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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 July 2004 (29.07.2004)

PCT

(10) International Publication Number
WO 2004/063906 A2

(51) International Patent Classification[7]: G06F

(21) International Application Number:
PCT/US2004/000793

(22) International Filing Date: 13 January 2004 (13.01.2004)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/439,666	13 January 2003 (13.01.2003)	US
10/386,236	11 March 2003 (11.03.2003)	US
10/385,908	11 March 2003 (11.03.2003)	US

(71) Applicant: **RAMBUS INC.** [US/US]; 4440 El Camino Real, Los Altos, CA 94022 (US).

(72) Inventors: **PEREGO, Richard, E.**; 5938 Pala Mesa Drive, San Jose, CA 95123 (US). **WARE, Federick, A.**; 13961 Fremont Pines Lane, Los Altos Hills, CA 94022 (US).

(74) Agent: **DENIRO, Kirk, J.**; Vierra Magen Marcus Harmon & Deniro LLP, 685 Market Street, Suite 540, San Francisco, CA 94105 (US).

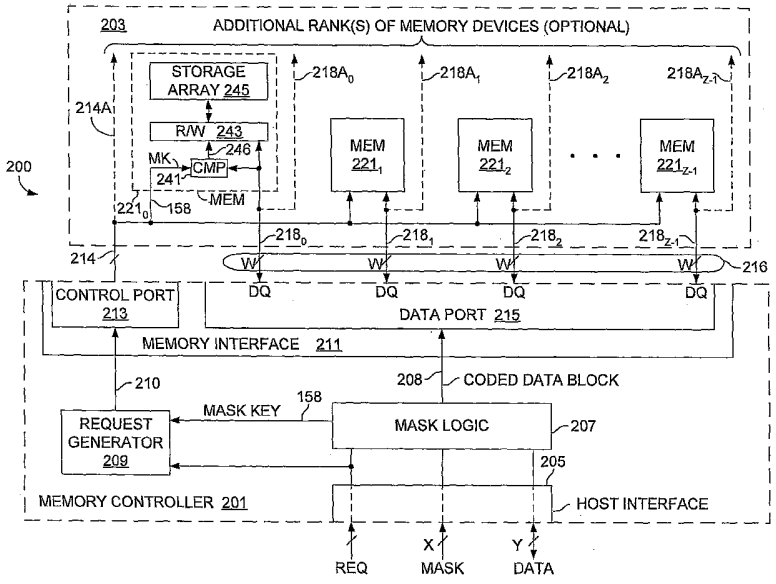
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR,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

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

[Continued on next page]

(54) Title: CODED WRITE MASKING



(57) Abstract: A memory system having a memory controller and a memory device coupled to the memory controller. The memory controller outputs a write data value to the memory device. The memory device receives the write data value from the memory controller, and compares the write data value with a mask key value. If the write data value matches the mask key value, the memory device does not store the write data value. If the write data value does not match the mask key value, the memory device stores the write data value.

WO 2004/063906 A2

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

CODED WRITE MASKING

CROSS-REFERENCE TO RELATED APPLICATIONS

The present application claims priority to United States Provisional Patent Application No. 60/439,666 filed January 13, 2003, entitled "Coded Write Masking", by inventors Richard E. Perego and Frederick A. Ware; United States Utility Patent Application No. 10/386,236 filed March 11, 2003, entitled "Coded Write Masking", by inventors Richard E. Perego and Frederick A. Ware; and United States Utility Patent Application No. 10/385,908 filed March 11, 2003, entitled "Memory System And Device With Serialized Data Transfer", by inventors Richard E. Perego and Frederick A. Ware.

FIELD OF THE INVENTION

[0001] The present invention relates generally to data transfer and storage technology, and more particularly to masked write operations in memory systems and devices that access memory systems.

BACKGROUND

[0002] Read and write accesses in modern memory systems are typically transacted through transfers of multi-byte blocks of data. When less than a full data block is to be read from the memory system, the address of a data block that encompasses the desired data is provided to the memory system and the full data block is read. Generally speaking, there is no penalty to reading more data than needed. By contrast, when writing a value smaller than a full data block, it is important that the stored data block remain unchanged except for the value written. This is typically accomplished through one of two types of specialized write operations: merged write operations or masked write operations.

[0003] In a merged write operation (sometimes called a read-merge-write operation or read-modify-write operation), a memory controller reads the data block to be updated, merges the write data value into the data block at the appropriate offset, then writes the updated data block back to storage. Because two memory accesses are required (read and write), merged write operations

substantially reduce peak bandwidth of the memory system and therefore are typically not used in high performance systems.

[0004] In a masked write operation, the memory controller issues mask signals to the storage subsystem to qualify each data value within the data block as being masked or unmasked. The storage subsystem responds by storing only unmasked data values. For legacy purposes, the granularity of data masking typically extends to byte (i.e., eight-bit) granularity. Data masking with eight bit or byte granularity is sometimes referred to as byte-masking. While byte-masking has the disadvantage of requiring additional hardware in the storage subsystem (i.e., to detect and respond to the mask signals), the double-access performance penalty associated with merged write operations is avoided.

[0005] Figure 1 illustrates a prior art byte-masking memory system 100 having a memory controller 101 and a storage subsystem 103. The memory controller 100 includes a host interface 105 to receive access requests (REQ), data blocks (DATA) and mask information (MASK) from an access requestor, and a memory interface 107 to issue corresponding requests, data blocks and mask information to the storage subsystem. In a masked write operation, a 64-byte write data block (512 bits) is received via the host interface 105 along with a 64-bit mask word and a masked-write request. Each bit of the mask word corresponds to a respective byte of the write data block and, if set, indicates that the byte is a masked byte not to be stored within the storage subsystem. The memory controller 101 responds to the masked-write request by issuing a masked-write instruction to the storage subsystem 103 via a request path 102, and by transferring the write data block and mask word to the storage subsystem via a data path 104. The data path includes 32 data lines 108 for parallel transfer of four data bytes and four mask lines 106 for transferring four corresponding mask bits. Consequently, the complete write data block and mask word are transferred to the storage subsystem in a sequence of sixteen data transfers, each transfer including four bytes of the write data block and four bits of the mask word.

[0006] The storage subsystem 103 is formed by a number of discrete memory devices, MEM_{R1}, each having a request interface and a data interface. The request

interface of each memory device is coupled to the request path 102 to receive the masked-write instruction (including an address value), and the data interface of each memory device is coupled to a respective 9-bit slice of the data path to receive a data byte and corresponding mask bit in each of the sixteen data transfers. For each data transfer, each of the memory devices stores the data byte at a location indicated by the address value (offset according to which of the sixteen data transfers is being acted on) only if the mask bit is not set.

[0007] One drawback to the prior-art memory system 100 is that a substantial portion of the data path 104, one line out of every nine, is dedicated to mask signal transfer. Thus, more than 10% of the data path bandwidth is reserved to support byte masking. This bandwidth penalty becomes worse as the device width (i.e., the width of the memory device data interface excluding the mask input) is reduced. For example, if the device width is reduced from eight bits to four bits, then 20% of the data path bandwidth (one out every five signal lines) is reserved for byte masking. Thus, in addition to imposing a substantial bandwidth penalty, the byte masking technique used in the prior-art memory system 100 effectively constrains the device widths of the memory devices within the storage subsystem 103 to be at least eight bits. This device width constraint translates directly into a memory size constraint for a given generation of memory devices and data path width. For example, assuming storage capacity of 512 megabits (Mb) for a given generation of memory devices and a data path width of 32 lines (excluding mask lines), the total size of memory that be coupled in point-to-point fashion to the memory controller is 512Mb*(32/8) = 2 Gigabits (Gb). While an additional group of memory devices may be coupled to the data path 104, as shown in dashed outline in Figure 1 by devices MEM_{R2}, the additional signal line connections effectively transform the data path 104 into a multi-drop bus. Multi-drop bus arrangements have different, and sometimes significantly less desirable, signaling characteristics than point-to-point arrangements.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] 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:

Figure 1 illustrates a prior art byte-masking memory system;

Figure 2 illustrates a method of operation within a memory controller according to an embodiment of the invention;

Figure 3 illustrates a method of operation within a storage subsystem according to an embodiment of the invention;

Figure 4 illustrates a memory system according to an embodiment of the invention;

Figure 5 illustrates a memory device according to an embodiment of the invention;

Figure 6 illustrates the mask logic of Figure 4 according to an embodiment of the invention;

Figure 7 illustrates an exemplary pipeline for propagation of write data blocks through the memory controller of Figure 4;

Figure 8 illustrates the operation of the key generator of Figure 6 according to an embodiment of the invention;

Figure 9 illustrates a key generator according to an embodiment of the invention;

Figure 10 illustrates an exemplary timing diagram for the transfer of byte-sized write data values over a data path between the memory controller and storage subsystem of Figure 4;

Figure 11 illustrates the storage capacity that can be achieved in a single-rank of memory devices using the data transfer scheme of Figure 10;

Figure 12 illustrates a data transfer scheme in an alternative embodiment of the invention that enables the per-rank storage capacity of the storage subsystem to be substantially increased;

Figure 13 illustrates the storage capacity that can be achieved in a single-rank of memory devices using the data transfer scheme of Figure 12;

Figure 14 illustrates the data port of a serial-transfer memory controller according to an embodiment of the invention;

Figure 15 illustrates an exemplary embodiment of a serializing circuit that may be used within the data port of Figure 14;

Figure 16 illustrates a serial-transfer memory device according to an embodiment of the invention;

Figure 17 illustrates the operation of a key selecting memory controller according to an embodiment of the invention;

Figure 18 illustrates a two-phase masked-write operation applied to an exemplary write data block and corresponding mask word;

Figure 19 illustrates the content of a storage subsystem before and after each phase of the two-phase masked-write operation of Figure 18;

Figure 20 illustrates a key-selecting mask logic circuit according to an embodiment of the invention;

Figure 21 illustrates an exemplary state diagram implemented by the key select circuit of Figure 20;

Figure 22 illustrates the key select circuit of Figure 20 according to an embodiment of the invention;

Figure 23 illustrates a memory device according to a key-selecting embodiment of the invention;

Figure 24 illustrates an exemplary spatial partitioning of a write data block into two sub-blocks;

Figure 25 illustrates an exemplary temporal partitioning of a write data block into two sub-blocks; and

Figure 26 illustrates an exemplary encoding of a key selector that may be used within a key-selecting memory system of the present invention.

DETAILED DESCRIPTION

[0009] In the following description and in the accompanying drawings, specific terminology and drawing symbols are 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,

the interconnection between circuit elements or circuit blocks may be shown or described as multi-conductor or single conductor signal lines. Each of the multi-conductor signal lines may alternatively be single-conductor signal lines, and each of the single-conductor signal lines may alternatively be multi-conductor signal lines. Signals and signaling paths shown or described as being single-ended may also be differential, and vice-versa. Similarly, signals described or depicted as having active-high or active-low logic levels may have opposite logic levels in alternative embodiments. As another example, circuits described or depicted as including metal oxide semiconductor (MOS) transistors may alternatively be implemented using bipolar technology or any other technology in which a signal-controlled current flow may be achieved. With respect to terminology, a signal is said to be "asserted" when the signal is driven to a low or high logic state (or charged to a high logic state or discharged to a low logic state) to indicate a particular condition. Conversely, a signal is said to be "deasserted" to indicate that the signal is driven (or charged or discharged) to a state other than the asserted state (including a high or low logic state, or the floating state that may occur when the signal driving circuit is transitioned to a high impedance condition, such as an open drain or open collector condition). 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. A signal line is said to be "activated" when a signal is asserted on the signal line, and "deactivated" when the signal is deasserted. Additionally, the prefix symbol "/" attached to signal names indicates that the signal is an active low signal (i.e., the asserted state is a logic low state). A line over a signal name (e.g., '$\overline{<\text{signal name}>}$') is also used to indicate an active low signal.

[0010] A write-masking memory system in which a key value is substituted for masked data values within an incoming write data block is disclosed herein in various embodiments. In one embodiment, the key value is dynamically generated by the memory system, then substituted for each masked data value to produce a coded data block. The coded data block and key value are each output to a storage

subsystem that includes one or more discrete memory devices. The discrete memory devices compare the key value with the constituent write data values of the coded data block and store only those write data values determined not to match the key value. Because the key value is used to identify masked data values, transmission of dedicated mask signals (i.e., mask signal per write data value as in the memory system of Figure 1) is unnecessary, making the full data path between memory controller and storage subsystem available for write data transfer and achieving a substantial bandwidth savings over the prior art memory system of Figure 1.

[0011] In an alternative embodiment of the invention, the memory controller selects a key value from a table of predetermined values, then passes a selector value to the storage subsystem to indicate which of the predetermined values was selected. Each of the discrete memory devices within the storage subsystem includes the table of predetermined values and uses the selector value to select the selector-specified key value. The selected key value is then used to identify masked data values so that they are excluded from storage. Because the selector value may be significantly smaller than a key value, additional bandwidth savings are achieved.

[0012] Although the invention is described below primarily in reference to a memory system embodiment having a memory controller and storage subsystem, the techniques and structures described may be applied elsewhere within a processing system. For example, a processor or other memory requestor may perform the key value substitution in a write data block and transfer the resulting coded data block and key value to a memory controller. The memory controller may then forward the key value and coded data block to the storage subsystem to complete the masked write operation.

Memory System with Coded Write Masking

[0013] Figure 2 illustrates a method of operation within a memory controller according to an embodiment of the invention. At 153, the memory controller receives a write data block 150 containing X write data values, WD0-WD(X-1), and a corresponding mask word 152 containing X mask values, M0-M(X-1), each

mask value corresponding to a respective one of the write data values. The write data values may be any size (e.g., 8-bit, 16-bit, 32-bit, etc.) but, at least in the embodiment of Figure 2, include a sufficient number of constituent bits to ensure that there are at least X possible bit patterns. That is, if the write data value size is N bits, then N is large enough to ensure that $2^N \geq X$. This arrangement ensures that there is at least one pattern of N bits that will not match any of the X-1 possible bit patterns of the unmasked data values in a masked-write operation (i.e., because at least one of the X write data values in the write data block is masked). For the remainder of this description, the mask values are assumed to be single-bit values and are referred to as mask bits, though multi-bit mask values may alternatively be used.

[0014] At 155, the memory controller generates a key value, referred to herein as a mask key 158, having N constituent bits arranged in a pattern that does not match any of the unmasked data values. At 157, the mask key is substituted for each masked data value in the write data block to produce a coded data block 162. For example, as shown at 154, bits M1 and M(i) of the mask word are set (e.g., to a logic '1' value), indicating that the corresponding write data values are masked data values. Accordingly, the mask key 158 is substituted for the masked data values to generate the coded data block 162. At 159, the coded data block and mask key are transmitted to the storage subsystem.

[0015] Figure 3 illustrates a method of operation within the storage subsystem referred to in Figure 2 according to an embodiment of the invention. At 175, the storage subsystem receives the coded data block referred to in Figure 2, and a masked-write request that includes a masked-write command, the mask key referred to in Figure 2, and the address of a target storage location within the storage subsystem (e.g., starting address of the range of storage locations to be written). As discussed below, the command, mask key and/or address may be multiplexed onto a set of request lines or may be received via respective sets of signal lines. The coded data block may be received before, after or concurrently with the masked-write request or any portion thereof.

[0016] At 177, each write data value within the coded data block is compared with the mask key. If a given write data value does not match the mask key, then at 179 the write data value is stored within the storage subsystem at the address specified in the masked-write request. If the write data value matches the mask key, the write data value is excluded from storage within the storage subsystem, thereby achieving the selective storage function of the masked-write operation.

[0017] Figure 4 illustrates a memory system 200 according to an embodiment of the invention. The memory system 200 includes a memory controller 201 and storage subsystem 203 coupled to one another via a control path 214 and a data path 216. The memory controller 201 includes a host interface 205, memory interface, mask logic circuit 207 and request generator 209.

[0018] The host interface 205 is coupled to receive access requests, data and mask information from one or more access requestors (e.g., a processor, application specific integrated circuit (ASIC), DMA controller, or other requestor). When an access request specifies a masked-write operation, a write data block (DATA) having Y constituent bits and a mask word (MASK) having X constituent bits is received in the memory controller via the host interface. Thus, each mask bit corresponds to a respective group of Y/X bits of the write data block (i.e., mask granularity = Y/X). Each such maskable group of Y/X bits is referred to herein as a write data value, and the write data block is referred to as having X constituent write data values. As an example, if Y=512 (512-bit write data block) and X=64 (64-bit mask word), then the write data block includes 64 byte-sized write data values. As another example, if Y=512 and X = 16, then the write data block is deemed to include sixteen 32-bit write data values. Numerous other values of Y and X may be used.

[0019] Note that the data portion of the host interface is depicted as being coupled to Y data lines for purposes of example only. The data portion of the host interface may alternatively be coupled to more or fewer than Y data lines and the Y-bit write data block received in a sequence of write data transfers. Similarly, the mask portion of the host interface may be coupled to more or fewer than X mask signal lines and the X-bit mask word received in a sequence of mask transfers.

[0020] The masked-write request, write data block and mask word are supplied to the mask logic 207 which, in turn, generates a mask key 158 and coded data block 208. The coded data block 208 is supplied to a data port 215 within the memory interface, and the mask key 158 is supplied to the request generator 209. The request generator 209 also receives the masked-write request from the host interface and, in response, outputs corresponding control information 210 to a control port 213 within the memory interface 211, the control information including the mask key 158 and a write address indicated by the masked-write request. The control port includes a bank of output drivers (not specifically shown in Figure 4) to output the control information 210 to the storage subsystem via the control path 214. The data port similarly includes a bank of output drivers to output the coded data block 208 to the storage subsystem via the data path.

[0021] The storage subsystem 203 includes a group of Z discrete memory devices 221_0-221_{Z-1}, each coupled to the control path 214 and to a respective portion of the data path 216 (i.e., as indicated by 218_0-218_{Z-1}). By this arrangement, each of the memory devices 221 receives the control information 210 and a respective portion of the coded data block 208 from the memory controller 201. The number of data lines, W, coupled to each of the memory devices 221 is an integer multiple of the mask granularity (i.e., W=k(Y/X), where k = 1, 2, 3...) so that each of the memory devices receives at least one complete write data value in the form of a parallel set of bits (i.e., each bit of the write data value is received on a respective data line). In alternative embodiments, the number of data lines, W, need not be an integer multiple of the mask granularity.

[0022] Memory device 221_0 is shown in expanded view to illustrate a key-based write masking operation. The memory device 221_0, like each of the memory devices 221, includes a storage array 245, read/write circuit 243 and compare circuit 241. Each write data value received via data lines 218 is provided to the compare circuit 241 and the read/write circuit 243. A mask key 158 received via the control path 214 (or alternatively via the data path 216) is also provided to the compare circuit 241 for comparison with the incoming write data value. If the write data value matches the mask key 158, the compare circuit 241 asserts a mask

signal 246 to prevent the read/write circuit 243 from writing the write data value into the storage array 245. If the write data value does not match the mask key, the compare circuit deasserts the mask signal 246 to enable the read/write circuit 243 to write the data value into the storage array 245.

[0023] Still referring to Figure 4, the memory devices 221_0-221_{Z-1} collectively form an addressable unit of memory referred to herein as a memory rank. Each memory device within a memory rank shares the same physical address range such that, when an address is supplied on the control path 214 for a read or write access, all the memory devices within the rank are accessed at the specified address (or an address offset from the specified address). As shown in Figure 4, one or more additional ranks of memory devices may be included within the storage subsystem 203 to increase the overall storage capacity. In one embodiment, each additional rank of memory devices is coupled in common to the control path 214 as indicated by dashed arrow 214A, and each memory device of each additional rank is coupled to a respective portion of the data path as shown by arrows $218A_0$-$218A_{Z-1}$.

[0024] Figure 5 illustrates an exemplary memory device 260 that may be used to implement each of the discrete memory devices 221 of Figure 4. The memory device 260 includes a control interface 261, data interface 263, request logic 265, compare circuit 241, read/write circuit 243, address decoder 267 and storage array 245. The storage array 245 includes storage cells arranged in rows and columns, with each row of storage cells coupled to the address decoder via a respective word line 270 and each column of storage cells coupled to the read/write circuit 243 via a respective bit line 272 (or pair of bit lines in the case of differentially coupled storage cells). During a read or write access, an address 274 is received from the memory controller via the control interface 261 and supplied to the address decoder 267. The address decoder 267 decodes the address to activate a selected word line 270 (i.e., the word line specified by the address), thereby enabling read and write access to the row of storage cells coupled to the word line. In a read operation, activation of a selected word 270 line results in the data within the corresponding storage cells being supplied, via the bit lines 272, to a sense amplifier bank (not shown in Figure 5) within the read/write circuit 243. The sense amplifier bank

amplifies the relatively low level signals on the bit lines 272 to logic levels (e.g., complementary MOS levels) and outputs the amplified signals to a set of output drivers (not shown in Figure 5) within the data interface 263 for eventual transmission to the memory controller.

[0025] In a write operation, write data values are received via receive circuits 271 within the data interface 263 and loaded into respective write drivers 273_0-273_{K-1} within the read/write circuit 243. When a write strobe signal 248 (WR) is asserted by the request logic 265, the write drivers 273 output the write data values onto the bit lines 272 for storage within the row of storage cells coupled to an activated word line 270. In the embodiment of Figure 5, logic AND gates 275_0-275_{K-1} are provided to selectively block assertion of the write strobe signal 248 at respective enable inputs (EN) of the write drivers 273_0-273_{K-1}. Each AND gate 275 includes a non-inverting input coupled to receive the write strobe signal 248 and an inverting input coupled to receive a respective one of match signals 252_0-252_{K-1} from the compare circuit 241. Referring to AND gate 275_0, when match signal 252_0 is asserted (e.g., to a logic high state), the output of AND gate 275_0 remains low even as the write strobe signal 248 goes high, thereby preventing write driver 273_0 from outputting a write data value to the storage array 245. When the match signal 252_0 is deasserted, the AND gate 275_0 passes the write strobe signal 248 to the enable input of the write driver 273_0, enabling the write driver 273_0 to output a write data value to the storage array 245.

[0026] The compare circuit 241 includes a number of comparators, 269_0-269_{K-1}, each having an output coupled to the inverting input of a respective one of AND logic gates 275_0-275_{K-1}, and each having inputs coupled to receive a compare enable 246 (CE) signal from the request logic 265, the mask key 158 from the control interface 261, and a respective write data value from the data interface 263. When a write command other than a masked-write command is received within the request logic 265, the request logic 265 deasserts the compare enable signal 246 to disable the comparators 269 from asserting match signals 252, thereby enabling the write strobe signal 248 to pass through the logic AND gates 275_0-275_{K-1} to the respective enable inputs of the write drivers 273_1-273_{K-1}. When a masked-write

command is received within the request logic 265, the request logic 265 asserts the compare enable signal 246 to enable the comparators 269_0-269_{K-1} to assert match signals 252_0-252_{K-1}, respectively. Each of the comparators 269 compares the mask key 158 with a respective one of the write data values and, if the mask key and write data value match, asserts the corresponding match signal 252 to prevent the write strobe signal 248 from being passed to the enable input of the corresponding write driver 273. By this operation, each write data value that matches the mask key 158 is excluded from storage within the storage array 245.

[0027] Although the memory device 260 is depicted as including a multiple write drivers 273, multiple AND gates 275 and multiple comparators 269, the memory device 260 may alternatively have a single write driver 273, AND gate 275 and comparator 269, and a correspondingly reduced number of receiver circuits 271 within the data interface. Also, the storage cells within the storage array 245 may be implemented by any storage element capable of representing at least two data states. For example, in a dynamic random access memory (DRAM) device, each of the storage cells in storage array 245 is implemented by a capacitive device that may be in either a charged or discharged state to represent the two possible data states (multiple different charge levels may be used to achieve representation of more than two states, as where a single capacitive storage element is used to represent two or more bits of data). In a static random access memory (SRAM) device, a bi-stable latch circuit is used to implement each storage cell within storage array 245. Nonvolatile storage cells may also be used (e.g., floating gate devices, such as those used in flash erasable programmable read-only memory devices), as may non-semiconductor storage elements (e.g., magnetic or optical media).

[0028] Although read and write accesses within the memory device 260 have been described in terms of direct access to the storage array 245, such accesses may alternatively or additionally be directed to a page buffer within the memory device 260. For example, in the case of a DRAM device, a storage buffer (e.g., a bank of latches) may be used to temporarily store the contents of a row of the storage array 245 for subsequent read or write access directed to that row. In such a device,

masked-write operations may be directed to the storage buffer as well as the larger storage array 245.

[0029] Figure 6 illustrates the mask logic 207 of Figure 4 according to an embodiment of the invention. The mask logic 207 includes a key generator 291, mask buffer 295, write data buffer 297 and data coder 299. An incoming write data block and mask word (i.e., DATA and MASK received via the host interface) are stored in the write data buffer 297 and mask buffer 295, respectively, and are also supplied to the key generator 291 along with an incoming access request (REQ). The key generator 291 generates a mask key 158 having at least as many constituent bits as a write data value (i.e., Y/X bits, where Y is the number of bits in the write data block and X is the number of bits in the mask word; note that the mask key may have fewer constituent bits than the write data value, for example, in embodiments in which one or more of the Y/X bits of the write data value are omitted from comparison with the mask key), and outputs the mask key 158 to the data coder 299 and to the request generator (i.e., circuit block 209 of Figure 4). In one embodiment, the write data buffer 297 and mask buffer 295 are queue circuits (i.e., first-in-first-out storage circuits) that are loaded with the write data block and mask word, respectively, in one or more load operations, and then used to advance the component values of the write data block and mask word to the data coder 299 in a sequence of shift operations. For example, in one embodiment, a 64-byte write data block and corresponding 64-bit mask word are loaded into the write data buffer 297 and mask buffer 295, respectively, and thereafter output to the data coder 299 in sixteen successive transfers of four bytes and four corresponding mask bits. After each transfer of write data values and mask bits to the data coder 299, the contents of the write data buffer 297 and mask buffer 295 are advanced so that a new group of write data values and mask bits are present at the head of the write buffer and mask buffer and available for transfer to the data coder 299. The group of write data values supplied to the data coder 299 in each successive transfer operation is referred to herein as a chunk. In one embodiment, the chunk size, R (i.e., number of write data values per chunk), is a function of the width of the data path between the memory controller and storage subsystem (e.g., path 216 of

Figure 4) and is set to enable each write data value of a given chunk to be simultaneously output onto the data path.

[0030] Still referring to Figure 6, the data coder 299 includes a set of R selector circuits 301_0-301_{R-1} each having a first data input coupled to receive the mask key 158 from the key generator 291, a second data input coupled to receive a write data value from a respective one of the R leading storage locations within the write data buffer 297, and a select input coupled to receive a mask bit from a respective one of the R leading storage locations within the mask buffer 295. Referring to selector circuit 301_0, as an example, if the mask bit received at the select input is high (e.g., a logic 1), the corresponding write data value is indicated to be a masked data value and the selector circuit 301_0 selects the mask key 158 to be output to the data port (i.e., element 215 of Figure 4) instead of the write data value. If the mask bit is low, the write data value is an unmasked data value and is output to the data port by the selector circuit 301_0. Thus, the data coder 299 generates the coded data block by substituting the mask key 158 for each masked data value within the incoming write data block.

[0031] Figure 7 illustrates an exemplary three-stage pipeline for propagation of write data blocks through the memory controller of Figure 4. Write data blocks, WDB0-WDBn, are received in successive intervals of a first stage of the pipeline, PS1. The write data blocks are then buffered and used to generate a mask key in successive intervals of a second stage of the pipeline, PS2. Finally, the third stage of the pipeline, the write data blocks are converted to coded data blocks (CDB) and transmitted to the storage subsystem. In the embodiment shown in Figure 7, operations within each of the three stages of the pipeline are executed concurrently on incrementally different write data values. For example, after write data block WDB0 is received in interval 0 (INT 0), write data block WDB1 is received in interval 1 concurrently with generation of mask key 0 based on write data block WDB0. Similarly, write data block WDB2 is received in interval 2 concurrently with generation mask key 1 (based on WDB1) and concurrently with transmission of coded data block 0 (i.e., write data block 0 after mask key substitution has been

performed in the data coder). In alternative embodiments, each stage of the pipeline may be decomposed into additional pipeline stages.

[0032] Figure 8 illustrates the operation of the key generator 291 of Figure 6 according to an embodiment of the invention. The operations of Figure 8 are described in terms of byte masking, but may also be applied in systems having other mask granularities. As discussed above, the key generator operates to generate a mask key that does not match any unmasked bytes within an incoming write data block. In the embodiment of Figure 8, the key generator achieves this result by iteratively modifying the mask key, one bit per iteration, until the number of unmasked bytes of the write data block that potentially match the mask key is reduced to zero. The group of unmasked bytes that potentially match the mask key is referred to herein as the match pool and is initially established at 311 by removing all masked bytes from the write data block. That is, assuming a write data block having X bytes and Q masked bytes (Q being at least one), and assuming an arbitrary initial value for the mask key, the mask key may potentially match all the unmasked bytes, X-Q, within an incoming write data block. At 313, bit position index i, which represents a bit position within the mask key, is initialized to zero, and the match pool size, j, which represents the number of bytes within the match pool, is initialized to X-Q. At 315, the ith bit of all the bytes within the match pool, B_0 - B_{j-1}, are summed to produce a tally, T. Thus, in a first iteration of operation 315, the tally represents the number of 1's in the first bit position of the match-pool bytes. At 317, the tally is compared with j/2 to determine if more bytes within the match pool have a 1 at the ith bit position or a 0 at the ith bit position. If the tally is greater than j/2, then most of the bytes in the match pool have a 1 at the ith bit position, and the ith bit of the mask key, MK[i] is set to 0 in at 319A. At 321A, all the bytes having a 1 at the ith bit position are eliminated from the match pool as those bytes are now certain not to match the mask key. At 323A, the match pool size, j, is reduced by the value of the tally, T, to reflect the number of bytes remaining in the match pool after the elimination operation at 321A. Returning to decision operation 317, if T is less than or equal to j/2, then at least half the bytes in the match pool have a 0 at the ith bit position. In that case, the ith bit of the mask

key is set to 1 at 319B, and the bytes having a 0 in the ith bit position are eliminated from the match pool at 321B as those bytes are now certain not to match the mask key. At 323B, the match pool size, j, is set to the value of the 1's tally, T, as that number represents the number of bytes remaining in the match pool after the elimination operation at 321B. At 325, the bit position index, i, is incremented in preparation for a tally at the next bit position. At 327, the bit position index is compared with a maximum value to determine whether the mask key generation is complete. In general, for a write data block having X write data values, at least one of which is a masked value, the maximum number of unique bit patterns in the initial match pool is X-1. Accordingly, if at least half of the bytes within the match pool are eliminated in each iteration of operations 315-323, then the match pool is guaranteed to be reduced to zero (i.e., mask key doesn't match any of the unmasked write data values within the write data block) after $\log_2(X)$ iterations of operations 319-323. For example, if the write data block includes 64 bytes, one of which is masked, then after $\log_2(64)=6$ iterations of operations 315-323, the mask key will not match any of the unmasked data bytes. Accordingly, if the bit position index i is determined to have reached $\log_2(X)$ at 327, the mask key generation operation is complete. If the bit position index has not reached $\log_2(X)$, then another iteration of operations 315-323 is performed starting with a tally of 1s at the updated bit position index at 315.

[0033] Because a different mask key bit is set or reset in each iteration of operations 315-323, only six bits of the mask key are significant from the standpoint of making the mask key unique relative to the unmasked write data values. The remaining bits of the mask key (e.g., remaining two bits in a byte-masking embodiment) may be padded with zeroes or ones. Also, the six bits that are used to establish mask key uniqueness may occur at any offset within the total number of bits allocated to the mask key. For example, in the byte-masking embodiment of Figure 8, the bit position index could be incremented from 2 to 7 instead of 0 to 5.

[0034] The removal of masked data values from the write data block at 311 and the subsequent X-Q calculation in operation 313 may be omitted in an alternative

embodiment key generation embodiment that performs one extra iteration of operations 315-323. For example, in the case of a 64-byte write data block, if at least seven bits of the mask key are assigned through respective iterations of operations 315-323, then the resulting mask key will not match any of the constituent write data bytes, even if all 64 bytes are unique. That is, because at least half of the bytes of the write data block are eliminated from the match pool in each iteration of operations 315-323, after seven iterations, a pattern guaranteed to be unique among the 128 patterns of a 7-bit value is obtained, meaning that the mask key is certain to be unique relative to the 64 bytes of the write data block.

[0035] Still referring to Figure 8, in another alternative key generation embodiment, the match pool size, j, is initialized to X at 313-ALT (shown in dashed outline in Figure 8), and the match pool size is halved in operation 323-ALT rather than being adjusted at 323A and 323B according to the tally determined at 315 (i.e., operation 323-ALT replaces operation 323A and operation 323B). By this approach, j represents the guaranteed maximum number of write data values in the match pool after any iteration of operations 315-321 rather than the actual size of the match pool. This alternative key generation embodiment is particularly well suited to implementation in combinatorial logic because the divide by 2 operation of 323-ALT can be achieved by right-shifting of the value j by one bit, and the arithmetic operations in 313 and 323A are avoided.

[0036] Figure 9 illustrates an embodiment of a key generator 350 that operates according to the alternative arrangement illustrated by operations 313-ALT and 323-ALT in Figure 8. For purposes of illustration only, the key generator 350 is assumed to operate in a byte-masking memory controller that receives 64-byte write data blocks. Other mask granularities and block sizes may be used in alternative embodiments. Initially, the bytes of an incoming write data block are logically ANDed with the complements of their respective mask bits to generate a pool of mask-qualified bytes, $BYTE_{00}$& M_{00}, $BYTE_{01}$& M_{01}, ... , $BYTE_{63}$& M_{63}. By this operation, each masked data value in the mask-qualified byte pool is forced to zero; an operation that corresponds to the masked byte removal operation 311 of Figure 8. A summing circuit 351 is used to generate a 1's tally (i.e., sum of 1s) for

bit 0 of each byte within the mask-qualified byte pool. Because at least one byte of the write data block is masked, the 1's tally generated by summing circuit 351 may range from 0 to 63 and therefore yields a 6-bit value, T0[5:0], in which the most significant bit, T0[5], if set, indicates that more than half the bytes in the mask-qualified byte pool have a 1 in the 0th bit position. Accordingly, bit zero of the mask key, MK[0], is set to the complement of T0[5] by the operation of inverter 353.

[0037] The circuitry for generating bit 1 of the mask key, MK[1], includes two summing circuits 361_0 and 361_1, and an elimination circuit 360. The elimination circuit includes 64 bitwise elimination circuits, E0, each of which corresponds to a respective mask-qualified byte and generates two signals that correspond to the two possible elimination results according to the state of the 0th bit of the mask key, MK[0]. For example, if MK[0] is 0, then all bytes of the mask-qualified byte pool for which bit 0 (b0) is 1 are to be eliminated, and if MK[0]=1, then all bytes for which b0=0 are to be eliminated. Accordingly, each of the bitwise elimination circuits, E0, generates a first qualified bit 1 ($qb1_0$) which is forced to 0 if b0=1, and set according to b1 if b0=0; and a second qualified bit 1 ($qb1_1$) which is forced to 0 if b0=0, and equal to b1 if b0=1. In Boolean notation ('&' indicating a bitwise AND operation):

$qb1_0 =$ b1 & /b0; and

$qb1_1 =$ b1 * b0.

[0038] Summing circuit 361_0 sums the $qb1_0$ values for each byte in the mask-qualified byte pool to generate a 1's tally, $T1_0$[4:0], that corresponds to the elimination result when MK[0] is 0, and summing circuit 361_1 sums the $qb1_1$ values for each byte in the mask-qualified byte pool to generate 1's tally, $T1_1$[4:0], that corresponds to the elimination result when MK[0] is 1. Because of the eliminations performed in elimination circuit 360, the 1's tally that corresponds to the actual state of MK[0] ranges from 0 to 31 and therefore yields a 5 bit value in which the most significant bit is set if more than half the 31 possible non-eliminated bytes in the mask-qualified byte pool have a 1 in the bit 1 position. Accordingly, the most significant bits (MSBs) of the two tally values, $T1_0$[4] and $T1_1$[4], are input to a

multiplexer 362 which selects one of the tally MSBs according to MK[0]. The selected tally MSB is then inverted by inverter 363 to form bit 1 of the mask key value, MK[1]. Thus, MK[1] is set to 0 if b1=1 for more than half the possible number of non-eliminated bytes in the mask-qualified byte pool, and MK[1] is set to 1 otherwise.

[0039] The circuitry for generating bit 2 of the mask key, MK[2] depends on twice as many previously determined mask key bits as the circuitry for generating bit 1 of the mask key and therefore includes four summing circuits 371_0-371_3 instead of two, and an elimination circuit 370 that generates four, rather than two, qualified bit values per mask-qualified byte. More specifically, the elimination circuit 370 includes 64 bitwise elimination circuits, E1, each of which generates four qualified bit values, $qb2_0$-$qb2_3$, that correspond to the four possible elimination results according to the four possible states of MK[1:0]. For example, if MK[1:0]=00, then all bytes of the mask-qualified byte pool for which b1=1 and b0=1 are to be eliminated; if MK[1:0]=01, then all bytes for which b1=0 and b0=1 are to be eliminated, and so forth. Accordingly, each of the bitwise elimination circuits, D1, generates four qualified bit values, $qb2_0$-$qb2_3$, according to the following expressions:

$qb2_0$ = b2 & /b1 & /b0;

$qb2_1$ = b2 & /b1 & b0;

$qb2_2$ = b2 & b1 & /b0; and

$qb2_3$ = b2 & b1 & b0.

[0040] Summing circuit 371_0 sums the $qb2_0$ values for each byte in the mask-qualified byte pool to generate a 1's tally, $T2_0$[3:0] that corresponds to the elimination result when MK[1:0]=00, and summing circuits 371_1-371_3 similarly generate three separate 1's tallies, $T2_1$[3:0]-$T2_3$[3:0], that correspond to the elimination result when MK[1:0] is 01, 10, and 11, respectively. Because of the eliminations performed in elimination circuit 370, the 1's tally that corresponds to the actual state of MK[1:0] ranges from 0 to 15 and therefore yields a 4 bit value in which the most significant bit is set if more than half the 15 possible non-eliminated bytes in the mask-qualified byte pool have a 1 in the bit 2 position. The

most significant bits (MSBs) of the four tally values, $T2_0[3]$-$T2_3[3]$, are input to a multiplexer 372 which selects one of the four tally MSBs according to the MK[1:0] value. The selected tally MSB is then inverted by inverter 373 to form bit 2 of the mask key value, MK[2]. Thus, MK[2] is set to 0 if b2=1 for more than half the possible number of non-eliminated bytes in the mask-qualified byte pool, and MK[2] is set to 1 otherwise.

[0041] In the embodiment of Figure 9, the circuitry for generating mask key bits MK[3], MK[4] and MK[5] is similar to the circuitry for generating MK[2], except that, at each stage, the number of summing circuits and the number of qualified bit values generated by the elimination circuit is doubled. Thus, the circuitry for generating MK[5] includes 32 summing circuits, 391_0-391_{31}, and the elimination circuit 390 includes 64 bitwise elimination circuits, E5, each of which generates 32 qualified bit values, $qb5_0$-$qb5_{31}$, that correspond to the 32 possible elimination results according to the 32 possible states of MK[4:0]. The summing circuits 391_0-391_{31} generate 32 separate 1's tallies, $T5_0[0]$ - $T5_{31}[0]$, that correspond to the 32 possible elimination results (i.e., according to MK[4:0]). Because of the eliminations performed in elimination circuit 390, the 1's tally that corresponds to the actual state of MK[4:0] ranges from 0 to 1 and therefore yields a single-bit tally which is set to 1 if the single possible remaining byte in the mask-qualified byte pool has a 1 in the bit 5 position. The 32 tally bits, $T5_0[0]$ - $T2_{32}[0]$, are input to a multiplexer 392 which selects one of the 32 tally bits according to the MK[4:0] value. The selected tally bit is then inverted by inverter 393 to form bit 5 of the mask key value, MK[5]. Thus, MK[5] is set to 0 if b5=1 for the single possible remaining byte in the mask-qualified byte pool, and MK[5] is set to 1 otherwise. As discussed above, the mask key may be padded with 1s or 0s in any remaining bit positions (e.g., bit positions MK[6:7] in a byte-masking embodiment). Alternatively, the remaining bit positions may be left at arbitrary values.

[0042] Numerous changes may be made to the key generator of Figure 9 in alternative embodiments of the invention. For example, the qualified bit values generated by each stage of elimination circuits (360, 370,...,390) may be provided to the next stage elimination circuit to reduce logic redundancy. For example, the

qualified bit values generated by circuits D0 may be provided to circuits D1. Also, the logic AND operation used to generate the initial match-qualified byte pool may be omitted if circuitry for generating an additional mask key bit is provided (e.g., circuitry for generating MK[6]). Moreover, the mask key may be generated in multiple bit-generating iterations rather than in a single iteration. Generally, for an N-bit mask key, the mask key generator may generate some number of mask key bits, K, per iteration, where K is less than or equal to N, feeding all computed key bits into each subsequent iteration (i.e., if more than one iteration is required). For example, if N=6 and K=1, the mask key generator will generate one mask key bit per iteration using a single summing circuit (e.g., summing circuit 351 of Figure 9), thereby completing the mask key in six iterations. If N=6 and K=3, the mask key generator will generate three mask key bits per iteration using seven summing circuits (i.e. 1+2+4), thereby completing the mask key in two iterations. Figure 9 illustrates an embodiment in which N=6 and K=6, using 63 summing circuits and a single iteration to generate all the mask key bits. Generally, a tradeoff exists between the amount of key-generation circuitry and the latency of the mask key generation. Accordingly, the mask key generator can be customized for virtually any values of N and K according to the goals and constraints of a particular application. While a combinatorial logic circuit for generating a mask key is shown in Figure 9, a general purpose processor, special purpose processor, dedicated state machine or other state-based circuit may alternatively be used to generate the mask key. More generally, any circuit for generating a mask key that does not match unmasked data values within a write data block may be used without departing from the spirit and scope of the present invention.

Serialized Transfer of Write Data Values

[0043] Figure 10 illustrates an exemplary timing diagram for the transfer of byte-sized write data values over a 32-line data path between the memory controller and storage subsystem of Figure 4. As shown, four bytes are transferred in parallel over the data path during each of sixteen transmit intervals, thereby achieving transfer of a complete 64-byte write data block over sixteen transmit intervals. As discussed in reference to Figures 4 and 5, mask keys are compared to each write data value

received within a constituent memory device of the storage subsystem. Thus, if the parallel data transfer scheme of Figure 10 is used, each memory device generally requires a data interface at least as wide as the size of a write data value (i.e., at least as wide as the mask granularity). Accordingly, as shown in Figure 11, the maximum storage capacity that can be achieved in a single-rank of memory devices 401 according to the constraints of Figure 10 (i.e., 32-line data path and byte-mask granularity) is 4 x SC bits, where SC is the maximum storage capacity of a given generation of memory devices 401. More generally, the maximum storage capacity of a single rank of memory devices is SC x (DPW)/(MG), where DPW is the width of the data path between memory controller and storage subsystem, and MG is the mask granularity. Additional ranks of memory devices may be added to increase the capacity of the storage subsystem, but the added signal drops generally result in degraded signaling performance on the data path.

[0044] Figure 12 illustrates a data transfer scheme in an alternative embodiment of the invention that enables the per-rank storage capacity of the storage subsystem to be substantially increased. As shown, rather than transfer write data values over the data path in parallel, each write data value is transferred to the storage subsystem serially via a single data line. For example, during transmit interval 0, bit 0 of each of 32 bytes (B0-B31) is transferred to the storage subsystem via a respective line of the data path. Referring specifically to data line DQ00, bits 0-7 of byte B0 are transferred serially over data line DQ00 during respective transmit intervals 0 through 7. Bytes B1-B31 are similarly transferred serially over data lines DQ01-DQ31, respectively, during transmit intervals 0-7. During transmit intervals 8-15, the remaining bytes of the write data block, B32-B63, are transferred serially over data lines DQ00-DQ31, respectively. By serially transferring write data values in this manner, the data interface of each memory device within the storage subsystem may be made as narrow as a single bit and coupled to a corresponding one of the 32 data lines (more or fewer data lines may be used in an alternative embodiment). As shown in Figure 13, such an arrangement enables 32 SC-sized memory devices 403 to be used per rank of the storage subsystem, effectively multiplying the per-rank storage capacity of the

storage subsystem by the size of a write data value. For example, in a byte-masking memory system having a 32-line data path, the per-rank storage capacity of the storage subsystem is increased by a factor of 8, from the 4 x SC capacity of Figure 11 to the 32 x SC capacity of Figure 13. The increase in per-rank storage capacity becomes even more pronounced at larger mask granularities.

[0045] Figure 14 illustrates the data port 421 of a serial-transfer memory controller 420 according to an embodiment of the invention. The data port 421 includes a set of K output drivers, 425_0-425_{K-1}, and a corresponding set of K serializing circuits 423_0-423_{K-1}. A respective N-bit write data value is loaded into each of the serializing circuits 423 after every Nth cycle of a transmit clock signal 422, TCLK, and then shifted out of the serializing circuit 423, one bit at a time, during each of the following N cycles of TCLK. By this arrangement, after a write data value is shifted out of a given serializing circuit 423, the serializing circuit is reloaded with a new write data value, if available. Each of the output drivers is coupled to the output 426 of a respective serializing circuit to receive the serial stream of bits that form a write data value. In the embodiment of Figure 14, each output driver 425_0-425_{K-1} transmits a constituent bit of a write data value on a corresponding one of data lines DQ_0-DQ_{K-1} during each cycle of the transmit clock signal 422. Alternatively, the output drivers 425 may transmit in response to a clock signal that is phase offset from the clock signal used to control the shift operations within the serializing circuits 423. Also, in one embodiment, the serializing circuits 423 and output drivers 425 respond to both edges of the transmit clock signal 422, the N bits of a write data value being shifted out of a given serializing circuit after N/2 clock cycles of the transmit clock signal 422, and two bits of the write data value being transmitted one after the other on the corresponding data line per cycle of the transmit clock signal 422. In yet another embodiment, the output drivers 425 are multi-level output drivers capable of conveying two or more bits in a single signal transmission (e.g., transmitting one of four possible signal levels to convey a two-bit symbol per transmission). In such an embodiment, the clock signal used to clock the output drivers 425 may be operated at a lower frequency than the clock signal used to control the shift

operations within the serializing circuits 423 (e.g., half the frequency in the case of a two-bit symbol transmission). Alternatively, two bits may be output from each serializing circuit 423 in each shift operation.

[0046] Figure 15 illustrates an exemplary embodiment of a serializing circuit 430 that may be used within the data port 421 of Figure 14. The serializing circuit 430 includes a shift register 433 and a modulo N counter 431 each coupled to receive a transmit clock signal 422 (TCLK). The modulo N counter 431 is incremented in response to each rising and/or falling edge of TCLK and repeatedly counts from 0 to a terminal count of N-1, asserting a load signal 432 to the shift register 433 when the terminal count is reached. The shift register responds to assertion of the load signal 432 by loading N bits of a write data value 434 into a corresponding set of N storage elements within the shift register. The 0th bit of the write data value is stored at a head storage element 435 of the shift register 433 and output onto signal line 426 (i.e., to be received by an output driver). After the write data value 434 is loaded into the shift register 433, the constituent bits of the write data value are shifted forward one bit at a time (i.e., shifted toward the head storage element 435) in response to rising and/or falling edges of TCLK. By this arrangement, each bit of the write data value is shifted in turn to the head storage element 435 of the shift register 433 and output onto signal line 426. In an alternative embodiment, the write data value 434 may be shifted out most significant bit first. Also, two or more shift registers 433 may be provided to supply more than one bit to an output driver at a time, as in the case of a multi-level output driver.

[0047] Figure 16 illustrates a serial-transfer memory device 440 according to an embodiment of the invention. The memory device 440 includes a control interface 261, data interface 441, storage array 245, request logic 265, comparator 269, logic AND gate 275 and write driver 273. Though not specifically shown, the write driver 273 and logic gate 275 form part of a read/write circuit that includes a bank of sense amplifiers coupled to the storage array via bit lines 272. The control interface 261, storage array 245, request logic 265, address decoder 267, comparator 269, write driver 273 and logic AND gate 275 operate generally as described above in reference to Figure 5, but the data interface 441 is modified to

enable receipt of a write data value via a single data line, DQ. More specifically, the data interface 441 includes a receiver circuit 271 and deserializing circuit 443 each coupled to receive a receive clock signal 442 (RCLK). The receive circuit samples the signal line 271 in response to each rising and/or falling edge of the receive clock signal 442 and outputs the sampled bits one after another to the deserializing circuit 443. In one embodiment, the deserializing circuit 443 is a shift circuit that is loaded bit by bit in response to respective transitions of the receive clock signal 442. That is, as the receive circuit 271 outputs each sampled bit, the bit is shifted into the deserializing circuit 443 to reconstruct the write data value that was serialized by the memory controller. In a multi-level signaling embodiment (i.e., each symbol received by the receive circuit 271 represents more than one bit of information) the receive circuit 271 may output two or more bits per received symbol, with the two or more bits being strobed into the deserializing circuit 443 in response to each transition of the receive clock signal 442. After every N transitions of the receive clock signal 442 (i.e., each transition occurring on a rising edge, falling edge, or both) a complete write data value has been shifted into the deserializing circuit 443 and, in the case of a masked-write operation, the request logic 265 asserts the compare enable signal 246 to enable the comparator 269 to compare the write data value with the mask key 242 received via the control interface 261. If the mask key and write data value match, then the write data value is a masked data value and the comparator 269 asserts a mask signal 252 to prevent storage of the write data value. That is, as described in reference to Figure 5, the asserted mask signal 252 prevents a write strobe signal 248 generated by the request logic from being asserted the enable input of the write driver 273. If the write data value from the deserializing circuit 443 does not match the mask key 242, the comparator 269 does not assert the mask signal 252, and the write strobe signal 248 is passed via logic AND gate 275 to the enable input of the write driver 273, thereby enabling the write driver 273 to store the write data value within the storage array 245.

[0048] Still referring to Figure 16, the receive circuit 271, deserializing circuit 443, comparator 269, logic gate 275 and write driver 273 form a write data channel

within the memory device 440. While only one such write data channel is depicted in the embodiment of Figure 16, multiple write data channels may alternatively be provided, with the receive circuit 271 for each write data channel being coupled to a respective external data line, DQ.

Key Selecting Embodiment

[0049] In the exemplary memory systems described in reference to Figures 4-9 and 14-16, a memory controller generates a mask key for each incoming write data block, then transfers the mask key to a storage subsystem. In the case of a 64-byte write data block having byte-mask granularity, transfer of an 8-bit mask key instead of 64 discrete mask signals (i.e., the discrete mask signals described in reference to Figure 1) achieves an 8:1 reduction in the amount of mask information transferred between memory controller and storage subsystem. In terms of the overall interface between memory controller and storage subsystem, a bandwidth savings of 56 bits per write data block is achieved.

[0050] In an alternative embodiment of the invention, referred to herein as a key selecting embodiment, even greater bandwidth savings and greater reduction of mask information transfer are achieved. In the key selecting embodiment, the memory controller selects a mask key from a table of predetermined mask keys, then supplies a value representative of the selected mask key (referred to herein as a key selector) to the storage subsystem. The memory controller uses the selected mask key to generate a coded data block then outputs the coded data block to the storage subsystem in either the parallel-transfer or serial-transfer modes described in reference to Figures 10 and 12. Each of the memory devices (or memory device) that form the storage subsystem includes the table of predetermined mask keys and selects one of the predetermined keys according to the key selector received from the memory controller. Each memory device then uses the selected mask key in the manner described above in reference to Figures 5 and 16 to identify masked data values and exclude the masked data values from storage in the storage subsystem.

[0051] Because the key selector may have significantly fewer bits than a mask key value (the mask key being sized according to the mask granularity), transfer of the key selector to the storage subsystem instead of a mask key may achieve

additional bandwidth savings in the path between memory controller and storage subsystem and achieve further reduction in mask information transfer. For example, in one embodiment, the key selector is a single-bit value used to select between two predetermined mask keys. Transferring the key selector instead of, for example, an 8-bit mask key achieves an 8:1 reduction in mask key information transferred from the memory controller to storage subsystem. Compared to the 64 mask bits transferred from memory controller to storage subsystem in the prior art system described in reference to Figure 1, transferring a single-bit key selector value achieves a 64:1 reduction in mask information transfer.

[0052] Figure 17 illustrates the operation of a key selecting memory controller according to an embodiment of the invention. Byte-masking granularity is assumed for purposes of description, but other mask granularities may be used in alternative embodiments. At 475, a write data block having X constituent bytes is received via the host interface. At 477, a first predetermined mask key, mask key A, is compared against each of the unmasked bytes within the write data block. If none of the unmasked bytes match mask key A (decision block 479), then at 481 mask key A is assigned to be the selected mask key (SMK), and the key selector (KSEL) is set to a value that corresponds to mask key A (KSEL=0 in this example). Returning to decision block 479, if any one of the unmasked bytes of the write data block is determined to match mask key A, then at 487 each of the unmasked bytes is compared with a second predetermined mask key, mask key B. If none of the unmasked bytes of the write data block match mask key B (decision block 489), then at 491 mask key B is assigned to be the selected mask key, and the key selector is set to a corresponding value (KSEL=1 in this example).

[0053] If at least one of the predetermined mask keys is determined not to match any of the unmasked bytes of the write data block, then at 483 a coded data block is generated by substituting the selected mask key for each masked byte within the write data block. At 485, the coded data block and the key selector are transmitted to the storage subsystem.

[0054] If neither of the predetermined mask keys is determined to be unique relative to the unmasked bytes of the write data block (i.e., each mask key matches

at least one unmasked byte resulting in affirmative determinations at 479 and 489), then a condition referred to herein as a mask conflict has arisen. In the embodiment of Figure 17, the mask conflict is resolved through a two-phase masked-write operation. In the two-phase masked-write operation, a first phase masked-write operation is executed using mask key A, and a second phase masked-write operation is executed using mask key B. That is, at 493, mask key A is assigned to be the selected mask key (and the key selector set to 0) and the operations 483 and 485 are executed (indicated by the circled 'A' at 495) to complete the first phase masked-write operation, then at 497, mask key B is assigned to be the selected mask key (and the key selector set to 1) and the operations 483 and 485 are executed a second time (indicated at 499) to complete the second phase of the two-phase masked-write operation. Because of the mask conflict, one or more unmasked bytes that match mask key A will inadvertently be masked during the first phase of the two-phase masked-write, but written during the second phase.

[0055] Figure 18 illustrates a two-phase masked-write operation applied to an exemplary write data block (WDB) and corresponding mask word (MSK). For purposes of example only, byte-mask granularity is assumed, and mask keys A and B are assumed to be hexadecimal values 55 (binary value 01010101) and AA (binary value 10101010), respectively. A mask conflict exists in the scenario shown because the write data block contains unmasked values that match both mask key A and mask key B (i.e., unmasked values of 55 and AA, respectively). Accordingly, mask key A is assigned to be the selected mask key for purposes of generating a coded data block, CDB-A, to be written in a first phase of a two-phase masked-write operation. As shown by the shaded '55' entries in coded data block CDB-A, mask key A is substituted for masked bytes within the write data block (i.e., 12 , 70 and FF) to generate the coded data block, CDB-A. Also, as shown by the bold box 511 in coded data block CDB-A, the unmasked 55 value in the write data block, by happenstance, matches mask key A and therefore will be treated like a masked data value within the storage subsystem. Figure 19 illustrates the content of the storage subsystem before and after each phase of the two-phase masked-write operation. For purposes of example, the storage area to which the masked-

write operation is directed is assumed to contain zero-valued entries. Accordingly, after the first phase of the two-phase masked-write operation, all the storage locations are updated with write data bytes except for those locations for which the corresponding write data byte matches mask key A. Consequently, the storage location 515 which corresponds to the unmasked 55 value in the original write data block is not updated in the first phase of the two-phase masked-write operation, even though the intent of the host-requested write operation was to write the value 55 into storage location 515.

[0056] Referring again to Figure 18, in the second phase of the two-phase masked-write operation, mask key B is substituted for each masked byte within the write data block WDB as indicated by the shaded, AA-valued entries. As shown by the bold box 513, the unmasked AA value, by happenstance matches mask key B and therefore will be treated like a masked data value within the storage subsystem. Referring to Figure 19, in the second phase of the two-phase masked-write operation, all the storage locations are updated with write data bytes except for those locations for which the corresponding write data byte matches mask key B. Because the unmasked 55 value now does not match the mask key value, the unmasked 55 value is written in location 515 as originally intended. Even though the unmasked AA value inadvertently matches the mask key value and is not written in the second phase of the two-phase masked-write operation, the unmasked AA value was already written in the first phase write operation. Thus, the first and second phase write operations complement one another by achieving storage of the unmasked value that, by happenstance, matches the mask key value for the counter-part phase of the two-phase masked-write operation.

[0057] It should be noted that execution of a two-phase masked-write operation, though effective for resolving a mask conflict, has the undesirable characteristic of requiring two masked-write accesses to the storage subsystem instead of one. Consequently, the greater the frequency of two-phase masked-write operations, the lower the effective memory bandwidth of the memory system. One direct way to reduce the frequency of two-phase masked-write operations is to increase the number of predetermined mask keys from which the mask key is selected. As a

statistical matter, assuming a set of R predetermined mask keys and a population of X write data values each having a unique pattern of N constituent bits, each additional mask key decreases the likelihood of a mask conflict by a factor of (X-R)/(2^N-R). For example, in a system having a population of 64 write data values (one masked), byte-mask granularity, and two predetermined mask keys, the likelihood of a mask conflict in the population is (63/256)*(62/255) = ~6%. If two additional predetermined mask keys are provided, the likelihood of a mask conflict is reduced to (63/256)*(62/255)* (61/254)*(60/253) = ~0.34%. Letting P represent the number of predetermined mask keys, the number of constituent bits required in the key selector is $\log_2(P)$. In general, so long as $\log_2(P)$ is smaller than the mask granularity, a bandwidth savings is achieved over a mask-key-transfer embodiment.

[0058] Other techniques for reducing the likelihood of mask conflicts include choosing the set of predetermined mask keys according to known or anticipated bias within the write data population. For example, in systems which use hexadecimal values FF and 00 to represent true and false Boolean states, using patterns other than FF and 00 for each of the predetermined mask keys may reduce the likelihood of a mask conflict. Another approach to reducing mask conflicts is to subdivide the write data block into two or more sub-blocks, thereby reducing the population size, X, in the mask conflict probability expression ($X/2^N$)*((X-1)/2^N-1))*...*((X-R)/(2^N-R)). Assuming that X>>R, for example, subdividing the write data block into N sub blocks reduces the probability of a mask conflict by ~N.

[0059] Figure 20 illustrates a key-selecting mask logic circuit 550 according to an embodiment of the invention. The mask logic circuit 550 includes a key table 551, key select circuit 553, mask buffer 555, write data buffer 557 and data coder . The mask buffer 555, write data buffer 557 and data coder 299 operate generally as described above in reference to Figure 6 to generate a coded data block based on an incoming mask word (MASK) and write data block (DATA). Instead of receiving a mask key from a key generator, however, the data coder 299 receives a predetermined mask key selected from the mask table 551 by the key select circuit 553. That is, the key select circuit 553 selects one of the predetermined mask keys from table 551 to be a selected mask key 554 (SMK) and supplies the selected

mask key 554 to the data coder for generation of the coded data block. The key select circuit 553 also generates a key selector 552 that corresponds to the selected mask key 554, and outputs the key selector 552 to the request generator. Thus, the request generator receives and forwards the key selector 552 instead of a mask key.

[0060] In one embodiment, the key table 551 includes a bank of programmable storage circuits that are programmed with host-supplied mask keys during system initialization. The programmable storage circuits may be implemented using volatile or non-volatile storage elements and, in the case of non-volatile storage elements may alternatively be programmed during system production or in a configuration-setting operation. In an alternative embodiment, the key table 551 includes a hard-wired set of mask keys (e.g., achieved through dedicated circuit design or through one-time programmable operation such as a fuse-blowing programming operation). Also, while the key table 551 is depicted as including two predetermined mask keys (MKA and MKB) in Figure 20, the key table 551 may include more than two predetermined mask keys in alternative embodiments.

[0061] The key select circuit 553 is coupled to receive the mask word and write data block from the host interface, as well as the predetermined mask keys from the key table 551. In one embodiment, the key select circuit 553 is implemented by combinatorial logic that compares each of the predetermined mask keys in the key table 551 with the unmasked write data values of the write data block and, if at least one of the predetermined mask keys is unique relative to the unmasked write data values, generates a key selector that selects the unique predetermined mask key to be the selected mask key 554. If none of the predetermined mask keys is unique relative to the unmasked write data values (i.e., each mask key in the key table 551 matches at least one unmasked write data value), the key select circuit 553 selects a first default mask key from the key table 551 (e.g., mask key A) and asserts a conflict signal 556 to indicate the mask conflict condition. In the embodiment of Figure 20, the conflict signal 556 is supplied to the mask buffer and write data buffer to maintain the content of those buffers through first and second phases of a two-phase masked-write operation. That is, the conflict signal 556 is asserted to prevent the contents of the mask buffer 555 and write data buffer 557

from being overwritten by an incoming mask word and write data block. The conflict signal 556 may additionally, or alternatively, be supplied to other control circuitry within the memory controller to enable the control circuitry to prevent the mask buffer and write data buffer from being overwritten before completion of the two-phase masked-write operation. Such control circuitry may also output a busy signal to notify access requestors that the memory controller is temporarily unable to receive a write data block.

[0062] Still referring to the mask conflict condition, the first default mask key is supplied to the data coder 299 as the selected mask key 554 for the first phase of a two-phase masked-write operation. After the coded data block (generated using the first default mask key) and selected mask key 554 are output to the storage subsystem, a second default mask key (e.g., mask key B) is selected by the key select circuit and provided to the data coder as the selected mask key. The second default mask key is then used to generate the coded data block for the second phase of the two-phase masked-write operation.

[0063] Figure 21 illustrates an exemplary state diagram 565 implemented by the key select circuit 553 of Figure 20. Initially, the key select circuit 553 is in state 556, and outputs logic low key selector and conflict signals (i.e., /KSEL, /CNFL). The key select circuit remains in state 556 so long as mask key A (MKA) does not match any unmasked data values within an incoming write data block (i.e., MKA<>WDB). If mask key A matches one or more unmasked data values within an incoming write data block, and mask key B does not match any unmasked data values within the write data block (i.e., (MKA=WDB) AND (MKB<>WDB)), then the key select circuit transitions to state 557. In state 557, the conflict signal remains low, but the key selector goes high to select mask key B for use in generating the coded data block. When the next write data block is received, the key logic returns to state 556 if mask key A does not match any unmasked data values. If the mask key A matches an unmasked data value within the write data block, but mask key B does not, the key select circuit remains in state 557. Whether in state 556 or 557, if mask key A and mask key B are each determined to match an unmasked data value within an incoming write data block, then a mask

conflict has occurred and the key select circuit transitions to state 558. The key selector remains in state 558 for the duration of a two-phase masked-write operation, then transitions to either state 556 or 557 if no mask conflict is detected within the next write data block (i.e., transitioning to state 556 if mask key A matches an unmasked write data value within the write data block and to state 557 otherwise).

[0064] Figure 22 illustrates the key select circuit 553 of Figure 20 according to an embodiment of the invention. The key select circuit 553 includes a multiplexer 571 and a select signal generator 575, each coupled to receive the predetermined mask keys from the key table. Four predetermined mask keys, MK0-MK3, are assumed in the exemplary embodiment of Figure 22, though more or fewer mask keys may alternatively be used. The select signal generator 575 receives the write data block and mask word as inputs and generates the key selector 552 and conflict signal 556 described in reference to Figure 21. In the embodiment of Figure 22, the key selector 552 is a 2-bit signal, KSEL[1:0] and is supplied to a select input of the multiplexer 571 to select one of the four predetermined mask keys to be output as the selected mask key 554 (SMK).

[0065] The select signal generator 575 includes a set of X match detect circuits 579_0-579_{X-1} (MD), logic AND gates 585_0-585_3, and encoder 587. Each of the match detect circuits 579 is coupled to receive the four predetermined mask key values, MK0-MK3, from a key table, along with a respective one of the X write data values of the write data block and a corresponding mask bit of the mask word. Each of the match detect circuits 579 generates a respective set of qualified match signals (i.e., $QM0_{0-3}$, $QM1_{0-3}$,..., $QM(X-1)_{0-3}$) that correspond to the predetermined mask keys MK0-MK3, each qualified match signal indicating whether the corresponding mask key matches an unmasked write data value. Referring to the detail view of match detect circuit 579_0, as an example, each predetermined mask key is compared with write data value, WD0 (i.e., a write data value of the incoming write data block), in a respective one of comparators 581. If the write data value and predetermined mask key received by a given comparator 581 match, the comparator outputs a logic high match signal on a corresponding one of match

lines 582. The match lines 582 are coupled to inputs of respective AND gates 583, with each of the AND gates having an inverting input coupled in common to receive mask bit, M0. Accordingly, each of the AND gates 583 outputs a respective qualified match signal in a logic high state if a match is signaled by the corresponding comparator 581 and the write data value is unmasked (i.e., mask bit M0 is low). Expressed in analytic form, each of the X mask detection circuits 579_0-579_{X-1} (MD_0-MD_{X-1}) in the exemplary embodiment of Figure 22 outputs four qualified match signals as follows:

MD_0: $QM0_0$: (WD0 = MK0) & /M0
$QM0_1$: (WD0 = MK1) & /M0
$QM0_2$: (WD0 = MK2) & /M0
$QM0_3$: (WD0 = MK3) & /M0

MD_1: $QM1_0$: (WD1 = MK0) & /M1
$QM1_1$: (WD1 = MK1) & /M1
$QM1_2$: (WD1 = MK2) & /M1
$QM1_3$: (WD1 = MK3) & /M1

•
•
•

MD_{X-1}: $QM(X-1)_0$: (WD(X-1) = MK0) & /M(X-1)
$QM(X-1)_1$: (WD(X-1) = MK1) & /M(X-1)
$QM(X-1)_2$: (WD(X-1) = MK2) & /M(X-1)
$QM(X-1)_3$: (WD(X-1) = MK3) & /M(X-1)

[0066] Still referring to Figure 22, the qualified signals $QM0_0$, $QM1_0$, ..., $QM(X-1)_0$ all correspond to mask key MK0 (each indicating whether MK0 matches a respective unmasked data value within the write data block), and are supplied to respective inverting inputs of AND gate 585_0. Thus, if all the qualified match signals corresponding to MK0 are low, the output of AND gate 585_0 (i.e., S0) will

be high to indicate that MK0 does not match any unmasked data values within the write data block. Similarly, qualified match signals $QM0_1$, $QM1_1$, ..., $QM(X-1)_1$ all correspond to MK1 and are supplied to inverting inputs of AND gate 585_1; qualified match signals $QM0_2$, $QM1_2$, ..., $QM(X-1)_2$ all correspond to MK2 and are supplied to inverting inputs of AND gate 585_2; and qualified match signals $QM0_3$, $QM1_3$, ..., $QM(X-1)_3$ all correspond to MK3 and are supplied to inverting inputs of AND gate 585_3. Thus each of the AND gates 585_0-585_3 will output a logic high signal if the corresponding mask key, MK0-MK3, does not match any unmasked data values within the write data block. The outputs of the AND gates 585_0-585_3 (i.e., signals S0-S3, respectively) are supplied to the encoder 587 where they are used to set the states of the key selector 552 and conflict signal 556. In one embodiment, the encoder 587 generates a key selector 552 that corresponds to the lowest numbered match key for which the output of the corresponding one of signals S0-S3 is high. That is, KSEL[1:0] is set to 00 to select MK0 if S0 is high; 01 to select MK1 if S0 is low and S1 is high; 10 to select MK2 if S0 is low, S1 is low and S2 is high; and 11 to select MK3 if S0 is low, S1 is low, S2 is low and S3 is high. If signals S0-S3 are all low, then none of the mask keys MK0-MK3 are unique relative to the write data block and a conflict condition exists. In the embodiment of Figure 22, the encoder 587 asserts the conflict signal 556 to indicate the mask conflict condition, and sets the key selector 552 to select mask key MK0 to be the default mask key for a first phase of a two-phase masked-write. After the first phase of the two-phase masked-write, the encoder 557 sets the key selector to select mask key MK1 to be the default mask key for the second phase of the two-phase masked-write. Other key table selections may be used as the default mask keys for the first and/or second phases of the two-phase masked-write in alternative embodiments.

[0067] Figure 23 illustrates a memory device 600 according to a key-selecting embodiment of the invention. The memory device 600 includes a host interface 601, data interface 603, address decoder 267, request logic 265, key table 609, multiplexer 611, comparator 269, write driver 273 and logic gate 275. The data interface 603 (including receivers 271), address decoder 267, request logic 265,

comparator 269, write driver 273 and logic gate 275 all operate generally as described above in reference to Figure 5, although the data interface 603 may alternatively be a deserializing data interface as described in reference to Figure 16. The host interface 601 also operates as described in reference to Figure 5 except that a key selector 552 is received instead of a mask key. The key selector 552 is supplied to a select input of the multiplexer 611 to select a mask key from the key table 609, the selected mask key 610 being provided to the comparator 269 for comparison with an incoming write data value (i.e., when the comparator 269 is enabled by the compare enable 246 signal from the request logic 265). Note that while a single write data channel is shown in Figure 23, the memory device 600 may alternatively have any number of write data channels in alternative embodiments (i.e., wider data interface 603, multiple comparators 269, multiple write drivers 273 and logic gates 275 as in Figure 5).

[0068] In the embodiment of Figure 23, the key table 609 includes four predetermined mask keys, and the incoming key selector 552 is a 2-bit value, KSEL[1:0], to enable the one-of-four selection. More or fewer predetermined mask keys may be provided within the key table 609 in an alternative embodiment, and the number of bits of the key selector 552 correspondingly increased or decreased. In one embodiment, the key table 609 includes a bank of programmable storage circuits that are programmed with mask keys supplied by the memory controller during system initialization. The programmable storage circuits may be implemented using volatile or non-volatile storage elements and, in the case of non-volatile storage elements may be programmed during system production or in a configuration-setting operation. In an alternative embodiment, the key table 609 includes a hard-wired set of mask keys (e.g., achieved through dedicated circuit design or through one-time programmable operation such as a fuse-blowing programming operation).

Sub-Block Coding

[0069] As mentioned briefly above, one approach to reducing mask conflicts in a key-selecting memory controller is to select multiple mask keys for each incoming write data block, each selected mask key corresponding to a respective sub-block

(i.e., portion) of the write data block. Each selected mask key is then substituted for mask values within the corresponding sub-block to generate a coded sub-block, and the coded sub-blocks are transmitted to the storage subsystem along with the corresponding sub-block key selectors. The reduced population size of each sub-block (i.e., reduced number of write data values as compared to the complete write data block), produces an approximately proportional reduction in the probability of a mask conflict.

[0070] Figure 24 illustrates an exemplary spatial partitioning of a 64-byte write data block into two 32-byte sub-blocks, sub-block 0 and sub-block 1. As shown, the 32 data bytes transmitted on data lines DQ00 to DQ15 form sub-block 0, and the 32 data bytes transmitted on data lines DQ16-DQ31 form sub-block 2. Finer spatial partitioning may be used in alternative embodiments to further reduce the likelihood of mask conflicts (e.g., partitioning into 4 sub-blocks each transmitted on set of eight lines of the data path).

[0071] Figure 25 illustrates an exemplary temporal partitioning of a 64-byte write data block into two 32-byte sub-blocks, sub-block 0 and sub-block 1. As shown, the 32 data bytes transmitted during transmission intervals 0-7 form sub-block 0, and the 32-data bytes transmitted during transmission intervals 8-15 form sub-block 1. Finer temporal partitioning may be used in alternative embodiments (e.g., partitioning into 4 sub-blocks each transmitted one after another during respective sets of transmission intervals). Also, combinations of temporal and spatial partitioning may also be used. For example, the spatial partition depicted in Figure 24 may be combined with the temporal partition of Figure 25 to produce a four-quadrant partition. Further, while serialized data transfer schemes are illustrated in Figures 24 and 25, parallel data transfer schemes (e.g., as shown in Figure 12) may alternatively be used.

[0072] Regardless of the type and number of partitions, separate key selector values are generated based on the write data values within each of the sub-blocks, and used to select respective mask keys for coding the sub-blocks. The coded sub-blocks and corresponding key selectors are then output to the storage subsystem.

[0073] Figure 26 illustrates an exemplary encoding of a key selector that may be used within a key-selecting memory system of the present invention. As shown, a 2-bit key selector, KSEL[1:0], is used to indicate whether a given write operation is a masked-write operation or an unmasked write operation, and if a masked-write operation, to specify one of three predetermined mask keys. That is, if the key selector specifies any one of the three predetermined mask keys, a masked-write operation is indicated. Otherwise the key selector is set to 00 to indicate an unmasked write operation.

[0074] Section headings have been provided in this detailed description for convenience of reference only, and in no way define, limit, construe or describe the scope or extent of such sections. Also, while the invention has been described with reference to specific exemplary embodiments thereof, it will be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention. Accordingly, the specification and drawings are to be regarded in an illustrative rather than a restrictive sense.

CLAIMS

What is claimed is:

1. A memory controller comprising:
a host interface to receive a block of write data values and a plurality of mask values, each mask value indicating whether a corresponding one of the write data values is a masked data value to be excluded from storage within a storage subsystem;
a mask logic circuit coupled to receive the write data values and mask values from the host interface and adapted to generate a coded block of write data values by substituting a key value for each masked data value within the block of write data values; and
a memory interface coupled to receive the coded block of write data values from the mask logic circuit and adapted to output the coded block of write data values to the storage subsystem.

2. The memory controller of claim 1 wherein each of the mask values is a bit having either a first state or a second state, the first state indicating that the corresponding one of the write data values is a masked data value.

3. The memory controller of claim 1 wherein each of the write data values received by the host interface is a byte having eight constituent bits and wherein each of the mask values indicates whether the corresponding one of the bytes is a masked byte to be excluded from storage within the storage subsystem.

4. The memory controller of claim 1 wherein the mask logic circuit comprises a key generating circuit to generate the key value.

5. The memory controller of claim 4 wherein each of the write data values not indicated to be a masked data value constitutes an unmasked data value, and wherein the key generating circuit is adapted to generate the key value such that the key value does not match any unmasked data value within the block of write data values.

6. The memory controller of claim 1 wherein the memory interface is additionally coupled to receive the key value from the mask logic circuit and adapted to output the key value to the storage subsystem.

7. The memory controller of claim 6 wherein the memory interface comprises a data interface to output the coded block of write data values to the storage subsystem and a request interface to output the key value to the storage subsystem.

8. The memory controller of claim 7 wherein the data interface comprises:
a data transmission circuit to output, from the memory controller, signals representative of the coded block of write data values; and
a request transmission circuit to output from the memory controller, signals representative of the key value.

9. The memory controller of claim 8 wherein the data transmission circuit comprises a first plurality of output driver circuits each adapted to concurrently output a signal representative of a respective portion of the coded block of write data values.

10. The memory controller of claim 9 wherein each output driver circuit of the first plurality of output driver circuits is adapted to output a respective bit of a first write data value of the coded block of write data values such that signals representative of the first write data value are output in parallel from the memory controller.

11. The memory controller of claim 10 further comprising a second plurality of output driver circuits adapted to output a second write data value from the memory controller concurrently with the first write data value.

12. The memory controller of claim 8 wherein the data transmission circuit comprises a first output driver circuit adapted to serially output each of the constituent bits of a first write data value of the coded block of write data values.

13. The memory controller of claim 8 wherein the data transmission circuit comprises a plurality of output driver circuits each adapted to serially output constituent bits of a respective write data value of the coded block of write data values.

14. The memory controller of claim 1 wherein the mask logic circuit comprises a key selector circuit to select the key value from of a plurality of predetermined values.

15. The memory controller of claim 14 further comprising a plurality of storage circuits to store the plurality of predetermined values.

16. The memory controller of claim 15 further comprising control logic to program the plurality of predetermined values into the storage circuits.

17. The memory controller of claim 16 wherein the control logic is adapted to program the plurality of predetermined values into the storage circuits in response to a command received via the host interface.

18. The memory controller of claim 14 wherein the key selector circuit comprises a multiplexer having a plurality of data inputs to receive the plurality of predetermined values, and a control input to receive a key select value, the

multiplexer being adapted to output one of the plurality of predetermined values as the key value according to the key select value.

19. The memory controller of claim 18 wherein the key selector further comprises a select value generator to generate the select value.

20. The memory controller of claim 19 wherein each of the write data values not indicated to be a masked data value constitutes an unmasked data value, and wherein the select value generator is coupled to receive the block of write data values and the plurality of mask values from the host interface, the select value generator being adapted to identify one of the plurality of predetermined values that does not match any unmasked data value within the block of write data values, the select value generator being further adapted to generate the select signal according to the identified one of the plurality of predetermined values.

21. A method of operation within a memory controller, the method comprising:
receiving a block of write data values and a plurality of mask values, each mask value indicating whether a corresponding one of the write data values is a masked data value;
generating a coded block of write data values by substituting a key value for each masked data value; and
outputting the coded block of write data values to a storage subsystem.

22. The method of claim 21 further comprising outputting a request to the storage subsystem to store each write data value that does not match the key value.

23. The method of claim 21 further comprising generating the key value.

24. The method of claim 23 wherein each of the write data values not indicated to be a masked data value constitutes an unmasked data value, and wherein generating the key value comprises generating a key value that does not match any unmasked data value within the block of write data values.

25. The method of claim 23 further comprising outputting the key value to the storage subsystem.

26. The method of claim 21 further comprising selecting the key value from a table of predetermined key values.

27. The method of claim 26 further comprising outputting a selector value to the storage subsystem, the selector value corresponding to the key value selected from the table of predetermined keys.

28. The method of claim 27 further comprising the predetermined keys into the table of predetermined keys in an initialization operation.

29. The method of claim 21 wherein outputting the coded data block of write data values to the storage subsystem comprises outputting at least one write data value of the coded data block to the storage subsystem as a serial stream of bits.

30. The method of claim 26 wherein outputting the at least one write data value as a serial stream of bits comprises outputting each constituent bit of the write data value in successive transmissions over a first signal line.

31. A memory device comprising:
a storage array;
a compare circuit to compare a write data value with a key value and to generate a match signal that indicates whether the write data value

matches the key value; and

a write circuit coupled to the storage array and the compare circuit, the write circuit being adapted to store the write data value in the storage array if the match signal indicates that the write data value does not match the key value.

32. The memory device of claim 31 further comprising a receive circuit to receive the write data value and to output the write data value to the compare circuit and the write circuit.

33. The memory device of claim 32 wherein the receive circuit is adapted to receive the key value and to output the key value to the compare circuit.

34. The memory device of claim 33 wherein the receive circuit comprises a plurality of sampling circuits, including at least one sampling circuit to receive the write data value and at least one other sampling circuit to receive the key value.

35. The memory device of claim 31 wherein the storage array comprises an array of capacitive elements, each adapted to be in either a charged state or a discharged state according to a binary datum to be stored therein.

36. The memory device of claim 31 wherein the storage array comprises an array of latch circuits, each adapted to latch a first state or a second state according to a binary datum to be stored therein.

37. The memory device of claim 32 further comprising a table to store a plurality of predetermined values, and a select circuit to select one of the predetermined values to be the key value.

38. The memory device of claim 37 wherein the receive circuit is adapted to receive a selector value, and wherein select circuit is adapted to select the one of the predetermined values according to the selector value.

39. The memory device of claim 32 wherein the receive circuit comprises at least one sampling circuit to receive the write data value in a sequence of sampling operations.

40. A method of operation within a memory device, the method comprising:
receiving a write data value to be stored within a storage array of the memory device;
comparing the write data value with a key value; and
storing the write data value within the storage array if the write data value does not match the key value.

41. The method of claim 40 further comprising receiving the key value.

42. The method of claim 40 further comprising selecting one of a plurality of predetermined values to be the key value.

43. The method of claim 42 further comprising receiving the plurality of predetermined values and storing the plurality of predetermined values within the memory device.

44. The method of claim 42 further comprising receiving a select value and wherein selecting one of the plurality of predetermined values comprises selecting one of the plurality of predetermined values indicated by the select value.

45. A memory system comprising:
a memory controller to output a write data value; and

a memory device coupled to receive the write data value from the memory controller, the memory device being adapted to compare the write data value with a first key value and to store the write data value if the write data value does not match the first key value.

46. The memory system of claim 45 wherein the memory controller is adapted to generate the first key value and to output the first key value to the memory device.

47. The memory system of claim 45 wherein the memory controller comprises a first table of predetermined values, and wherein the memory controller is adapted to select a second key value from the first table of predetermined values.

48. The memory system of claim 47 wherein the memory controller is further adapted to output a selector value to the memory device, the selector value corresponding to one of the predetermined values within the first table of predetermined values.

49. The memory system of claim 48 wherein the memory device comprises a second table of predetermined values, and wherein the memory device is adapted to receive the selector value and to select a is adapted select the first key value from the second table of predetermined values based on the selector value.

50. A method of operation within a memory system, the method comprising:

receiving a block of write data values and a corresponding plurality of mask values;

generating a coded block of write data values by substituting a key value for each of the write data values for which a corresponding one of the mask values is in a first state;

transferring the coded block of write data values to a storage subsystem;

comparing, within the storage subsystem, each of the write data values with the key value; and

storing, within the storage subsystem, each of the write data values determined not to match the key value.

51. The method of claim 50 further comprising generating the key value and transferring the key value to the storage subsystem.

52. The method of claim 51 wherein generating the key value comprises generating a key value that does not match any of the write data values for which a corresponding one of the mask values indicates a mask condition.

53. The method of claim 50 further comprising:

selecting the key value from a first table of predetermined values using a selector value;

transferring the selector value to the storage subsystem; and

selecting the key value from a second table of predetermined values within the storage subsystem.

54. The method of claim 53 further comprising storing predetermined values into the second table of predetermined values in an initialization operation.

55. A memory controller comprising:

a request interface to receive a first write data value and a request to store the first write data value in a storage subsystem; and

a serializing driver circuit coupled to receive the first write data value from the request interface as a set of parallel constituent bits and adapted to output the first write data value to the storage subsystem via a first signal line as a serial stream of constituent bits.

56. The memory controller of claim 55 wherein the serializing driver circuit comprises:

a serializing circuit to convert the set of parallel constituent bits to the serial stream of constituent bits; and

an output driver to receive each of bit of the serial stream of constituent bits in succession from the serializing circuit, and to output each bit of the serial stream of constituent bits onto the first signal line.

57. The memory controller of claim 56 wherein memory controller further comprises an interface to couple to an external signal line, and wherein the first signal line is coupled between the output driver and the interface.

58. The memory controller of claim 56 wherein the serializing circuit comprises a shift register having a plurality of storage elements, the shift register being adapted to receive the set of parallel constituent bits within the plurality of response to a load signal transition and to output each bit of the set of parallel constituent bits to the output driver in response to a respective transition of a shift signal.

59. The memory controller of claim 55 wherein the interface is adapted to receive a second write data value concurrently with the first write data value, and wherein the serializing driver circuit is further adapted to receive the second write data value from the request interface as a set of parallel constituent bits and to output the second write data value to the storage subsystem via a second signal line as a serial stream of constituent bits.

60. The memory controller of claim 59 wherein the serializing driver circuit is further adapted to concurrently output the first and second write data values onto the first and second signal lines, respectively.

61. The memory controller of claim 60 wherein the serializing driver circuit comprises a first output driver circuit to output the first write data value via the first signal line, and a second output driver circuit to output the second write data value via the second signal line, and wherein the first output driver

circuit and the second output driver circuit are adapted to output the first and second write data values, respectively, in response to transitions of a transmit clock signal.

62. The memory controller of claim 55 wherein the request interface is adapted to receive a mask value that indicates whether the first write data value is to be excluded from storage in the storage subsystem, and wherein the memory controller further comprises a mask logic circuit to substitute a key value for the first write data value if the mask value indicates that the first write data value is to be excluded from storage in the storage subsystem.

63. The memory controller of claim 62 wherein the mask value is a bit having either a first state or a second state, the first state indicating that the first write data value is to be excluded from storage in the storage subsystem.

64. The memory controller of claim 62 wherein the first write data value is a byte having eight constituent bits.

65. The memory controller of claim 62 wherein further comprising a request interface to output the key value to the storage subsystem.

66. A memory device comprising:

a storage array having a plurality storage elements and bit lines coupled to respective columns of the storage elements;

a receiver to receive a serial stream of at least eight bits from a first signal line, the serial stream of bits constituting a first write data value;

a deserializing circuit coupled to receive the serial stream of bits from the receiver and adapted to convert the serial stream of bits into a set of parallel bits; and

a write driver circuit coupled to the bit lines and coupled to receive the set of parallel bits from the deserializing circuit, the write driver circuit being adapted to output each bit of the parallel set of bits onto a respective

one of the bit lines to enable storage of the first write data value within the storage array.

67. The memory device of claim 66 wherein the write driver circuit is further adapted to concurrently output the parallel set of bits onto the bit lines.

68. The memory device of claim 66 wherein the deserializing circuit comprises a shift register.

69. The memory device of claim 66 further comprising a write enable circuit to compare the set of parallel bits to a key value and to generate a write enable signal having a first state if the set of parallel bits matches the key value and a second state if the set of parallel bits does not match the key value.

70. The memory device of claim 69 wherein the write driver circuit has an enable input coupled to receive the write enable signal from the write enable circuit, and wherein the write driver circuit is further adapted to output the parallel set of bits onto the bit lines if the write enable signal is in the second state.

71. The memory device of claim 69 further comprising a request interface to receive the key value.

72. The memory device of claim 69 wherein the write enable signal circuit is adapted to select the key value from a plurality of predetermined values in response to a select signal.

73. The memory device of claim 72 further comprising a request interface to receive the select signal.

74. The memory device of claim 72 further comprising a plurality of storage circuits to store the plurality of predetermined values.

75. A memory system comprising:

a memory controller to receive a plurality of write data values from a host and to output the write data values as respective serial streams of bits; and

a plurality of memory devices each coupled to receive at least one of the serial streams of bits from the memory controller and adapted convert the at least one of the serial streams of bits to a set of parallel bits for storage.

76. The memory system of claim 75 further comprising a plurality of signal lines coupled between the memory controller and the plurality of memory devices, and wherein the memory controller is adapted to output each of the write data values as a serial stream of bits via a respective one of the plurality of signal lines.

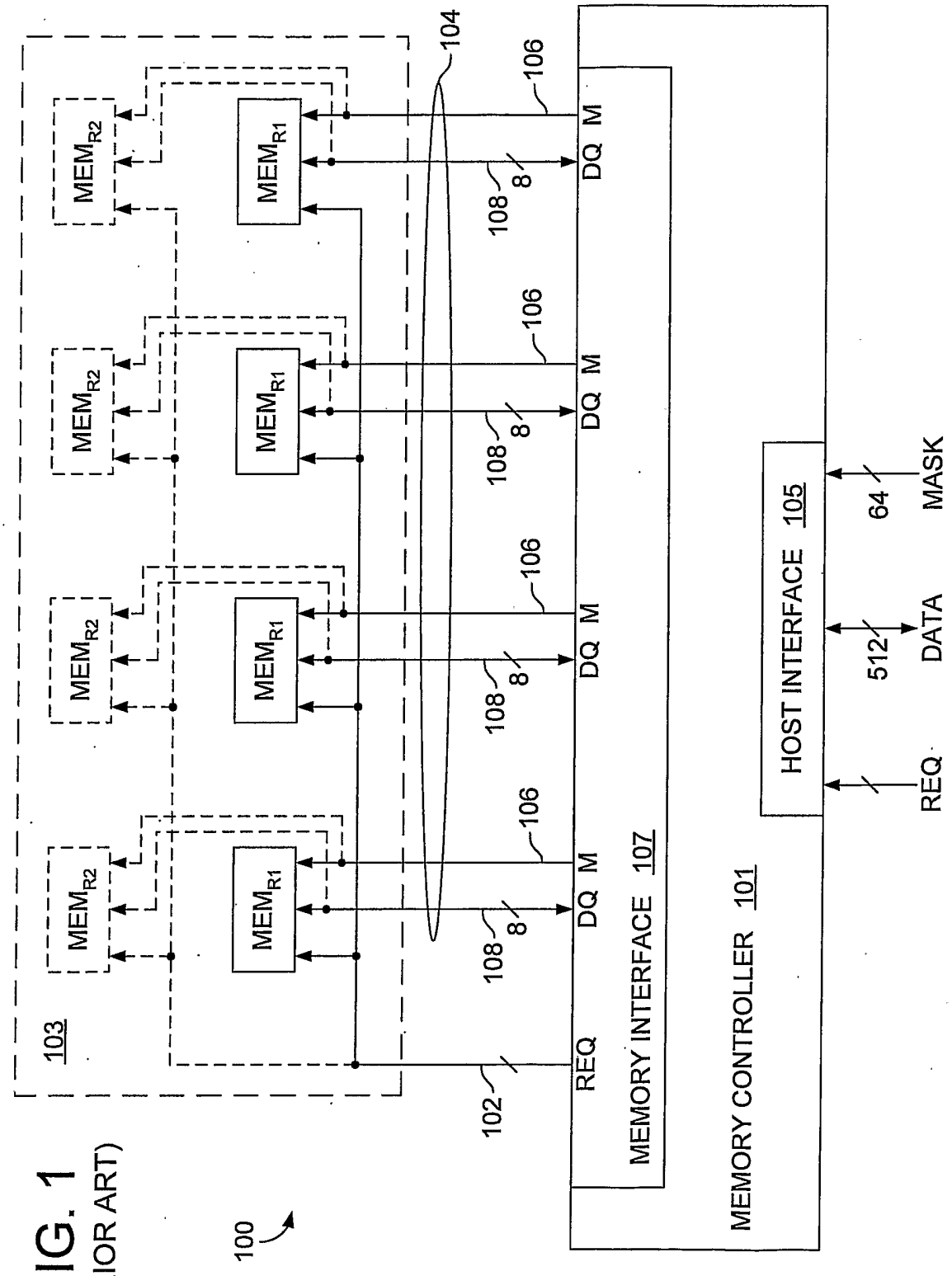


FIG. 1
(PRIOR ART)

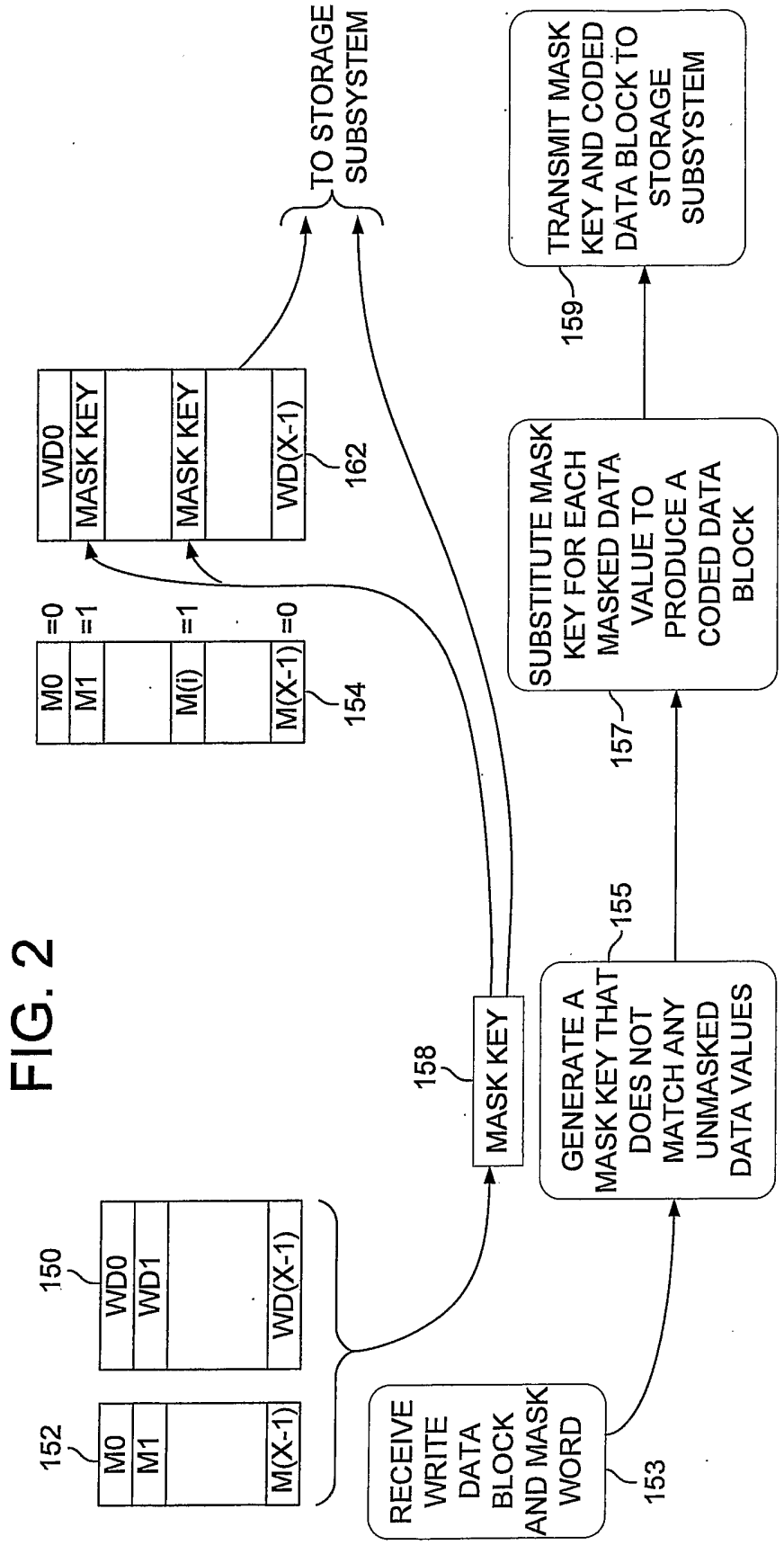

RECEIVE WRITE DATA BLOCK AND MASK WORD
153
GENERATE A MASK KEY THAT DOES NOT MATCH ANY UNMASKED DATA VALUES
155
SUBSTITUTE MASK KEY FOR EACH MASKED DATA VALUE TO PRODUCE A CODED DATA BLOCK
157
TRANSMIT MASK KEY AND CODED DATA BLOCK TO STORAGE SUBSYSTEM
159
TO STORAGE SUBSYSTEM
MASK KEY
158
M0
M1
M(X-1)
152
WD0
WD1
WD(X-1)
150
M0 =0
M1 =1
M(i) =1
M(X-1) =0
154
WD0
MASK KEY
MASK KEY
WD(X-1)
162

FIG. 2

RECEIVE CODED DATA BLOCK AND MASKED-WRITE REQUEST, MASKED WRITE REQUEST INCLUDING ADDRESS, COMMAND AND MASK KEY
175
WRITE DATA VALUE MATCHES MASK KEY?
177
NO
STORE WRITE DATA VALUE AT ADDRESS-SPECIFIED LOCATION
179

FIG. 3

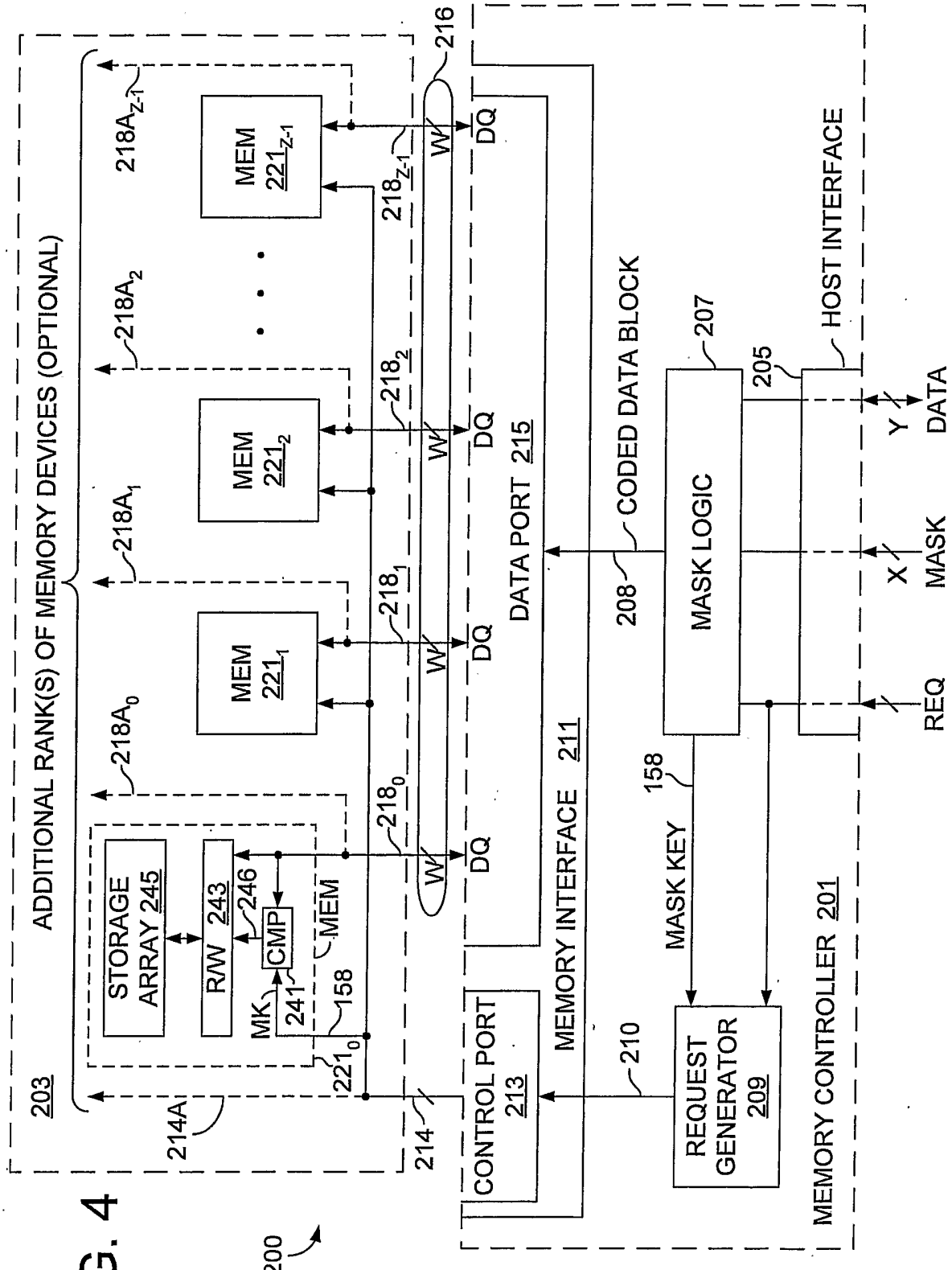

ADDITIONAL RANK(S) OF MEMORY DEVICES (OPTIONAL)
203
STORAGE ARRAY 245
R/W 243
246
CMP
MEM
MK
241
158
221_0
214A
214
$218A_0$
$218A_1$
$218A_2$
$218A_{Z-1}$
MEM 221_1
MEM 221_2
MEM 221_{Z-1}
218_0
218_1
218_2
218_{Z-1}
216
W
DQ
DATA PORT 215
MEMORY INTERFACE 211
CONTROL PORT 213
CODED DATA BLOCK
208
MASK LOGIC
207
205
HOST INTERFACE
Y
DATA
X
MASK
REQ
158
MASK KEY
210
REQUEST GENERATOR 209
MEMORY CONTROLLER 201
200

FIG. 4

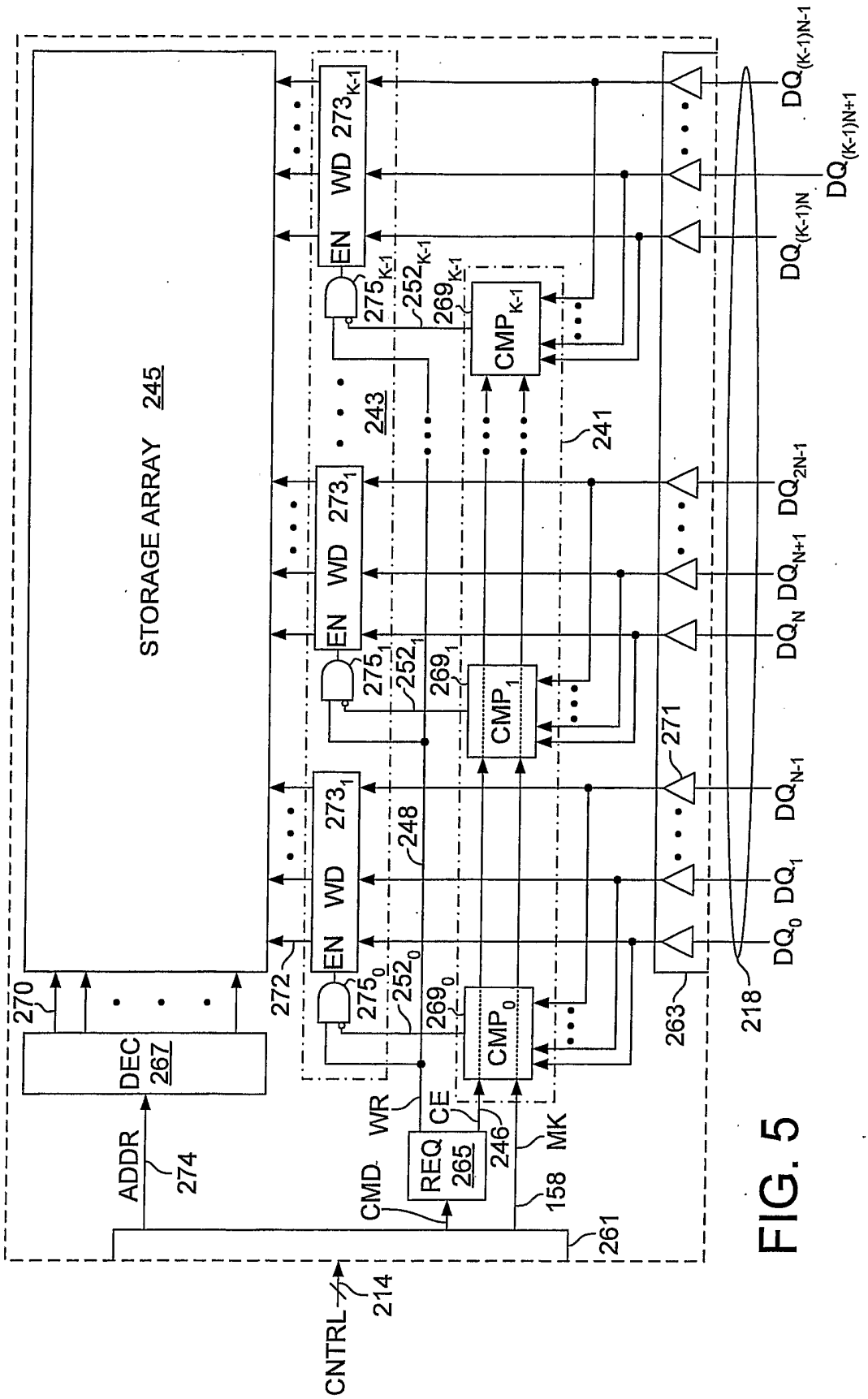

STORAGE ARRAY 245
DEC 267
REQ 265
CMP$_0$
CMP$_1$
CMP$_{K-1}$
EN WD 273$_1$
EN WD 273$_{K-1}$
ADDR 274
CMD
WR
CE
MK
CNTRL 214
261
158
246
248
241
243
270
272
263
271
218
DQ$_0$
DQ$_1$
DQ$_{N-1}$
DQ$_N$
DQ$_{N+1}$
DQ$_{2N-1}$
DQ$_{(K-1)N}$
DQ$_{(K-1)N+1}$
DQ$_{(K-1)N-1}$

FIG. 5

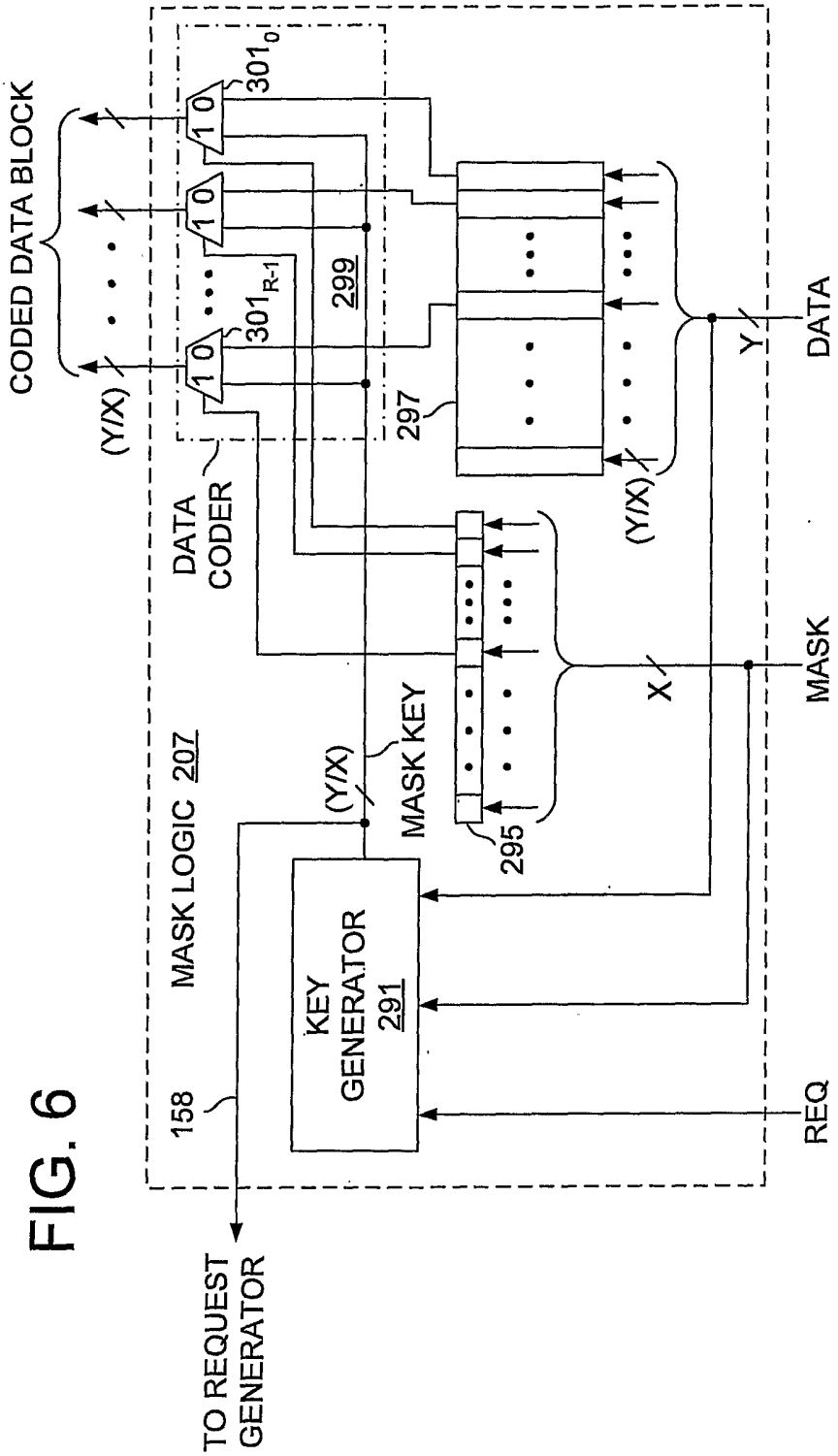

CODED DATA BLOCK
(Y/X)
1 0
301_0
301_{R-1}
299
297
DATA CODER
DATA
Y
(Y/X)
MASK
X
MASK LOGIC 207
(Y/X)
MASK KEY
295
KEY GENERATOR 291
158
REQ
TO REQUEST GENERATOR

FIG. 6

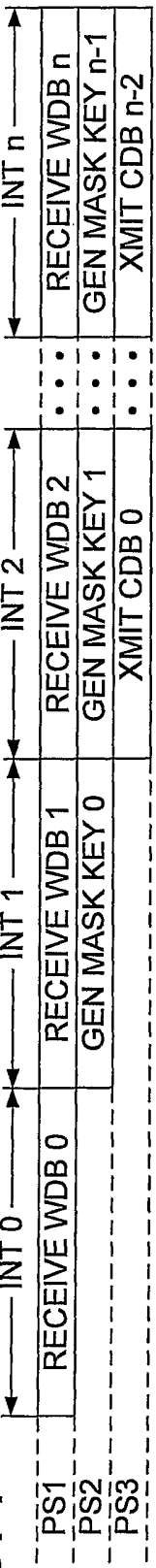

INT 0
INT 1
INT 2
INT n
PS1
PS2
PS3
RECEIVE WDB 0
RECEIVE WDB 1
RECEIVE WDB 2
RECEIVE WDB n
GEN MASK KEY 0
GEN MASK KEY 1
GEN MASK KEY n-1
XMIT CDB 0
XMIT CDB n-2

FIG. 7

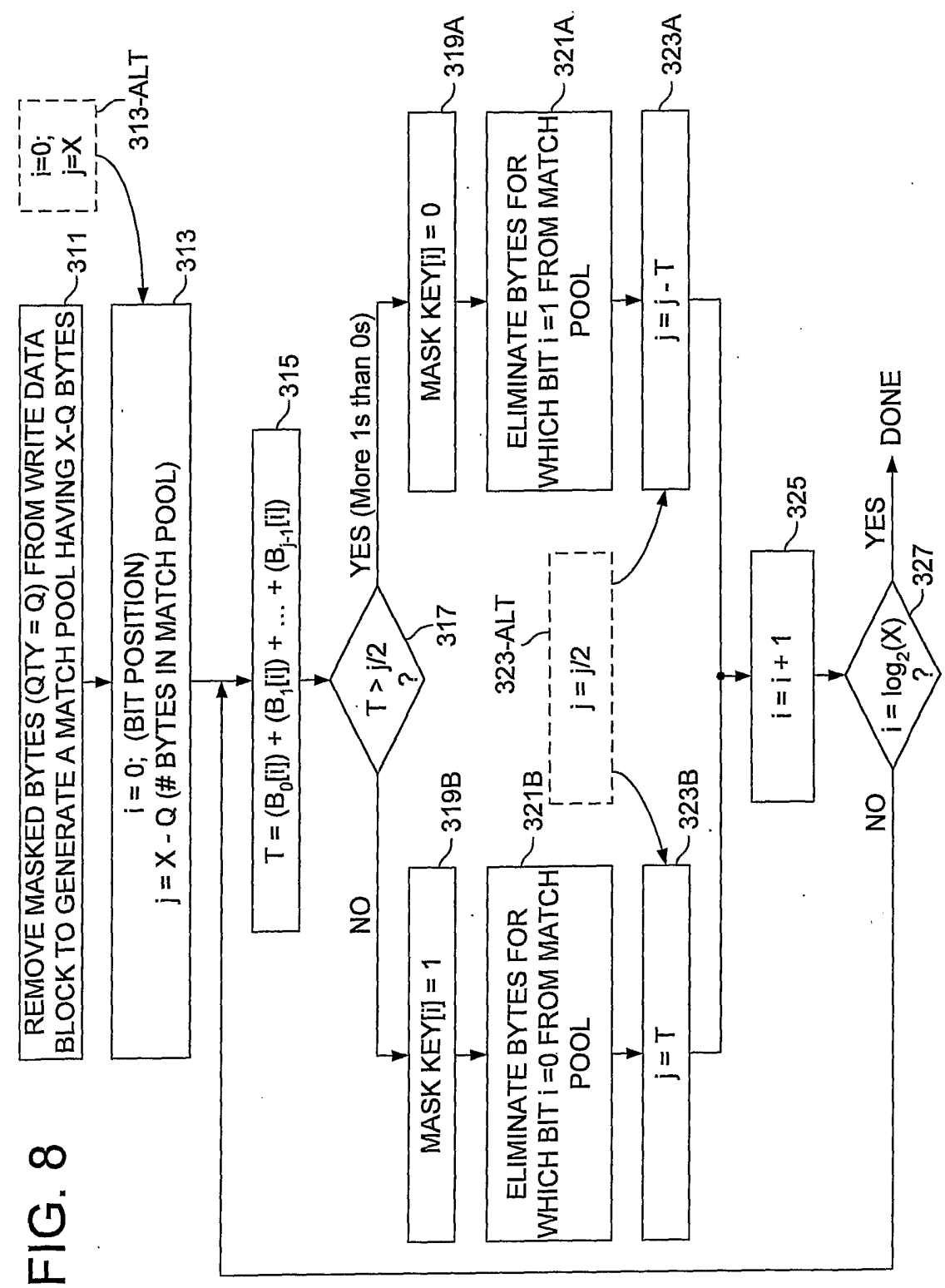

REMOVE MASKED BYTES (QTY = Q) FROM WRITE DATA BLOCK TO GENERATE A MATCH POOL HAVING X-Q BYTES
311
i=0; j=X
313-ALT
i = 0; (BIT POSITION) j = X - Q (# BYTES IN MATCH POOL)
313
T = (B0[i]) + (B1[i]) + ... + (Bj-1[i])
315
T > j/2 ?
317
YES (More 1s than 0s)
NO
MASK KEY[i] = 0
319A
ELIMINATE BYTES FOR WHICH BIT i =1 FROM MATCH POOL
321A
j = j - T
323A
MASK KEY[i] = 1
319B
ELIMINATE BYTES FOR WHICH BIT i =0 FROM MATCH POOL
321B
j = T
323B
323-ALT
j = j/2
i = i + 1
325
i = log2(X) ?
327
YES
DONE
NO

FIG. 8

FIG. 9

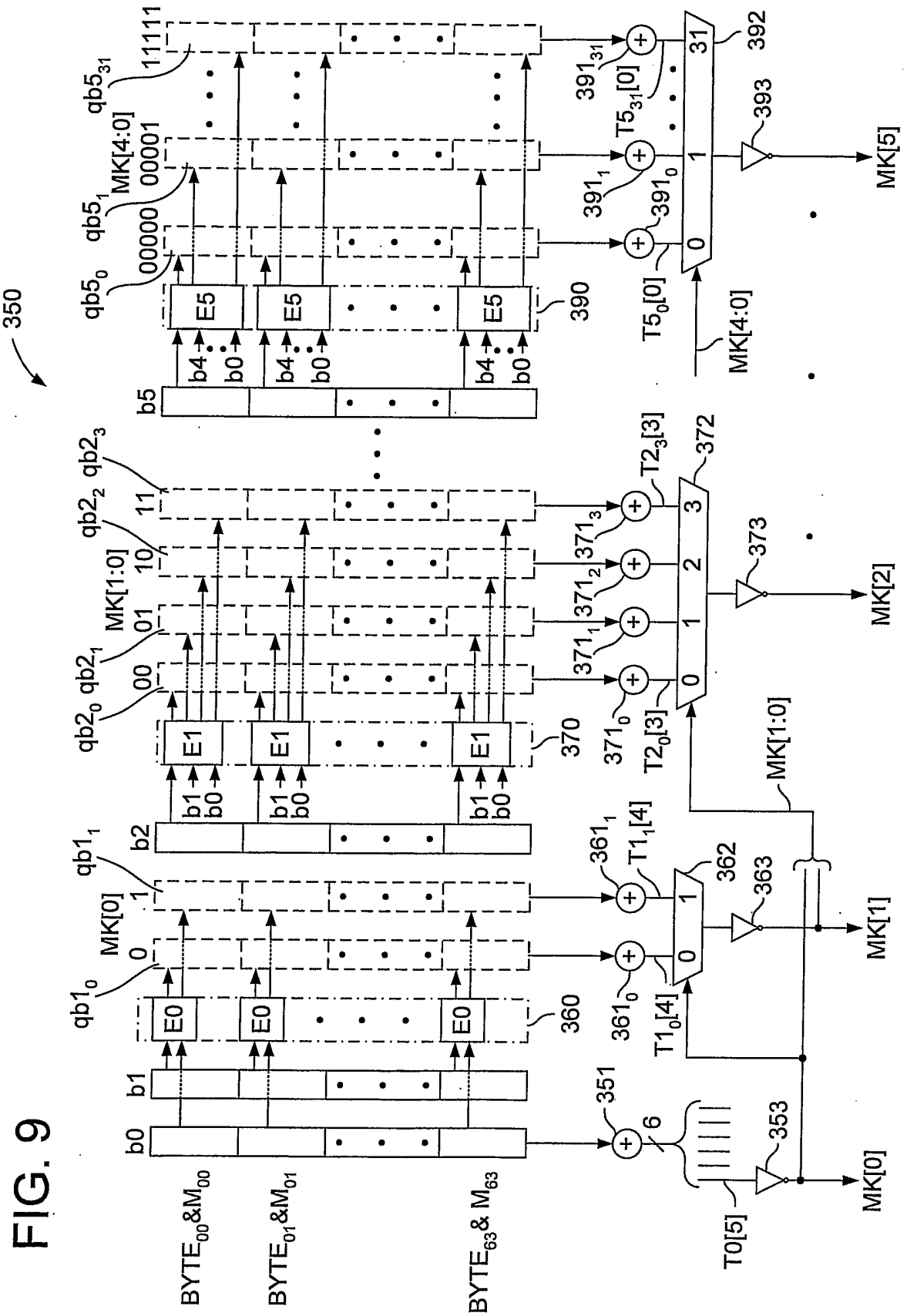

350
BYTE00&M00
BYTE01&M01
BYTE63& M63
b0
b1
b2
b5
E0
E1
E5
360
370
390
MK[0]
MK[1:0]
MK[4:0]
MK[1]
MK[2]
MK[5]
MK[0]
T0[5]
T1[4]
T2[3]
T5[0]
T5[31]
351
353
362
363
372
373
392
393

FIG. 10

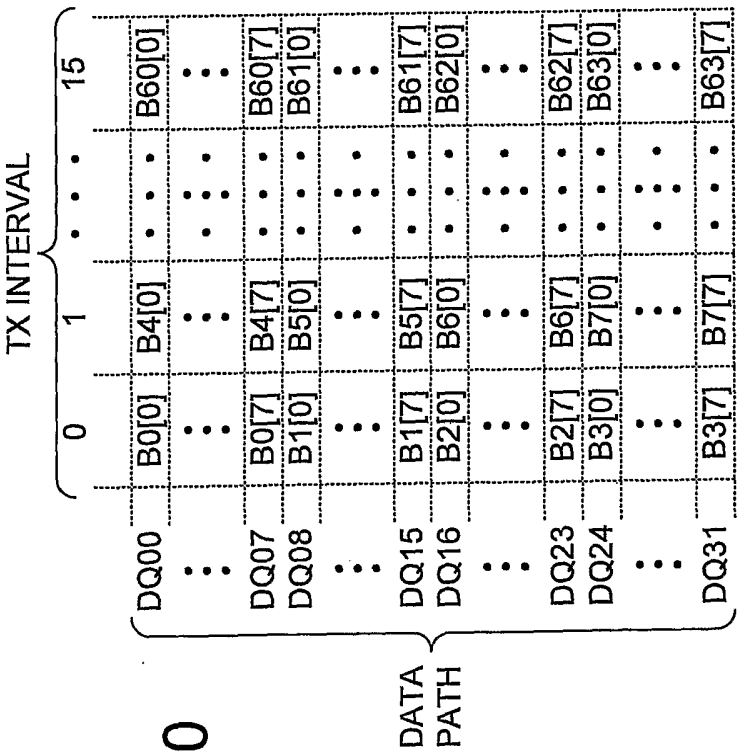


FIG. 11

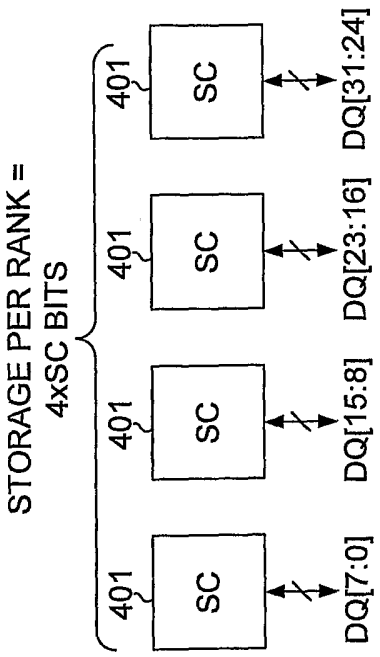


FIG. 13

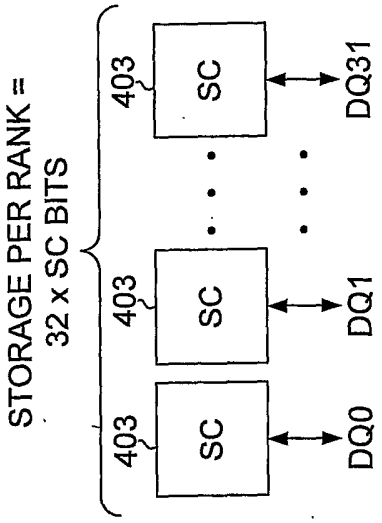


FIG. 12

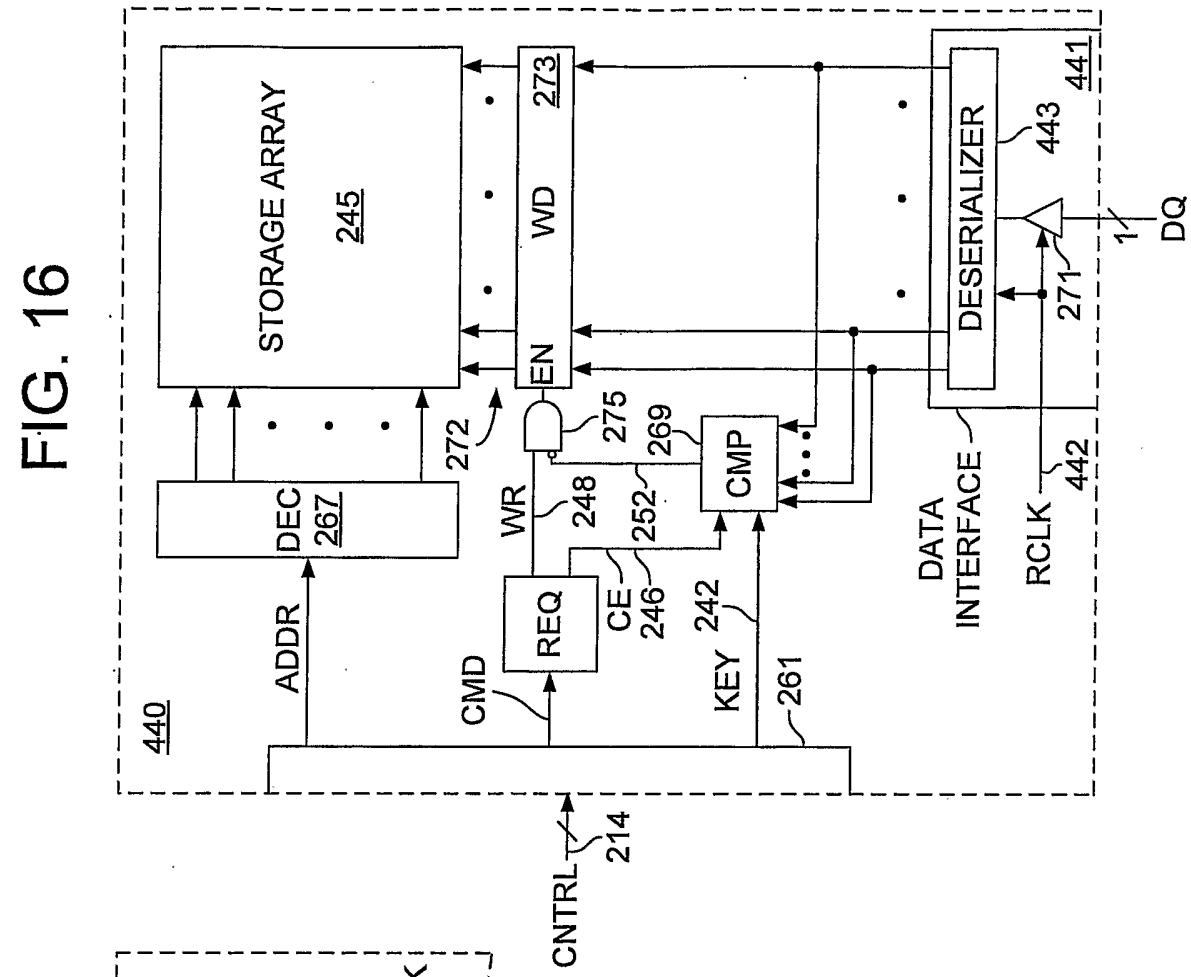


FIG. 16

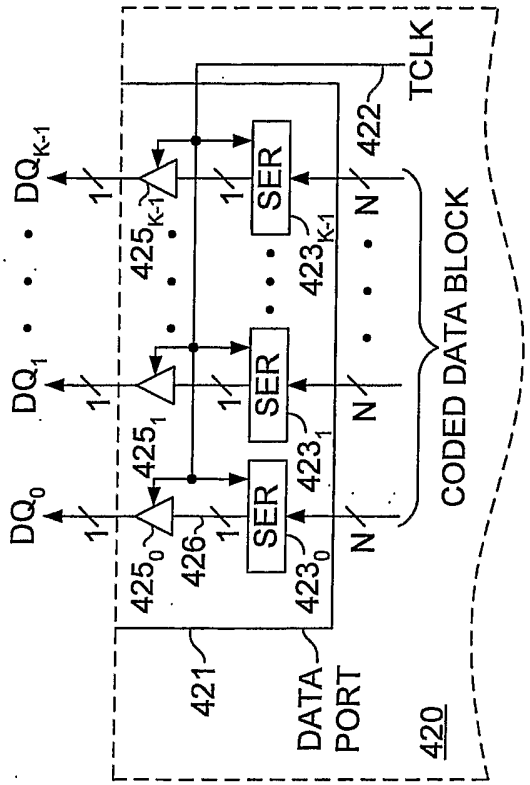


FIG. 14

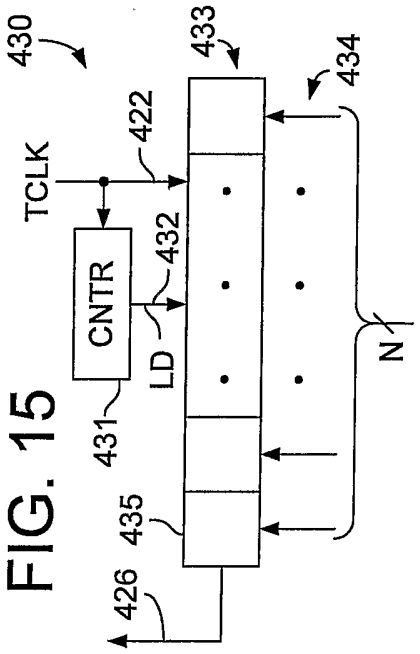
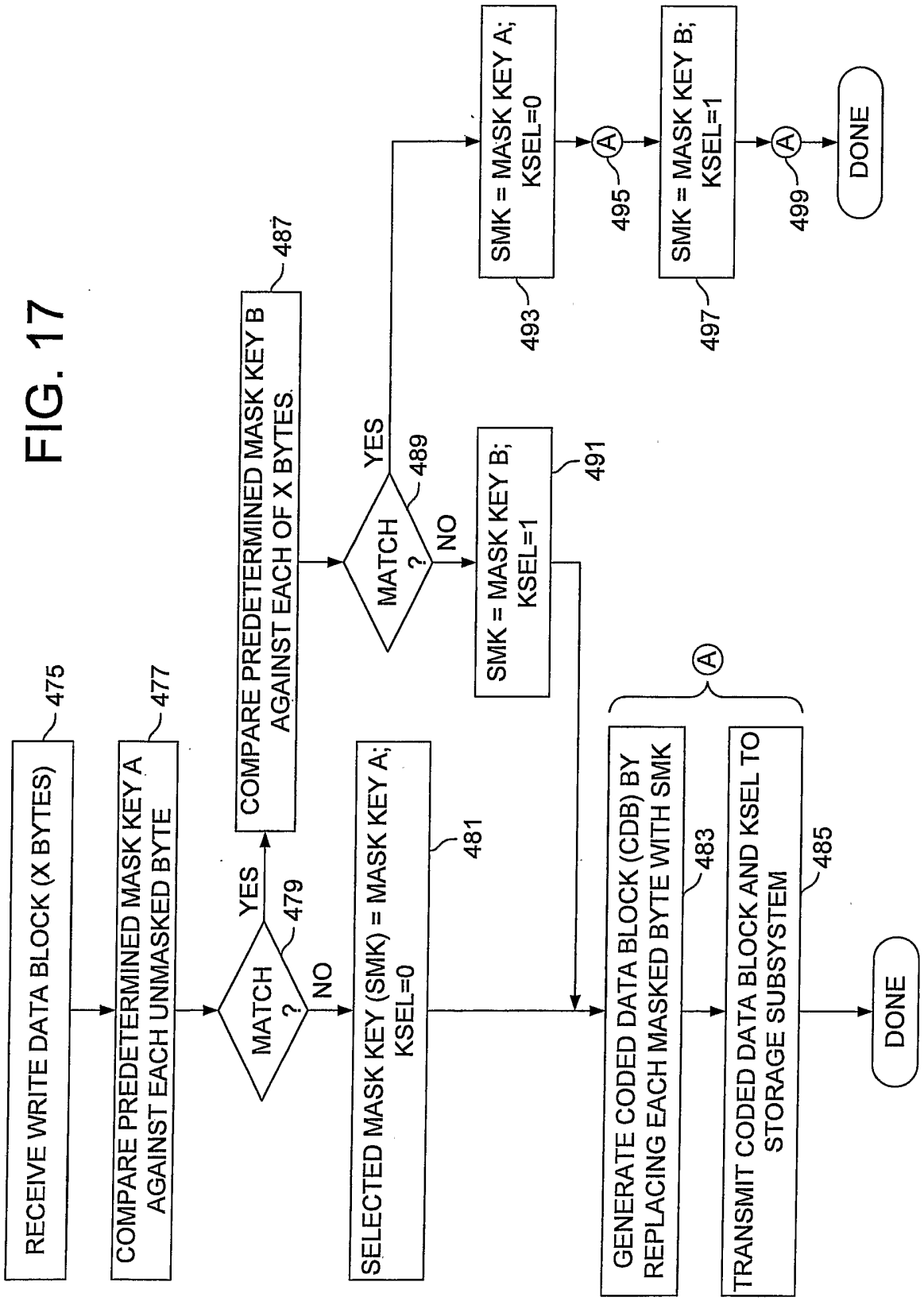


FIG. 15

FIG. 17


RECEIVE WRITE DATA BLOCK (X BYTES)
475
COMPARE PREDETERMINED MASK KEY A AGAINST EACH UNMASKED BYTE
477
MATCH ?
479
YES
NO
COMPARE PREDETERMINED MASK KEY B AGAINST EACH OF X BYTES
487
MATCH ?
489
YES
NO
SELECTED MASK KEY (SMK) = MASK KEY A; KSEL=0
481
SMK = MASK KEY B; KSEL=1
491
SMK = MASK KEY A; KSEL=0
493
SMK = MASK KEY B; KSEL=1
497
A
495
499
GENERATE CODED DATA BLOCK (CDB) BY REPLACING EACH MASKED BYTE WITH SMK
483
TRANSMIT CODED DATA BLOCK AND KSEL TO STORAGE SUBSYSTEM
485
DONE
DONE

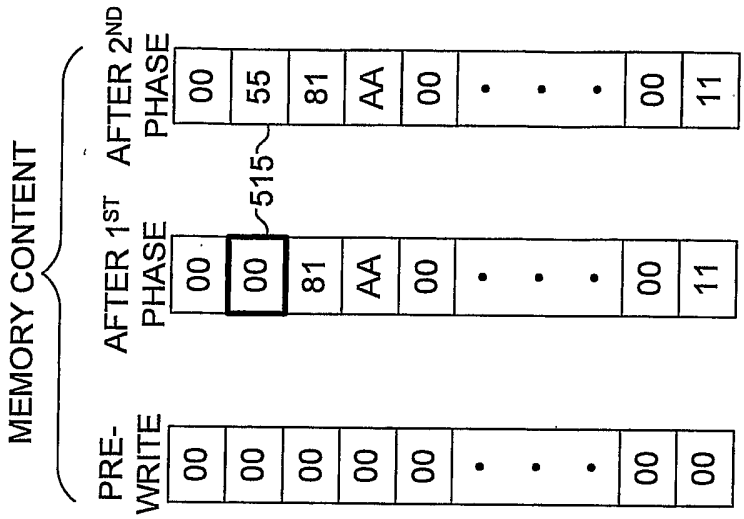

MEMORY CONTENT
PRE-WRITE
AFTER 1ST PHASE
AFTER 2ND PHASE
515
00 00 00 00 00 . . . 00 00
00 00 81 AA 00 . . . 00 11
00 55 81 AA 00 . . . 00 11

FIG. 19

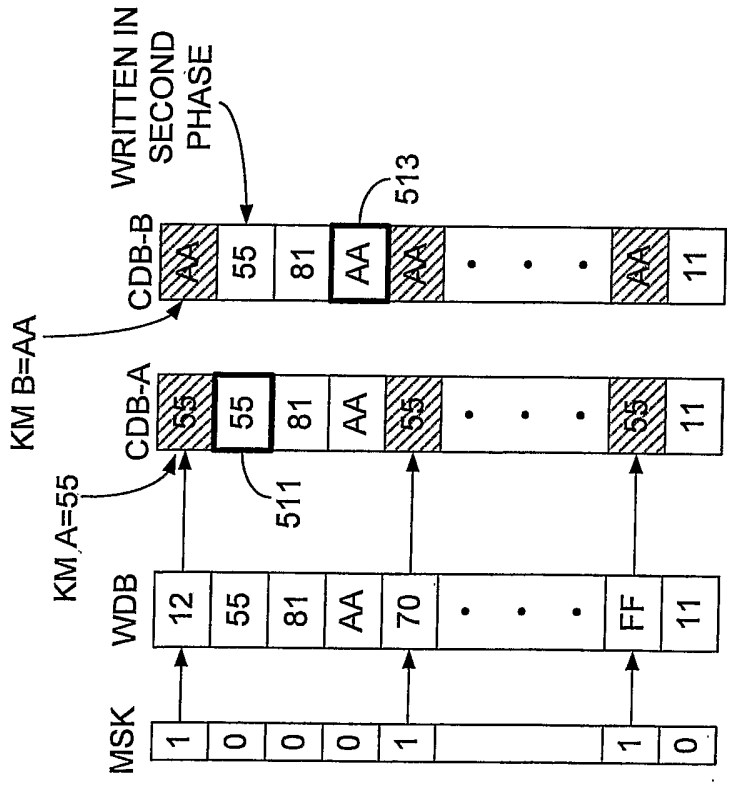

KM B=AA
KM A=55
MSK
WDB
CDB-A
CDB-B
WRITTEN IN SECOND PHASE
511
513
1 0 0 0 1 . . . 1 0
12 55 81 AA 70 . . . FF 11
55 55 81 AA 55 . . . 55 11
AA 55 81 AA AA . . . AA 11

FIG. 18

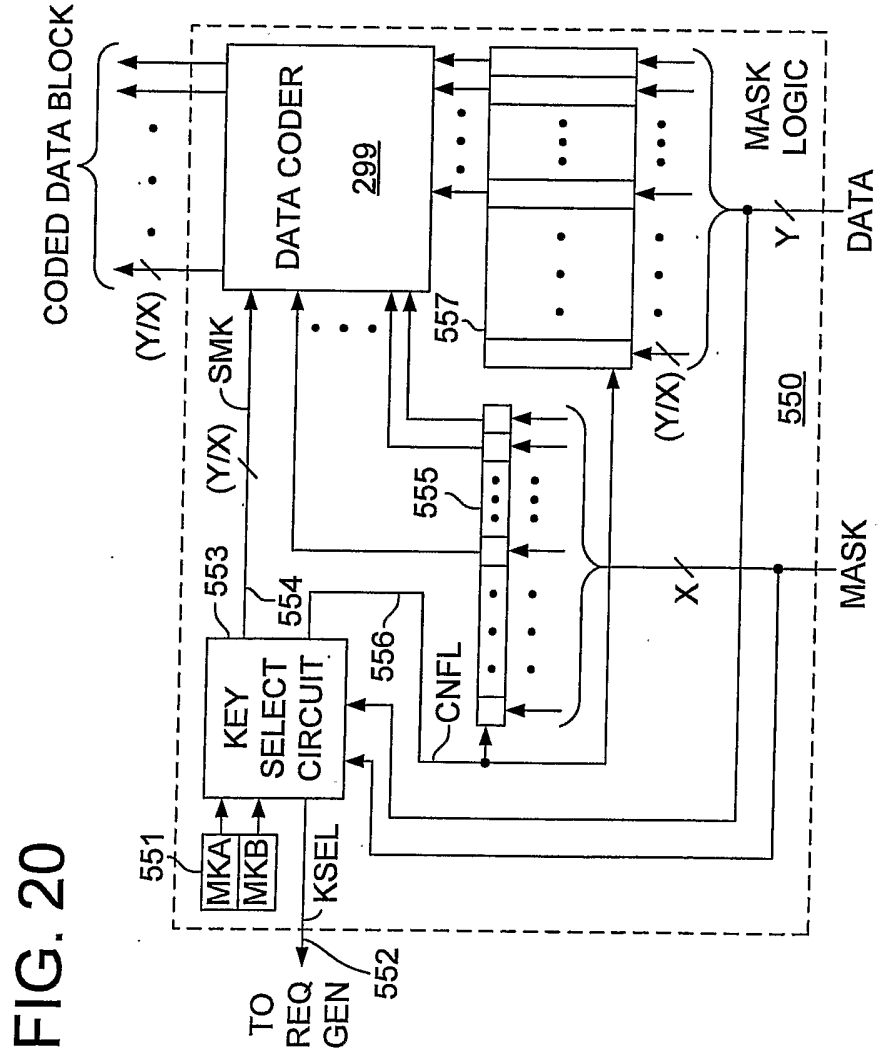

CODED DATA BLOCK
(Y/X)
DATA CODER
299
SMK
(Y/X)
557
(Y/X)
MASK LOGIC
Y
DATA
550
555
553
554
556
CNFL
X
MASK
KEY SELECT CIRCUIT
551
MKA
MKB
KSEL
TO REQ GEN
552

FIG. 20

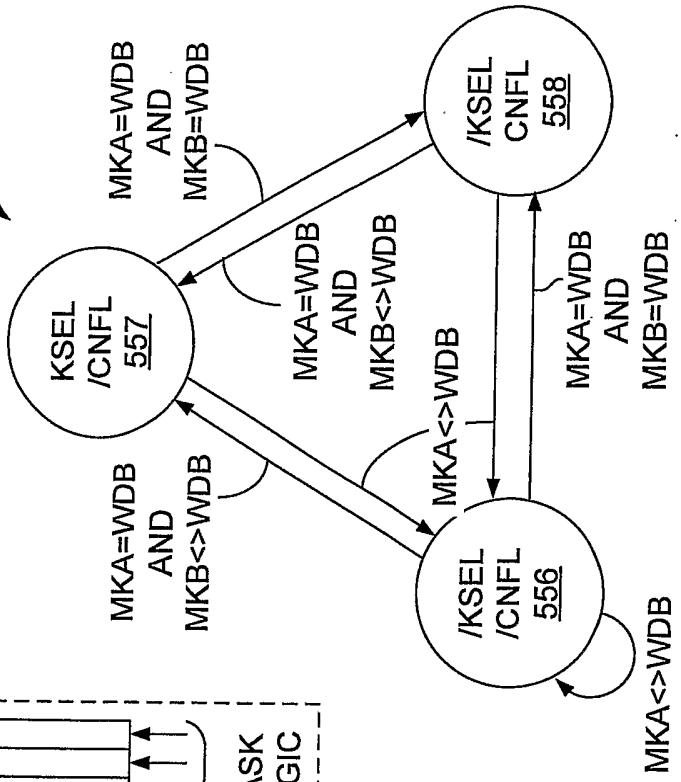

565
KSEL /CNFL 557
/KSEL CNFL 558
/KSEL /CNFL 556
MKA=WDB AND MKB=WDB
MKA=WDB AND MKB<>WDB
MKA=WDB AND MKB<>WDB
MKA<>WDB
MKA=WDB AND MKB=WDB
MKA<>WDB

FIG. 21

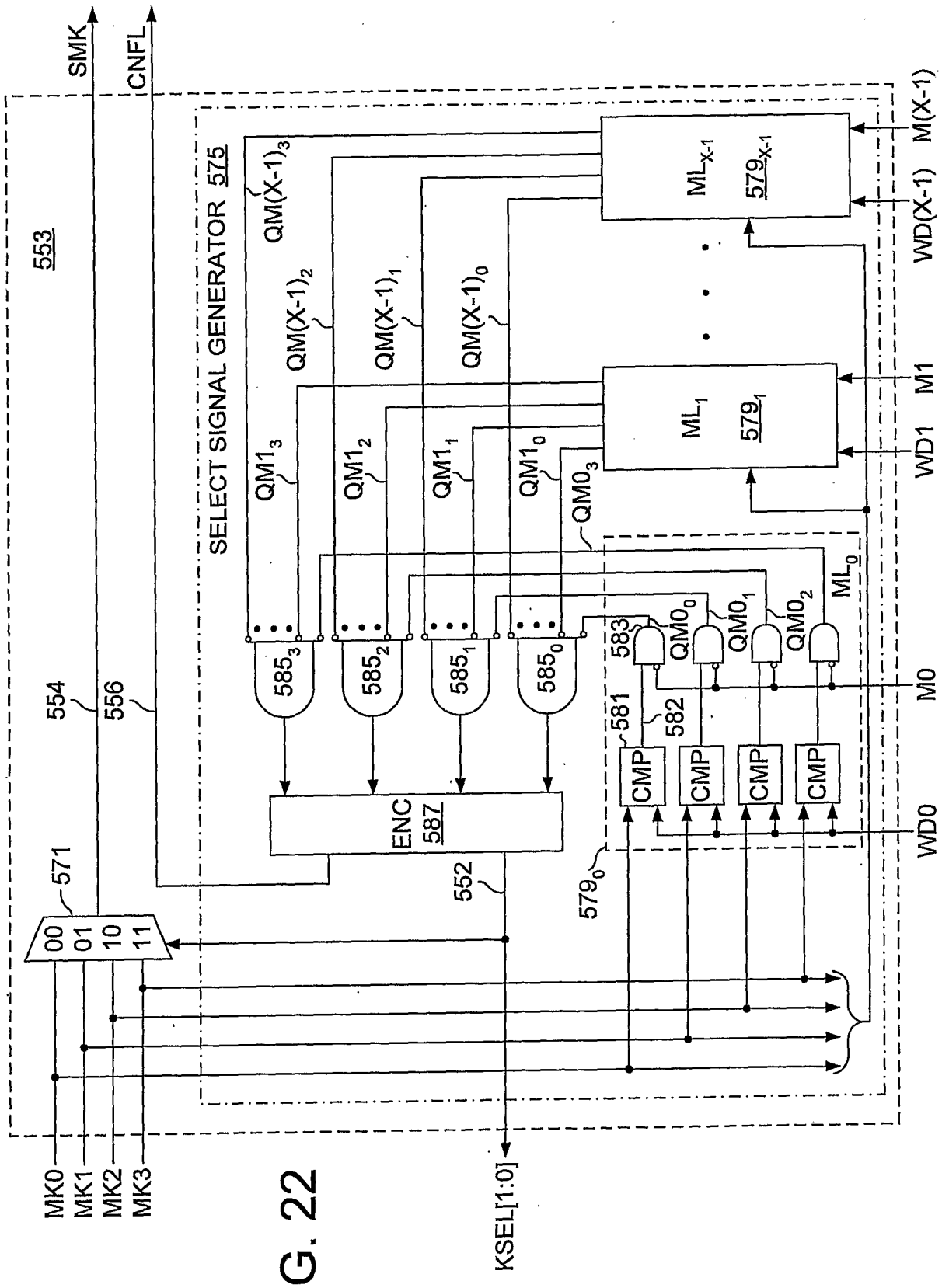

SMK
CNFL
553
SELECT SIGNAL GENERATOR 575
554
556
571
00
01
10
11
MK0
MK1
MK2
MK3
552
KSEL[1:0]
ENC 587
585$_3$
585$_2$
585$_1$
585$_0$
QM1$_3$
QM1$_2$
QM1$_1$
QM1$_0$
QM0$_3$
QM(X-1)$_3$
QM(X-1)$_2$
QM(X-1)$_1$
QM(X-1)$_0$
ML$_{X-1}$
579$_{X-1}$
ML$_1$
579$_1$
ML$_0$
579$_0$
583
581
582
QM0$_0$
QM0$_1$
QM0$_2$
CMP
M(X-1)
WD(X-1)
M1
WD1
M0
WD0

FIG. 22

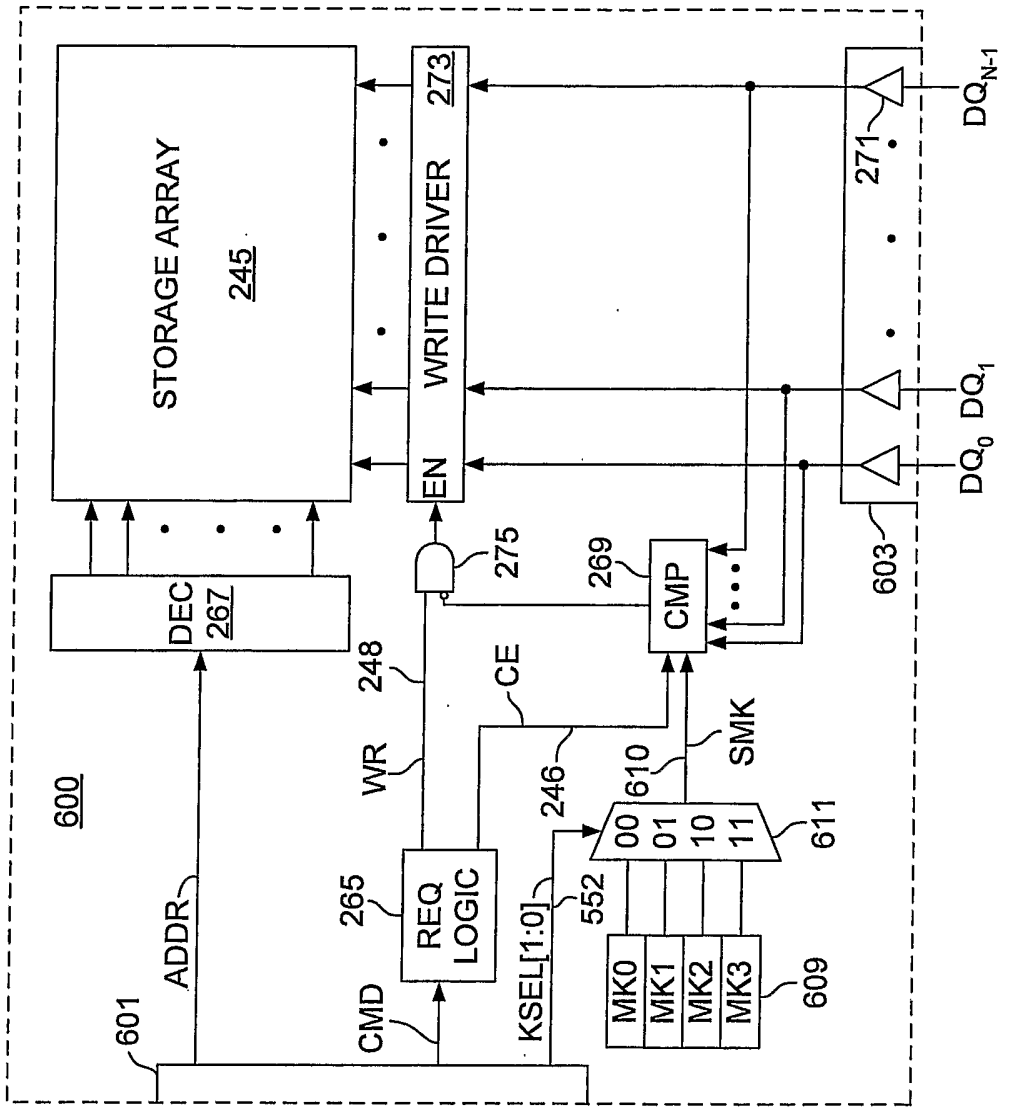

STORAGE ARRAY
245
WRITE DRIVER
273
EN
DEC
267
275
269
CMP
271
603
DQ_{N-1}
DQ_1
DQ_0
600
601
ADDR
CMD
248
WR
CE
246
610
SMK
265
REQ LOGIC
552
KSEL[1:0]
00
01
10
11
611
MK0
MK1
MK2
MK3
609

FIG. 23

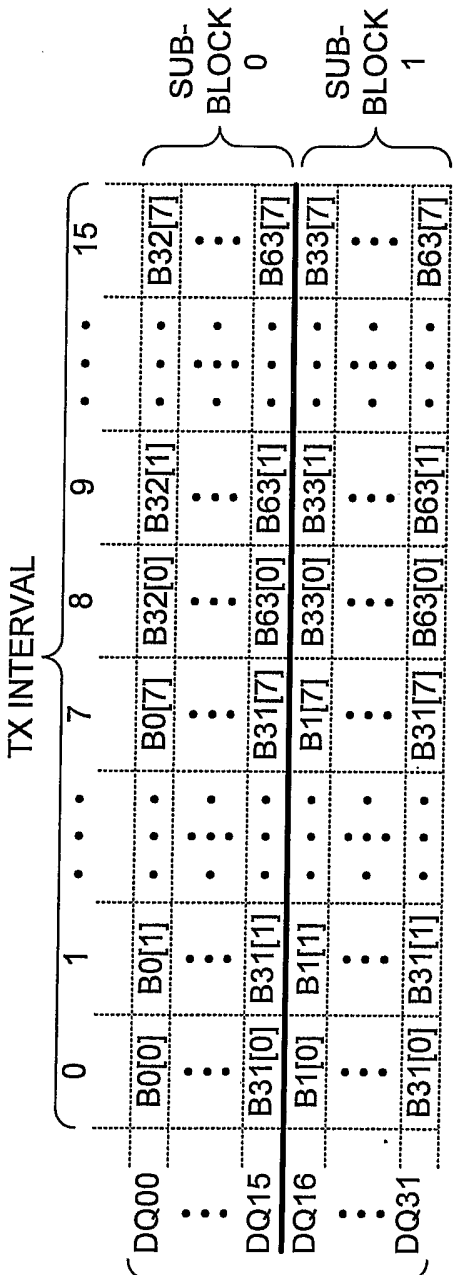


FIG. 24

FIG. 25

KSEL[1:0]	COMMAND
00	UNMASKED WRITE
01	MASKED WRITE WITH KEY 0
10	MASKED WRITE WITH KEY 1
11	MASKED WRITE WITH KEY 2

FIG. 26