (RQ0-RQ3) being enabled to output respective 32-bit uncorrectable CA values (UNCA) from control port 0 during each cycle of the controller clock. In other configurations, uncorrectable CA values or a combination of correctable and uncorrectable CA values may be received from two or more control ports.

**[0050]** Bypassing the ECC generator bank 301 potentially reduces memory access latency as removal of that stage in the command pipeline saves at least a clock cycle. Note that, because the interleaving operation in interleave circuit 319 effectively reverses the even/odd splitting effected within splitting circuits 293-296, the incoming CA values may alternatively be routed directly from the CA input source to respective UNCA inputs of multiplexers 321 and 323, bypassing splitter circuits 293-296 (and interleave circuit 319) and thus potentially reducing memory access latency further relative to the error corrected paths.

**[0051]** Figure 6 illustrates a sequence of configuration operations that may be executed by controller component 101 of Figure 1 (and/or another component that controls the controller component) to configure the CA error detection/correction operation of memory system 100. Starting at 401, the controller component determines the memory component population (i.e., number of attached memory components) and characteristics of individual memory components by reading such information from a non-volatile configuration memory (e.g., a relatively small memory co-located with one or memory components within an IC package, on a memory module, or on a printed circuit board, such as a serial presence detect memory, or the like), from

non-volatile storage within the memory component(s) themselves, or from any other configuration data source. At 403, the controller component determines a CA error detection/correction (CA EDC) mode for each installed memory component according to the memory component population and characteristics of individual memory components (e.g., whether a given component supports CA EDC operation and, if so, the specific CA EDC options supported). At 405, the controller component configures the control-side CA error logic in accordance with the CA EDC mode or modes determined at 403, and at 407 the controller component programs each installed memory component to establish the memory component EDC modes determined at 403. As explained above, memory component programming may be effected, for example, by issuing register-write commands (together with mode values to be programmed) to respective memory components via one or more of the CA signaling links (i.e., links 102 in Figure 1), thereby concluding the CA EDC mode configuration within the memory system. As an example of this operation, if the controller component determines (e.g., by reading an SPD or other configuration information source) that the population of attached memory components consists of a uniform set of memory components having respective architectures as shown in Figure 3, the controller component may program its internal configuration register (e.g., element 281) and that of each installed memory component according to the memory component population size. That is, if a single memory component is coupled to the controller component, the controller component may configure itself and the memory component for x1 CA EDC operation, maximizing the ratio of transmitted CA error checking bits to CA bits and thus enabling triple error correction within the memory component. If two memory components are present, the controller component may configure the memory components and itself for x2 CA EDC operation (e.g., DEC-TED on respective blocks of 32 CA bits), and if four memory components are present, the controller component may configure the memory components and itself for x4 CA EDC operation (e.g., DEC-TED on respective blocks of 64 CA bits). The controller component may also disable CA EDC operation permanently or temporarily when instructed to choose the lower latency bypassed CA path instead. In any case, at 409, after CA EDC modes have been established within the various components of the memory system, the controller component issues CA values and corresponding ECC information which may include parity bits, checksum bits, cyclic-redundancy-check (CRC) values or any combination thereof) to individual memory components or to groups of memory components in accordance with the CA EDC mode(s) established therein.

**[0052]** While the memory system of Figure 1 and the memory component and controller components of Figures 3 and 4 have been described in terms of register-configured CA EDC

operation, the CA EDC operation to be formed with respect to a given command may alternatively (or additionally) controlled by information provided within the command itself. For example, information indicating whether to perform CA error detection/correction with respect to a given CA value and/or the nature of the EDC operation to be performed may be pre-pended or appended to the command/address transmission (or embedded in the command itself, for example, within otherwise unused bit fields) or to/within a previous command/address transmission (i.e., each command indicating whether CA EDC is to be carried out with respect to the command that follows). CA EDC information embedded within or appended/pre-pended to a command may include index bits to look-up the CA EDC operation to be performed within a hardwired or programmed lookup table, or may include operation control bits corresponding, for example, to the bypass and CA interface width bits otherwise programmed within the memory-component configuration register 203 shown in Figure 3.

**[0053]** It should be noted that the various circuits disclosed herein may be described using computer aided design tools and expressed (or represented), as data and/or instructions embodied in various computer-readable media, in terms of their behavioral, register transfer, logic component, transistor, layout geometries, and/or other characteristics. Formats of files and other objects in which such circuit expressions may be implemented include, but are not limited to, formats supporting behavioral languages such as C, Verilog, and VHDL, formats supporting register level description languages like RTL, and formats supporting geometry description languages such as GDSII, GDSIII, GDSIV, CIF, MEBES and any other suitable formats and languages. Computer-readable media in which such formatted data and/or instructions may be embodied include, but are not limited to, computer storage media in various forms (e.g., optical, magnetic or semiconductor storage media, whether independently distributed in that manner, or stored "in situ" in an operating system).

**[0054]** When received within a computer system via one or more computer-readable media, such data and/or instruction-based expressions of the above described circuits may be processed by a processing entity (e.g., one or more processors) within the computer system in conjunction with execution of one or more other computer programs including, without limitation, net-list generation programs, place and route programs and the like, to generate a representation or image of a physical manifestation of such circuits. Such representation or image may thereafter be used in device fabrication, for example, by enabling generation of one or more masks that are used to form various components of the circuits in a device fabrication process.

**[0055]** In the foregoing description and in the accompanying drawings, specific terminology and drawing symbols have been set forth to provide a thorough understanding of the present

invention. In some instances, the terminology and symbols may imply specific details that are not required to practice the invention. For example, any of the specific numbers of bits, signal path widths, signaling or operating frequencies, component circuits or devices and the like may be different from those described above in alternative embodiments. Additionally, links or other interconnection between integrated circuit devices or internal circuit elements or blocks may be shown as buses or as single signal lines. Each of the buses may alternatively be a single signal line, and each of the single signal lines may alternatively be buses. Signals and signaling links, however shown or described, may be single-ended or differential. A signal driving circuit is said to “output” a signal to a signal receiving circuit when the signal driving circuit asserts (or deasserts, if explicitly stated or indicated by context) the signal on a signal line coupled between the signal driving and signal receiving circuits. The term “coupled” is used herein to express a direct connection as well as a connection through one or more intervening circuits or structures. Integrated circuit device “programming” may include, for example and without limitation, loading a control value into a register or other storage circuit within the integrated circuit device in response to a host instruction (and thus controlling an operational aspect of the device and/or establishing a device configuration) or through a one-time programming operation (e.g., blowing fuses within a configuration circuit during device production), and/or connecting one or more selected pins or other contact structures of the device to reference voltage lines (also referred to as strapping) to establish a particular device configuration or operation aspect of the device. The terms “exemplary” and “embodiment” are used to express an example, not a preference or requirement.

**[0056]** While the invention has been described with reference to specific embodiments thereof, it will be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope. For example, features or aspects of any of the embodiments may be applied, at least where practicable, in combination with any other of the embodiments or in place of counterpart features or aspects thereof. Accordingly, the specification and drawings are to be regarded in an illustrative rather than a restrictive sense.

What is claimed is:

1. A memory component comprising:  
a signaling interface to receive request information and error-checking information; and  
logic to detect errors within the request information using the error-checking information according to a first error detection scheme in a first operating mode of the memory component and according to a second error detection scheme in a second operating mode of the memory component.
2. The memory component of claim 1 wherein a ratio of error-checking information to request information is higher in the first error detection scheme than in the second error detection/scheme.
3. The memory component of claim 1 wherein the first error detection scheme enables detection of a first number of bit errors within the request information, and the second error detection scheme enables detection of a second number of bit errors within the request information, the second number of bit errors being less than the first number of bit errors.
4. The memory component of claim 1 further comprising a configuration register to store a mode value that specifies one of a plurality of operating modes within the memory component, the plurality of operating modes including the first and second operating modes.
5. The memory component of claim 4 wherein the signaling interface is additionally to receive the mode value and wherein the memory component further comprises control logic to load the mode value into the configuration register.
6. The memory component of claim 1 wherein the request information specifies one of a plurality of operating modes within the memory component, the plurality of operating modes including the first and second operating modes.
7. The memory component of claim 1 further comprising a memory core, and wherein the request information includes an address that specifies a storage region of the memory core and a command that specifies an operation to be carried out with respect to the storage region.
8. The memory component of claim 7 wherein the memory core comprises an array of dynamic random access memory (DRAM) cells.

9. The memory component of claim 1 wherein the first operating mode and the second operating mode correspond to respective first and second widths of the signaling interface.
10. The memory component of claim 1 wherein the signaling interface comprises a control interface to receive the request information via one or more request signaling links, and an error information interface to receive the error-checking information via one or more error information signaling links.
11. The memory component of claim 10 wherein the control interface comprises a variable width, and wherein the control interface is set to a first width to receive the request information via a first number of request signaling links in the first operating mode, and the control interface is set to a second width to receive the request information via a second number of request signaling links in the second operating mode.
12. The memory component of claim 1 further comprising logic to correct errors within the request information detected using the first or second error detection schemes.
13. A controller component for controlling a memory component, the controller component comprising:  
a signaling interface to output request information and error-checking information; and  
logic to generate the error-checking information based on the request information  
    according to a first error detection scheme in a first operating mode of the controller  
    component and according to a second error detection scheme in a second operating  
    mode of the controller component.
14. The controller component of claim 13 wherein the ratio of error-checking information to request information is higher in the first error detection scheme than in the second error detection/scheme.
15. The controller component of claim 13 wherein the first error detection scheme enables a memory component that receives the request information and error-checking information to detect a first number of bit errors within the request information, and the second error detection scheme enables the memory component to detect a second number of bit errors within the request information, the second number of bit errors being less than the first number of bit errors.
16. The controller component of claim 13 wherein the signaling interface is additionally to output a mode value to the memory component to indicate to the memory component



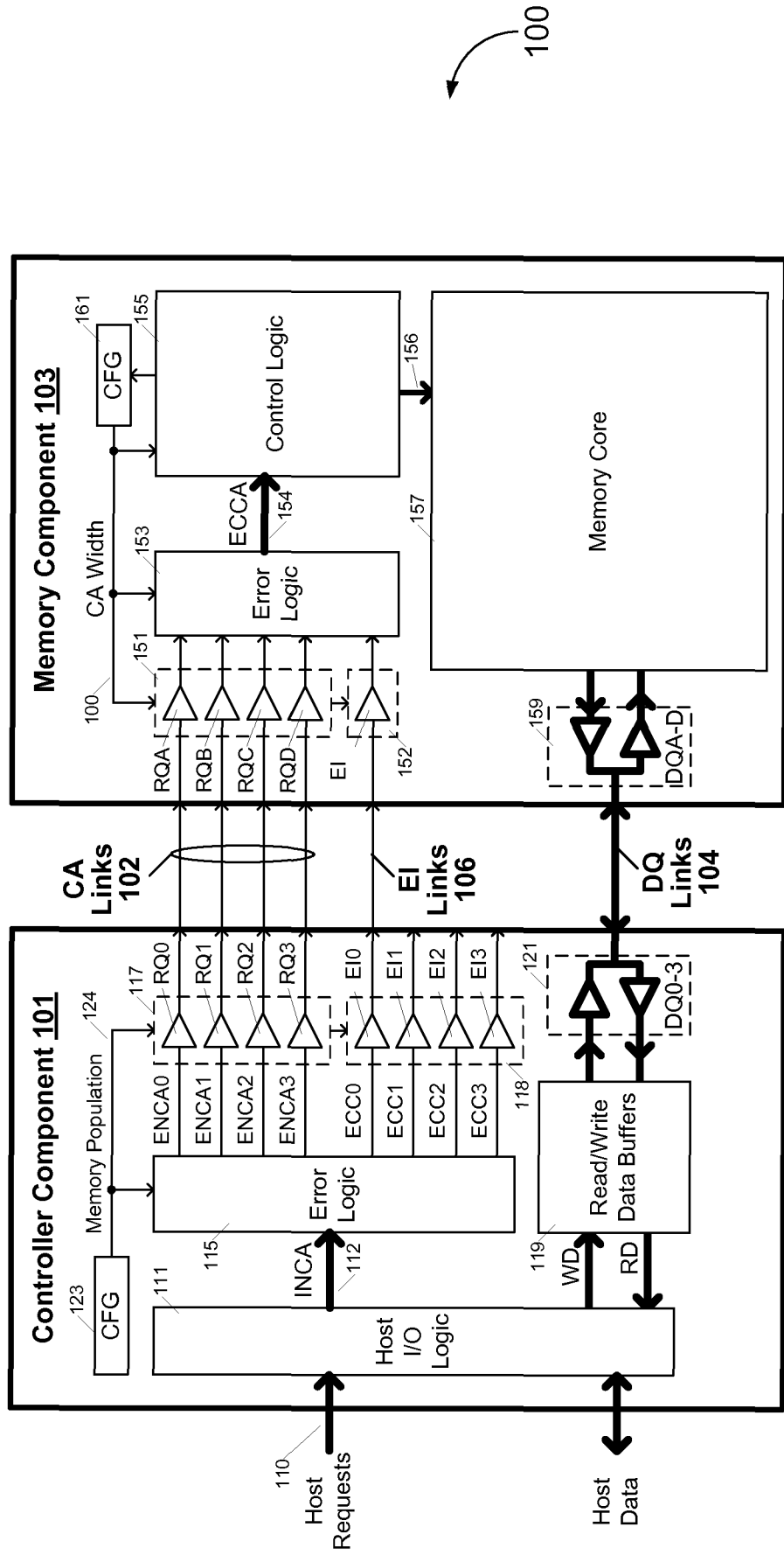
which of the first and second error detection schemes was used to generate the error-checking information.

17. The controller component of claim 13 further comprising a configuration register to store a mode value that specifies one of a plurality of operating modes within the controller component, the plurality of operating modes including the first and second operating modes.
18. The controller component of claim 13 wherein the error-checking information enables correction of errors within the request information.
19. A method of operation within a memory component, the method comprising:  
receiving request information;  
receiving error-checking information; and  
detecting errors within the request information using the error-checking information according to a first error detection scheme in a first operating mode of the memory component and according to a second error detection scheme in a second operating mode of the memory component.
20. The method of claim 19 wherein a ratio of error-checking information to request information is higher in the first error detection scheme than in the second error detection scheme.
21. The method of claim 19 wherein detecting errors according to the first error detection scheme comprises detecting a first number of bit errors within the request information, and detecting errors according to the second error detection scheme comprises detecting a second number of bit errors within the request information, the second number of bit errors being less than the first number of bit errors.
22. The method of claim 19 further comprising storing, within a configuration register of the memory component, a mode value that specifies one of a plurality of operating modes within the memory component, the plurality of operating modes including the first and second operating modes.
23. The method of claim 22 further comprising:  
receiving a register-write command;  
receiving the mode value in association with the register-write command; and  
loading receiving the mode value into the configuration register in response to receiving

the register-write command.

24. The method of claim 19 wherein receiving the request information comprises receiving information that specifies one of a plurality of operating modes within the memory component, the plurality of operating modes including the first and second operating modes.
25. The method of claim 19 further comprising correcting errors within the request information detected according to the first or second error detection schemes.
26. A method of controlling a memory component, the method comprising:  
generating error-checking information based on request information according to a first error detection scheme in a first operating mode and according to a second error detection scheme in a second operating mode; and  
outputting the request information and error-checking information to the memory component.
27. The method of claim 26 wherein the ratio of error-checking information to request information is higher in the first error detection scheme than in the second error detection scheme.
28. The method of claim 26 wherein the first error detection scheme enables the memory component to detect a first number of bit errors within the request information, and the second error detection scheme enables the memory component to detect a second number of bit errors within the request information, the second number of bit errors being less than the first number of bit errors.
29. The method of claim 26 further comprising outputting a mode value to the memory component to indicate to the memory component which of the first and second error detection schemes was used to generate the error-checking information.
30. A memory component comprising:  
means for receiving request information and error-checking information; and  
means for detecting errors within the request information using the error-checking information according to a first error detection scheme in a first operating mode of the memory component and according to a second error detection scheme in a second operating mode of the memory component.

FIG. 1



100

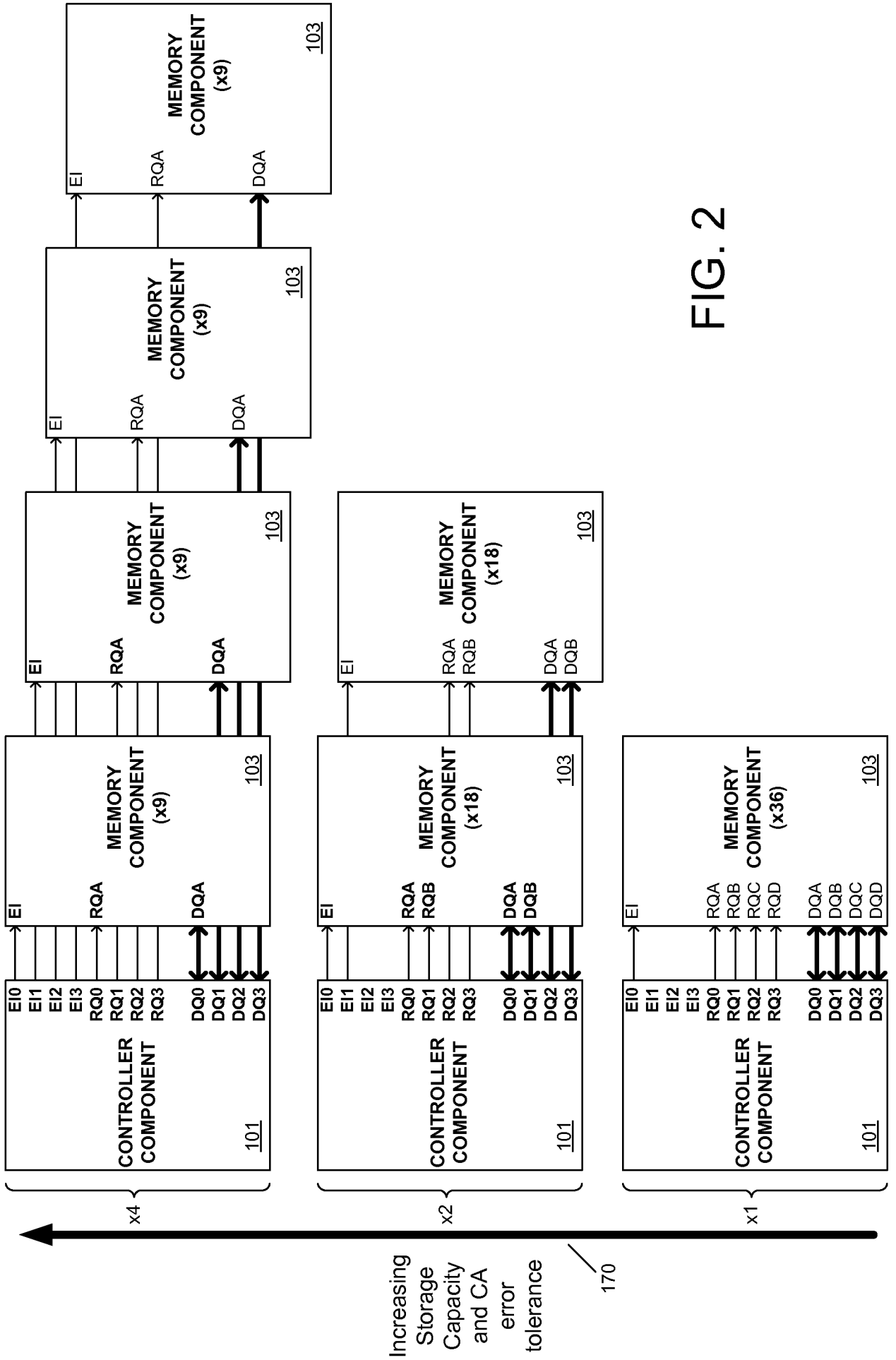


FIG. 2

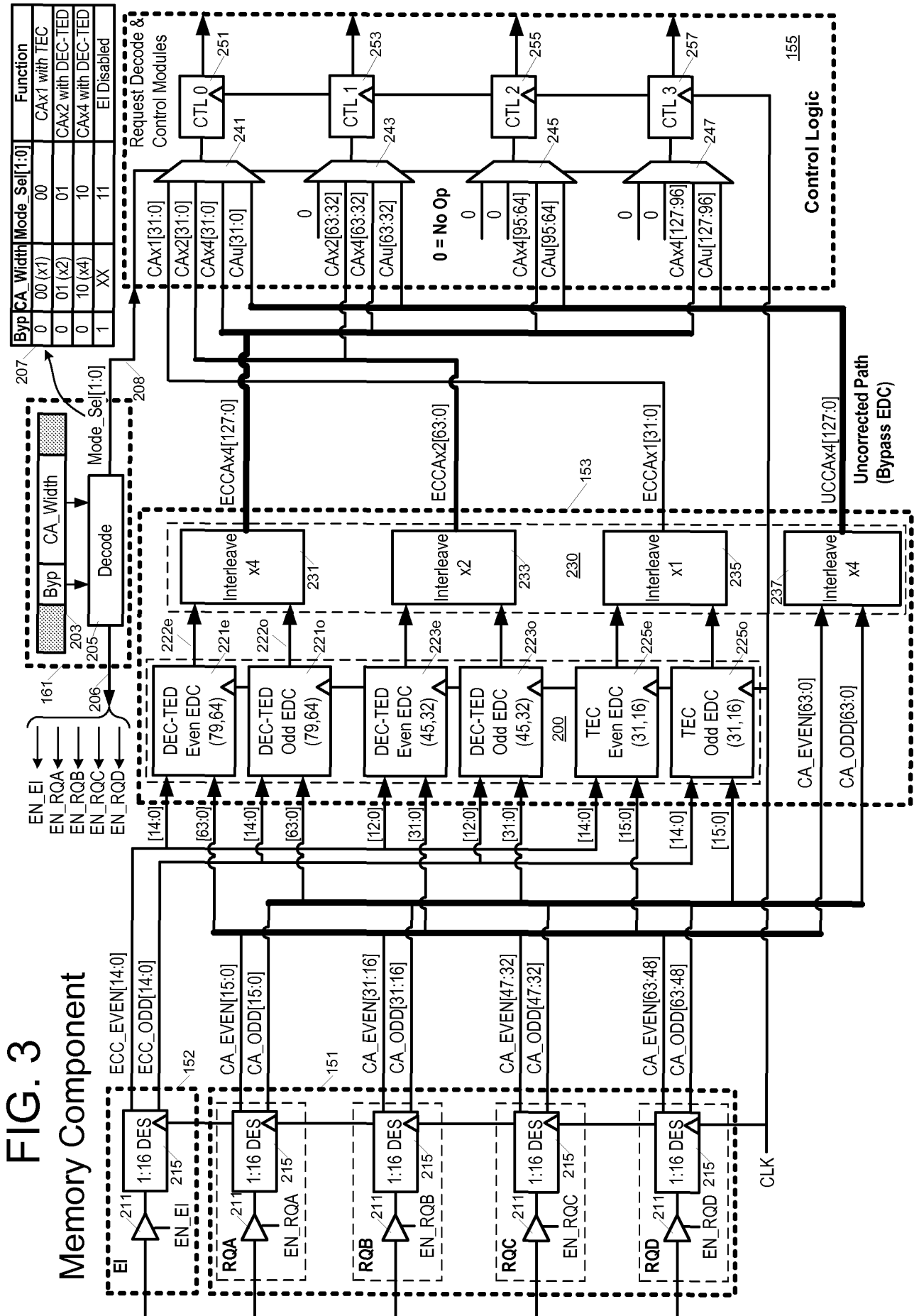
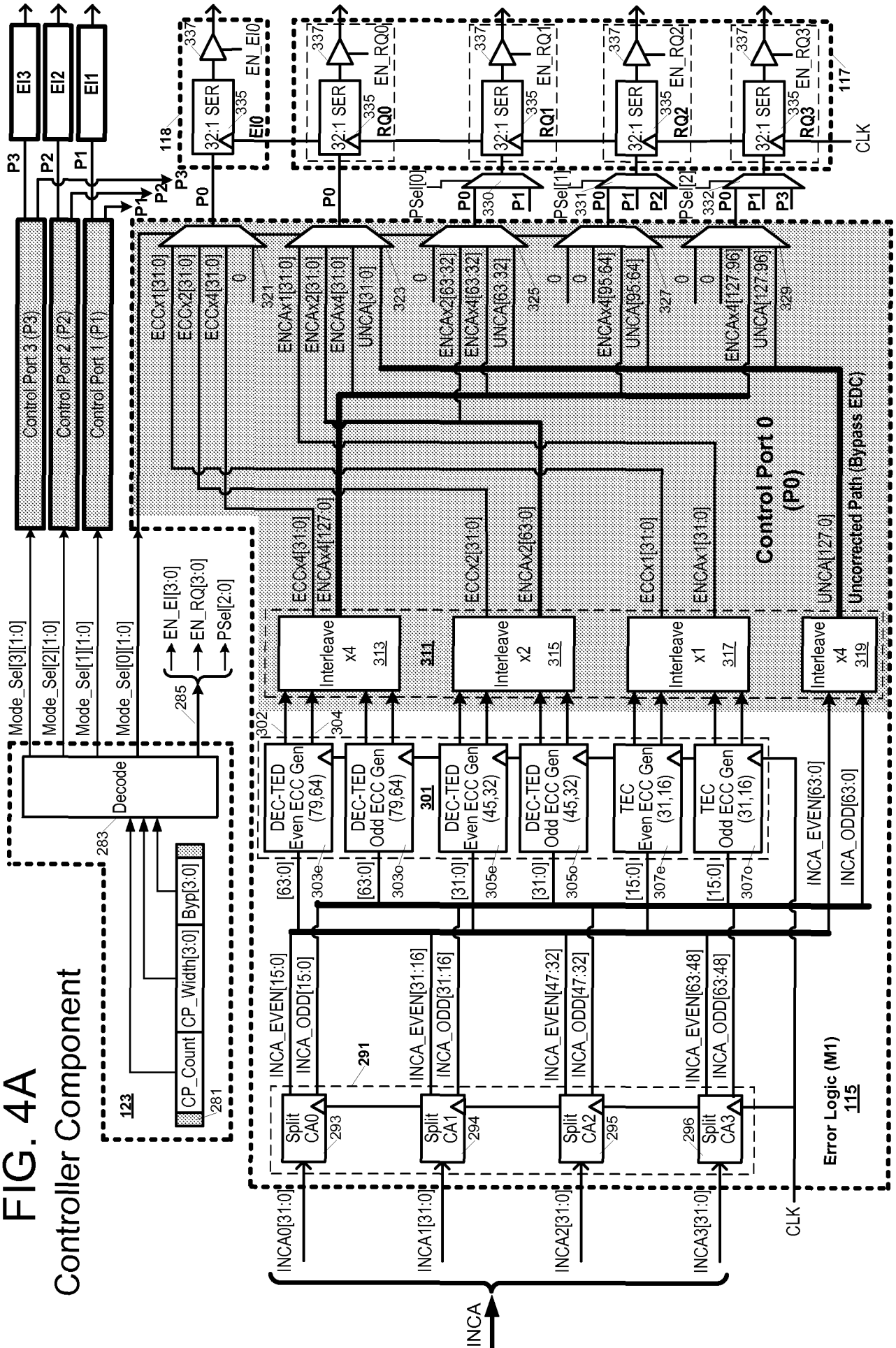


FIG. 4A  
Controller Component



**FIG. 4B**

**Control Port Selection**

CP Count	CP Width (P3:P2:P1:P0)	Control Port Selection for RQ1 (Psel[0])	Control Port Selection for RQ2 (Psel[1])	Control Port Selection for RQ3 (Psel[1])
1	01011	xx	xx	xx
	01012	P0	xx	xx
	01014	P0	P0	P0
	01011	P1	xx	xx
2	01012	P0	P1	xx
	01022	P0	P1	P1
	01111	P1	P2	xx
	01112	P0	P1	P2
4	11111	P1	P2	P3

**FIG. 4C**

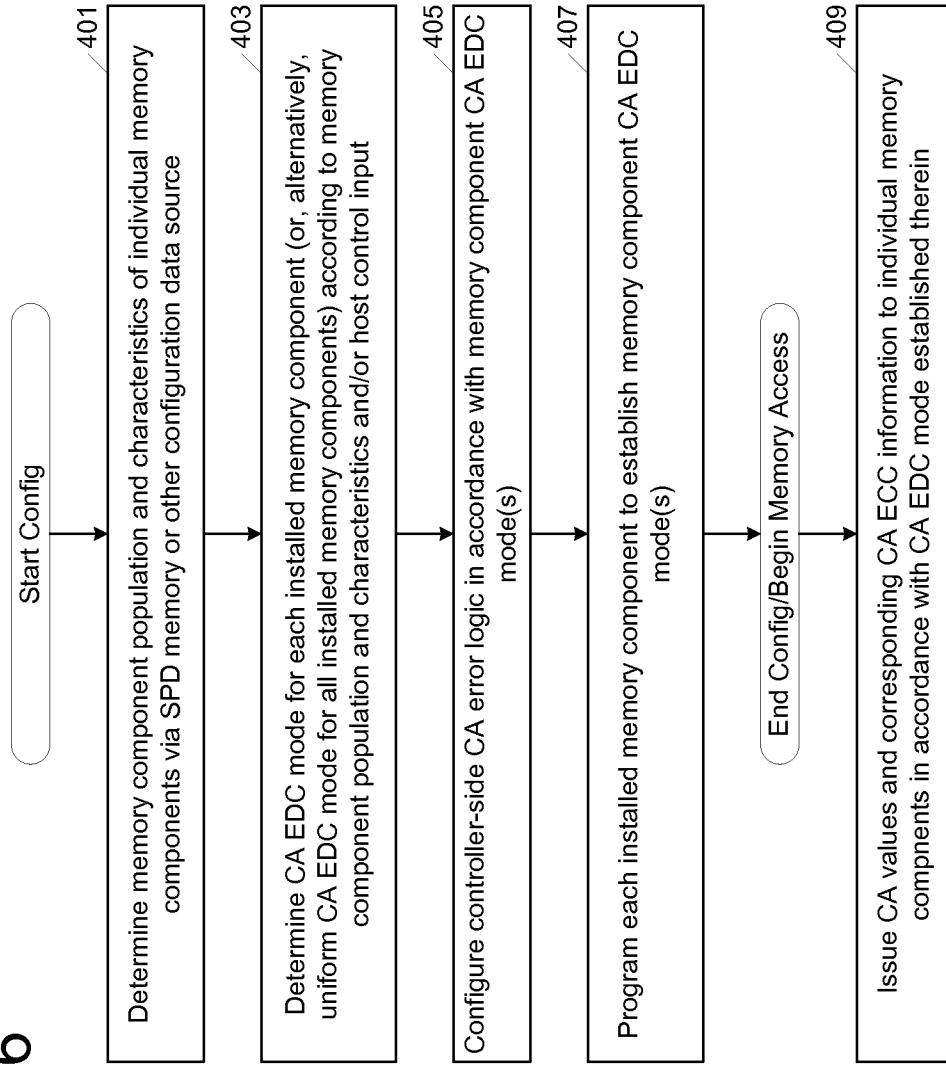
**Unbypassed, Control Port and CA Interface Configuration**

CP Count	CP Width	Control Port 0 Mode_Sel[0][1:0]	Control Port 1 Mode_Sel[1][1:0]	Control Port 2 Mode_Sel[2][1:0]	Control Port 3 Mode_Sel[3][1:0]	RQ0	RQ1	RQ2	RQ3
1	01011	00: CAX1 with TEC	xx: (unused)	xx: (unused)	xx: (unused)	P0:ENCAX1[31:0]	disabled	disabled	disabled
	01012	01: CAX2 with DEC-TED	xx: (unused)	xx: (unused)	xx: (unused)	P0:ENCAX2[31:0]	P0:ENCAX2[63:32]	disabled	disabled
	01014	10: CAX4 with DEC-TED	xx: (unused)	xx: (unused)	xx: (unused)	P0:ENCAX4[31:0]	P0:ENCAX4[63:32]	P0:ENCAX4[96:64]	P0:ENCAX4[127:96]
	01011	00: CAX1 with TEC	00: CAX1 with TEC	xx: (unused)	xx: (unused)	P0:ENCAX1[31:0]	P1:ENCAX1[31:0]	disabled	disabled
2	01012	01: CAX2 with DEC-TED	00: CAX1 with TEC	xx: (unused)	xx: (unused)	P0:ENCAX2[31:0]	P0:ENCAX2[63:32]	P1:ENCAX1[31:0]	disabled
	01022	01: CAX2 with DEC-TED	01: CAX2 with DEC-TED	xx: (unused)	xx: (unused)	P0:ENCAX2[31:0]	P0:ENCAX2[63:32]	P1:ENCAX2[31:0]	P1:ENCAX2[63:32]
	01111	00: CAX1 with TEC	00: CAX1 with TEC	00: CAX1 with TEC	xx: (unused)	P0:ENCAX1[31:0]	P1:ENCAX1[31:0]	P2:ENCAX1[31:0]	disabled
	01112	01: CAX2 with DEC-TED	00: CAX1 with TEC	00: CAX1 with TEC	xx: (unused)	P0:ENCAX2[31:0]	P0:ENCAX2[63:32]	P1:ENCAX1[31:0]	P2:ENCAX1[31:0]
4	11111	00: CAX1 with TEC	00: CAX1 with TEC	00: CAX1 with TEC	00: CAX1 with TEC	P0:ENCAX1[31:0]	P1:ENCAX1[31:0]	P2:ENCAX1[31:0]	P3:ENCAX1[31:0]





FIG. 6



**A. CLASSIFICATION OF SUBJECT MATTER***G06F 13/14(2006.01)i, G06F 13/16(2006.01)i, G06F 11/08(2006.01)i, G06F 12/00(2006.01)i*

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)

G06F 13/14; G11C 16/04; G06F 11/14; G06F 12/00

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

Korean utility models and applications for utility models  
Japanese utility models and applications for utility models

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

eKOMPASS(KIPO internal) &amp;

Keywords:memory,controller,component,register,buffer,error,check\*,correct\*,detect\*,informat\*,operat\*,request,logic,bit,interface,DRA  
M cell**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2007-0109858 A1 (KEVIN CONLEY et al.) 17 May 2007 See Abstract, Paragraphs [0006]-[0011], [0045] and Figures 1, 5.	1-30
A	US 2009-0187785 A1 (GONZALEZ CARLOS J. et al.) 23 July 2009 See Abstract, Paragraphs [0035]-[0038], and Figures 1A, 1B.	1-30
A	US 2004-0049629 A1 (SEIJI MIURA et al.) 11 March 2004 See Abstract, Paragraphs [0053]-[0079], [0236], and Figures 1, 22.	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 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

"&amp;" document member of the same patent family


Date of the actual completion of the international search

15 OCTOBER 2012 (15.10.2012)

Date of mailing of the international search report

**16 OCTOBER 2012 (16.10.2012)**

Name and mailing address of the ISA/KR

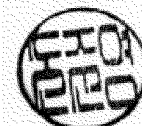

 Korean Intellectual Property Office  
189 Cheongsu-ro, Seo-gu, Daejeon Metropolitan  
City, 302-701, Republic of Korea

Facsimile No. 82-42-472-7140

Authorized officer

Son, June Young

Telephone No. 82-42-481-5720



**INTERNATIONAL SEARCH REPORT**

Information on patent family members

International application No.

**PCT/US2012/028658**

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2007-0109858 A1	17.05.2007	AU 2001-297851 A8	16.12.2002
		CN 100474452 C	01.04.2009
		CN 1620703 A	25.05.2005
		CN 1620703 C0	25.05.2005
		EP 1488429 A2	22.12.2004
		EP 1488429 B1	13.02.2008
		JP 04-129428 B2	06.08.2008
		JP 2005-507129 A	10.03.2005
		JP 2007-257831 A	04.10.2007
		KR 10-0921763 B1	15.10.2009
		TW 1241590B	11.10.2005
		TW 241590 A	11.10.2005
		TW 241590 B	11.10.2005
		US 2002-0126528 A1	12.09.2002
		US 2003-174555 A1	18.09.2003
		US 2005-0195653 A1	08.09.2005
		US 6349056 B1	19.02.2002
		US 6560143 B2	06.05.2003
		US 6972993 B2	06.12.2005
		US 7170782 B2	30.01.2007
		US 7376011 B2	20.05.2008
WO 02-099806 A2	12.12.2002		
WO 02-099806 A3	12.12.2002		
US 2009-0187785 A1	23.07.2009	CN 100541439 C	16.09.2009
		CN 101630279 A	20.01.2010
		CN 1882918 A	20.12.2006
		CN 1882918 C0	20.12.2006
		EP 1687720 A2	09.08.2006
		EP 1687720 B1	02.01.2008
		EP 1847930 A1	24.10.2007
		EP 1847930 B1	24.06.2009
		EP 1941368 A1	09.07.2008
		JP 04-723504 B2	15.04.2011
		JP 2007-507804 A	29.03.2007
		JP 2009-512119 A	19.03.2009
		JP 4723504 B2	13.07.2011
		KR 10-1127882 B1	21.03.2012
		KR 10-2008-0066959 A	17.07.2008
		US 2005-0073884 A1	07.04.2005
		US 2006-0039196 A1	23.02.2006
		US 2006-0062048 A1	23.03.2006
		US 2007-0211532 A1	13.09.2007
		US 2011-0055468 A1	03.03.2011
		US 7012835 B2	14.03.2006
US 7173852 B2	06.02.2007		
US 7224607 B2	29.05.2007		
US 7518919 B2	14.04.2009		
US 8004895 B2	23.08.2011		

**INTERNATIONAL SEARCH REPORT**

Information on patent family members

International application No.

**PCT/US2012/028658**

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
		US 8050095 B2	01.11.2011
		WO 2005-036401 A2	21.04.2005
		WO 2005-036401 A3	21.04.2005
		WO 2007-047110 A1	26.04.2007
US 2004-0049629 A1	11.03.2004	CN 100433192 C0	12.11.2008
		CN 101281494 A	08.10.2008
		CN 101281494 C0	08.10.2008
		CN 1482619 A	17.03.2004
		CN 1482619 C0	12.11.2008
		JP 04-499982 B2	23.04.2010
		JP 2004-102781 A	02.04.2004
		JP 4499982 B2	14.07.2010
		KR 10-0940163 B1	03.02.2010
		KR20040023486A	18.03.2004
		TW 1287795B	01.10.2007
		TW 287795 A	01.10.2007
		TW 287795 B	01.10.2007
		US 7136978 B2	14.11.2006