

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
28 February 2008 (28.02.2008)

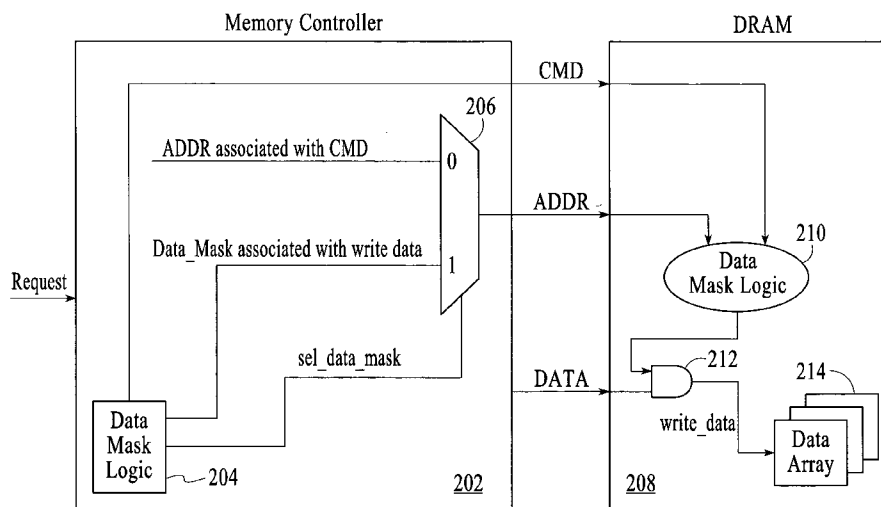
PCT

(10) International Publication Number
WO 2008/024336 A2

- (51) International Patent Classification: **Not classified**
- (21) International Application Number: PCT/US2007/018469
- (22) International Filing Date: 21 August 2007 (21.08.2007)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data: 11/509,143 23 August 2006 (23.08.2006) US
- (71) Applicant (for all designated States except US): **ATI TECHNOLOGIES, INC.** [CA/CA]; 1 Commerce Drive East, Markham, Ontario, L3T 7X6 (CA).
- (72) Inventors; and
- (75) Inventors/Applicants (for US only): **XU, Xiaoling** [US/US]; 1712 Harrison Street, Santa Clara, CA 95050 (US). **KRUGER, Warren, F.** [US/US]; 1309 Dunnock Way, Sunnyvale, CA 94087 (US).
- (74) Agents: **RICHARD, Gregory, L.** et al.; Courtney Staniford & Gregory Llp, P.O. Box 9686, San Jose, CA 95157 (US).
- (81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.
- (84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, 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, IS, IT, LT, LU, LV, MC, MT, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: WRITE DATA MASK METHOD AND SYSTEM



(57) Abstract: In various embodiments, dedicated mask pins are eliminated by sending a data mask on address lines of the interface. A memory controller receives a request for a memory write operation from a memory client and determines the granularity of the write data from a write data mask sent by the client. Granularity, as used herein, indicates a quantity of write data to which each bit of the received write data mask applies. In an embodiment, the memory controller generates a particular write command and a particular write data mask based on the granularity of the write data. The write command generated is typically the most efficient of several write commands available, but embodiments are not so limited. The write command is transmitted on command lines of the interface, and the write data mask is transmitted on address lines of the interface.

WO 2008/024336 A2



Published:

- *without international search report and to be republished upon receipt of that report*

WRITE DATA MASK METHOD AND SYSTEM

Inventors:

Xiaoling Xu
Warren F. Kruger

5

TECHNICAL FIELD

10 The invention is in the field of data transfer in computer and other digital systems.

BACKGROUND

As computer and other digital systems become more complex and more capable, methods and hardware to enhance the transfer of data between system
15 components or elements continually evolve. Data to be transferred includes digital signals representing data, commands, addresses, or any other information. System components or elements can include different functional hardware blocks on a single integrated circuit (IC), or on different ICs. The different ICs may or may not be on the same printed circuit board (PCB). System components typically include an
20 input/output (I/O) interface specifically designed to receive data from other system components and to transmit data to other system components.

One consistent trend as computing systems become more capable is an increase in the amount of data to be transferred per time period. Some applications that require high data rates include game consoles, high definition television (HDTV),
25 personal computers (PCs) main memory, graphics processors, and various consumer devices not already mentioned. In response to the demand for increased data rates, double data rate (DDR) standards have been developed to standardize the behavior of hardware and software using high data rates. Several generations of graphics DDR (GDDR) standards have been developed specifically for graphics processing and
30 video processing, which typically demand the capability to transfer and process very large amounts of data.

In some instances, it is necessary to write only a portion of the bits in a transmission. For example, in a system that transmits 32 bytes, or 256 bits per command, some but not all of the 32 bytes might be written in a write operation. This

will be referred to as a byte-write operation herein. One conventional byte-write method is to read the 256 bits that are in the DRAM, modify the portion to be changed, and write all of the 256 bits back. This method is commonly used in central processing units (CPUs), and is referred to as a read-modify-write operation.

5 Unfortunately, for high-speed applications, such as graphics applications, read-modify-write operations are both too slow and too complex to implement. One reason is that graphics processors often use a complex operation reordering schemes in order to optimize the memory interface.

10 So historically DRAMs provide one additional data mask pin for every eight data pins, which indicates whether the eight bit data should be written or not. This data mask method is referred to in existing DRAM specifications. One disadvantage of this method is it results in one extra pin for every eight data bit pins, thus increasing the pin count.

15 In addition, the existing data mask method in existing DRAM specifications is not applicable in high speed interfaces which are susceptible to data bit errors. One reason is that the errors on the data mask pins are fatal. This is due to the fact that an error on the data mask pin may result in a byte write that was not intended. The data in that byte would be destroyed (overwritten) and could not be retried. In addition, an error on the data mask will make the error detection/correction feature provided by
20 certain DRAMs useless.

Figure 1A is a block diagram of components of a prior art digital system 100. The components include a processor 102 and a memory component 104. The processor 102 controls the memory component 104 by communicating over an interface that includes command lines, address lines, data lines, and separate,
25 dedicated data mask lines. The memory component 100 is a DDR memory that communicates with the processor 102 over a DDR interface.

Figure 1B is a timing diagram illustrating a write operation 106 with a data mask, as performed by the prior art components of Figure 1A. In this illustration, a burst-of-8 write transaction has been issued. The top waveform is the clock (CLK) waveform. The waveform below the CLK waveform shows the command waveform
30 (CMD) on the command lines. Three write commands are shown, write 108, write 110 and write 112. The data waveform is shown below the command waveform. In each of the valid data periods labeled 1-8, eight bits of data are transmitted. A data mask waveform is shown below the data waveform. The data mask waveform

indicates the values being transmitted on the data mask lines. For write 108, the data mask line is high during the entire write. In this illustration, a low data mask bit, or a 0 logic level on a data mask line, indicates that the corresponding byte is to be written. So, for write 108, none of the bytes 108 are to be written.

5 Write 110 differs from write 108. Write 110 is a byte-write in which only particular bytes of the eight bytes are to be written. Referring to the data mask waveform for write 110, it can be seen that the data mask has a value of 01001000. Therefore, data mask pins and lines are required to be available for each write operation, even if byte-write operations make up a small percentage of write
10 operations, as is often the case in many modern applications.

 One approach to allowing byte-writes without the use of a dedicated mask pin is a reusable data mask. In DRAMs that support this capability, a memory controller can program the data mask into a register inside the DRAM and reuse it later. However, for many applications, including many graphics application, the number of
15 reusable mask is small, so the performance improvement may not be very great.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1A is a block diagram of components of a prior art digital system.

Figure 1B is a timing diagram illustrating a write operation with a data mask, as performed by the prior art components of Figure 1A.

5 **Figure 2** is a block diagram of elements of a system, according to an embodiment.

Figure 3 is a timing diagram illustrating a write operation with a write_without_mask command, according to another embodiment.

10 **Figure 4** is a timing diagram illustrating a write operation with a write_multiple_mask command, according to an embodiment.

Figure 5 is a timing diagram illustrating a write operation with a write_single_mask command, according to an embodiment.

Figure 6 is a flow diagram of a method performed by a memory controller, according to an embodiment.

15 **Figure 7** is a flow diagram of a method performed by a memory device, according to an embodiment.

Figure 8 is a diagram showing an alternative mapping of write data mask bits to data bytes according to an embodiment.

20 In the drawings, the same reference numbers identify identical or substantially similar elements or acts. To easily identify the discussion of any particular element or act, the most significant digit or digits in a reference number refer to the Figure number in which that element is first introduced (e.g., element 102 is first introduced and discussed with respect to Figure 1).

DETAILED DESCRIPTION

A method and apparatus for efficiently performing byte-write via a high-speed memory interface is described herein. As used herein a byte-write is a write of less than all of the data accommodated by a command cycle on the high-speed memory interface. In various embodiments, dedicated mask pins are eliminated by sending a data mask on address lines of the interface. A memory controller receives a request for a memory write operation from a memory client and determines the granularity of the write data from a write data mask sent by the client. Granularity, as used herein, indicates a quantity of write data to which each bit of the received write data mask applies. In an embodiment, the memory controller generates a particular write command and a particular write data mask based on the granularity of the write data. The write command generated is typically the most efficient of several write commands available, but embodiments are not so limited. The write command is transmitted on command lines of the interface, and the write data mask is transmitted on address lines of the interface. In the case of some write commands, circumstances, no additional cycles are required to perform a byte-write.

Figure 2 is a block diagram of elements of a system 200 according to an embodiment. The system 200 includes a memory controller 202 and a memory device 208. In an embodiment, the memory device 208 is a double data rate dynamic random access memory (DDRAM), but embodiments are not so limited. The memory controller 202 includes data mask logic 204 and selection logic 206. The memory controller 202 receives requests for memory access from various memory clients (not shown) in the system 200. As further described herein, the data mask logic examines write requests received and determines whether they include a write mask. If a received write request includes a write mask, the data mask logic determines which one of multiple data mask types should be used in performing the write operation. A write data mask type implies both a write data mask and a corresponding write command, as further described below. In an embodiment, the write data mask type is chosen based on which type would be most efficient, but in an alternative embodiment, the requester could specify a write data mask type.

When a write data mask type is determined, the data mask logic 204 generates the appropriate data mask associated with the write data and transmits it to the selection logic 206. The appropriate write command is generated as shown. In addition, the address associated with the write command is transmitted to the selection

logic 206. The data mask logic 204 also generates a sel_data_mask signal that selects either the address or the data mask to be transmitted to the memory device 208 on the address lines of the interface.

The memory device 208 includes data mask logic 210 that receives the
5 command on the command lines, the address and write data mask on the address lines. The data mask logic 210 determines from the command whether there is a write data mask, and also how to interpret the write data mask. Write data logic 212 receives the write data on data lines of the interface and also receives appropriately interpreted mask data from the data mask logic 210. The mask data includes individual bits that
10 are interpreted to apply to one or more bytes of write data. In an embodiment, a bit of mask data that applies to one or more particular bytes of write data is a 0 if the one or more particular bytes are to be written to data array 214, and the bit of mask data is a 1 if the one or more particular bytes are not to be written to the data array 214. Other coding schemes are equally applicable.

15 Figures 3-5 are timing diagrams used to illustrate the write commands and write data masks according to data mask types of particular embodiments. Figures 3-5 are specific examples offered for the purpose of thoroughly describing the claimed invention, but they are not intended to be limited in any way. For example, in Figures 3-5 a high-speed memory interface that transfers 32 bytes of data per command is
20 illustrated, but embodiments are applicable to other interface designs. An example protocol illustrated by Figures 3-5 is a graphics double-data-rate (GDDR) protocol such as GDDR5.

Figure 3 is a timing diagram illustrating a write operation with a write_without_mask command, referenced as WOM. At the top of the diagram is a
25 clock (clk) waveform showing the clock signal relevant to transmissions on the memory interface. A command line waveform (CMD) shows transmissions on the command lines. An address line waveform (ADDR) shows transmissions on the address lines. A data waveform (DATA) shows transmissions on the data lines.

One complete write operation will be described with reference to the first
30 write_without_mask command, WOM0, although additional, similar write operations are shown in Figure 3. The memory controller 202 has determined that there is no write data mask associated with a write request received from a memory client. Therefore, all bytes of data that are transmitted in the write operation are to be written to the memory device 208. The address associated with the WOM0 is 16 bits divided

into 8 low address bits (A0L) and 8 high address bits (A0H). In an embodiment, a “no-operation” or NOP command follows WOM0. Here, we assume the specific application is doing a long burst of write, such that a NOP is inserted here to avoid data bus corruption. This NOP is not required by DRAM protocol, but will appear for many applications. Accordingly, the address lines are idle for the remainder of the available cycle time. In response to WOM0, 32 bytes of data D00-D31 are transmitted on the data lines of the interface and all are written to the memory device 208.

Figure 4 is a timing diagram illustrating a write operation with a write_multiple_mask command, referenced as WDM in this example. The WDM command implies two bytes of data per write data mask bit as further described (“write double mask”), but in other embodiments the number could be three or more bytes of data per write data mask bit.

At the top of the diagram is a clock (clk) waveform showing the clock signal relevant to transmissions on the memory interface. A command line waveform (CMD) shows transmissions on the command lines. An address line waveform (ADDR) shows transmissions on the address lines. A data waveform (DATA) shows transmissions on the data lines.

One complete write operation will be described with reference to the first write_multiple_mask command, WDM0, although additional, similar write operations are shown in Figure 4. The memory controller 202 has determined that there is a write data mask associated with a write request received from a memory client. The memory controller has further determined that the granularity of the write data is two bytes. That is, one bit of the write data mask applies to two bytes of write data. The address associated with the WDM0 is 16 bits divided into 8 low address bits (A0L) and 8 high address bits (A0H). In an embodiment, a “no-operation” or NOP command follows WDM0. This NOP is not required by DRAM protocol, but will appear for many applications. This available cycle time is used to transmit the write data mask as two 8-bit data mask portions, DDM00 and DDM01. Thus, in this case no additional time or pins are required to convey write data mask information to the memory device 208. In response to WDM0, 32 bytes of data D00-D31 are transmitted on the data lines of the interface and various bytes are written to the memory device 208 according to the write data mask. The write data mask includes 16 bits, each of which applies to a particular pair of bytes. That is, each write data

mask bit is set or cleared to indicate whether or not to write a particular pair of bytes to the memory 208.

Figure 5 is a timing diagram illustrating a write operation with a write_single_mask command, referenced as WSM in this example. The WSM
5 command implies one bit of write data mask per write data byte.

At the top of the diagram is a clock (clk) waveform showing the clock signal relevant to transmissions on the memory interface. A command line waveform (CMD) shows transmissions on the command lines. An address line waveform (ADDR) shows transmissions on the address lines. A data waveform (DATA) shows
10 transmissions on the data lines.

One complete write operation will be described with reference to the first write_single_mask command, WSM0, although additional, similar write operations are shown in Figure 5. The memory controller 202 has determined that there is a write data mask associated with a write request received from a memory client. The
15 memory controller has further determined that the granularity of the write data is one byte. That is, one bit of the write data mask applies to one byte of write data. The address associated with the WSM0 is 16 bits divided into 8 low address bits (A0L) and 8 high address bits (A0H). In an embodiment, a “no-operation” or NOP command follows WSM0. This NOP is not required by DRAM protocol, but will
20 appear for many applications.

The write data mask associated with the WSM0 is 32 bits divided into four 8-bit portions, DM00, DM01, DM02, and DM03.

This available cycle time afforded by the NOP command is used to transmit DDM00 and DDM01. An additional NOP command is inserted to allow the
25 transmission of DDM02 and DDM03. Thus, in this case there is a penalty for using such a fine granularity, but in various cases this is justified. In response to WSM0, 32 bytes of data D00-D31 are transmitted on the data lines of the interface and various bytes are written to the memory device 208 according to the write data mask.. The write data mask includes 32 bits, each of which applies to a particular byte. That is,
30 each write data mask bit is set or cleared to indicate whether or not to write a particular byte to the memory 208.

In various common applications, such as graphics applications, WDM and WOM will be used a great percentage of the time, due in part to the prevalence of 16-bit granularity data. In the case of a “mandatory” NOP cycle after a write command,

the WDM command causes no performance degradation. Although the NOP cycle is not mandatory, in many applications such as graphic applications, the NOP is required to avoid data bus contention. So in most such applications any performance degradation is insignificant.

5 **Figure 6** is a flow diagram of a method 600 performed by a memory controller such as the memory controller 202 according to an embodiment. At 602, the memory controller determines whether a request received from a memory client is a write request. If the received request is a write request, the memory controller then determines at 604 whether a data mask is needed. In an embodiment, the memory
10 controller reads a data mask supplied by the client in the request, but embodiments are not so limited. If a data mask is not needed, the memory controller issues a WOM command and transmits the appropriate address bits at 606.

 If a data mask is needed, the memory controller determines whether the granularity of the data mask is one byte at 608. If the granularity of the data mask is
15 not one byte, the memory controller issues a WDM command and transmits the appropriate address bits at 610. If the granularity of the data mask is one byte, the memory controller issues a WSM command and transmits the appropriate address bits at 612.

 In other embodiments, the granularity of the data mask could be more than
20 two. In other embodiments, the desired data mask granularity and/or the desired write command could be specified by the client in the request.

Figure 7 is a flow diagram of a method 700 performed by a memory device such as the memory device 208 according to an embodiment. At 702 the memory device determines whether it has received a WOM command. If the memory device
25 received the WOM command, the memory device performs a write at 704 of all of the received data to a location indicates by received address bits.

 If the memory device did not receive a WOM command, the memory device determines whether it has received a WDM command at 706. If the memory device received a WDM command, the memory device performs a write of particular pairs of
30 bytes of write data as indicates by the write data mask at 708.

 If the memory device did not receive a WDM command, the memory device determines at 710 that is has received a WSM command. The memory device then performs a write of particular bytes of write data as indicates by the write data mask at 712.

Figure 8 is a diagram showing an alternative mapping of write data mask bits to data bytes for a WSM command according to an embodiment. Figure 8 is an example mapping between a specific address bit (in a specific burst location) to a specific data bit (in a specific burst location) which it will serve as a mask bit for the address bit.

Embodiments of the write data mask method and system disclosed herein include a method, comprising: transmitting a data mask from a first system component to a second system component, wherein the data mask indicates particular bytes of multiple bytes of data to be written in response to a write command, wherein the data mask is transmitted over address lines of an interface coupling the first system component and the second system component.

An embodiment comprises transmitting a write command from the first system component to the second system component, wherein the write command indicates one of a plurality of data mask types being transmitted, wherein the data mask types comprise a write_without_mask data mask type indicating that there is no data mask for the write command, a write_multiple-byte data mask type indicating that a mask bit of the data mask applies to a group of bytes of write data, and a write_single_byte data mask type indicating that one bit of the data mask applies to one byte of write data.

In an embodiment, each mask bit of the data mask may be configured to apply to any two bytes of write data as specified by a client component sending a request for a write operation to the first component.

In an embodiment, each mask bit of the data mask may be configured to apply to two any bytes of write data as configured by the first component.

An embodiment comprises the first component determining a data mask type of the plurality of data mask types based on a write request received from a client component.

In an embodiment, the first component and the second component communicate using a protocol that specifies transmission of 32 bytes per command, and wherein the write_without_mask data mask type indicates that 32 bytes of data are to be written in response to the write command.

In an embodiment, the first component and the second component communicate using a protocol that specifies transmission of 32 bytes per command, and wherein the write_multiple_byte data mask type indicates that each bit of a 16-bit

mask indicates whether two particular bytes of data are to be written in response to the write command.

In an embodiment, the first component and the second component communicate using a protocol that specifies transmission of 32 bytes per command, and wherein the write_single-byte data mask type indicates that each bit of a 16-bit mask indicates that a particular byte of data is to be written in response to the write command.

Embodiments of the write data mask method and system disclosed herein further include a method comprising: receiving a memory access request from a memory client; determining whether the request is a write request; if the request is a write request, determining a granularity of a received write data mask, wherein the received write data mask indicates a subset of the write data transmitted in the write request is to be written to a memory device, and wherein the granularity indicates a quantity of write data to which each bit of the received write data mask applies; selecting a write command to issue to the memory device based on the granularity of the received write data mask; and generating a write data mask to be transmitted to the memory device on the address lines.

An embodiment further comprises: if the request is a write request, determining whether the request includes a write data mask; and if the request does not include a write data mask, selecting a write command to issue to the memory device based on the determination, wherein the write command indicates that each byte of the write data is to be written to the memory device.

In an embodiment, the write command comprises a write_multiple_byte command that indicates which bits of the write data mask apply to particular groups of bytes of the write data.

In an embodiment, the write command comprises a write_single_byte command that indicates which bits of the write data mask apply to particular bytes of the write data.

In an embodiment, each bit of the generated write data mask is configured to map to a particular group of bytes of the write data according to the write_multiple_byte command.

In an embodiment, each bit of the generated write data mask is configured to map to a particular byte of the write data according to the write_single_byte command.

In an embodiment, each of the particular groups comprises two bytes.

In an embodiment, 32 bytes may be transmitted in response to a write command, and wherein two address bytes and two data mask bytes are transmitted in response to the write_multiple_byte command.

5 In an embodiment, 32 bytes may be transmitted in response to a write command, and wherein two address bytes and four data mask bytes are transmitted in response to the write_single_byte command, and two no-operation cycles are inserted on a command line to accommodate four additional mask bytes transmitted in response to the write_single_byte command.

10 In an embodiment, the write command comprises a write_without_mask command that is transmitted with a write address.

Embodiments of the write data mask method and system disclosed herein further include a method comprising: receiving one of a plurality of write commands on command lines of a memory interface; receiving address data on address lines of
15 the memory interface; determining whether a write data mask is received on the address lines; if a write data mask is received, determining a type of the write data mask; and based on the type, interpreting each bit of the data mask to indicate whether one or more bytes of write data are to be written to a memory device.

In an embodiment, determining whether a write data mask is received
20 comprises reading the received write command.

In an embodiment, determining the type of the write data mask comprises reading the received write command.

In an embodiment, the type of the write data mask comprises a write_multiple_byte type and a write_single_byte type.

25 An embodiment further comprises: in determining the type to be a write_multiple_byte type; and interpreting each bit of the write data mask to indicate whether a particular group of bytes of write data are to be written to the memory device.

An embodiment further comprises: in determining the type to be a
30 write_single_byte type; and interpreting each bit of the write data mask to indicate whether a particular byte of write data is to be written to the memory device.

Embodiments of the write data mask method and system disclosed herein further include a system comprising: a memory controller comprising data mask logic configured to, generate a write command from a received write request, wherein the

write command includes information regarding a write data mask type; generate a write data mask from the received write request; and transmit the write data mask to a memory device on address lines of a memory interface.

In an embodiment, the write command comprises: write_multiple_byte
5 command that indicates which bits of the write data mask apply to particular groups of bytes of write data; and a write_single_byte command that indicates which bits of the write data mask apply to particular bytes of the write data.

In an embodiment, the write command comprises a write_without_mask
10 command that indicates all bytes of write data are to be written to the memory device.

Embodiments of the write data mask method and system disclosed herein
15 further include a system comprising: a memory device comprising data mask logic configured to, receive a write command via a memory interface from a memory controller; determine a type of the write command; and based on the type of the write command, interpret a transmission on address lines of the memory interface as a write data mask.

In an embodiment, the type of the write command comprises:
write_multiple_byte command that indicates which bits of the write data mask apply to particular groups of bytes of write data; and a write_single_byte command that indicates which bits of the write data mask apply to particular bytes of the write data.

In an embodiment, the write command comprises a write_without_mask
20 command that indicates all bytes of write data are to be written to the memory device.

Embodiments of the write data mask method and system disclosed herein
25 further include a computer-readable medium storing instructions that, when executed in a system, cause a memory write method to be performed, the method comprising: transmitting a data mask from a first system component to a second system component, wherein the data mask indicates particular bytes of multiple bytes of data to be written in response to a write command, wherein the data mask is transmitted over address lines of an interface coupling the first system component and the second system component.

In an embodiment, the method further comprises transmitting a write
30 command from the first system component to the second system component, wherein the write command indicates one of a plurality of data mask types being transmitted, wherein the data mask types comprise a write_without_mask data mask type indicating that there is no data mask for the write command, a write_multiple_byte

data mask type indicating that a mask bit of the data mask applies to a group of bytes of write data, and a write_single_byte data mask type indicating that one bit of the data mask applies to one byte of write data.

In an embodiment, each mask bit of the data mask may be configured to apply
5 to any two bytes of write data as specified by a client component sending a request for a write operation to the first component.

In an embodiment, each mask bit of the data mask may be configured to apply to two any bytes of write data as configured by the first component.

An embodiment further comprises the first component determining a data
10 mask type of the plurality of data mask types based on a write request received from a client component.

Aspects of the embodiments described above may be implemented as functionality programmed into any of a variety of circuitry, including but not limited to programmable logic devices (PLDs), such as field programmable gate arrays
15 (FPGAs), programmable array logic (PAL) devices, electrically programmable logic and memory devices, and standard cell-based devices, as well as application specific integrated circuits (ASICs) and fully custom integrated circuits. Some other possibilities for implementing aspects of the embodiments include microcontrollers with memory (such as electronically erasable programmable read only memory
20 (EEPROM)), embedded microprocessors, firmware, software, etc. Furthermore, aspects of the embodiments may be embodied in microprocessors having software-based circuit emulation, discrete logic (sequential and combinatorial), custom devices, fuzzy (neural) logic, quantum devices, and hybrids of any of the above device types. Of course the underlying device technologies may be provided in a variety of
25 component types, e.g., metal-oxide semiconductor field-effect transistor (MOSFET) technologies such as complementary metal-oxide semiconductor (CMOS), bipolar technologies such as emitter-coupled logic (ECL), polymer technologies (e.g., silicon-conjugated polymer and metal-conjugated polymer-metal structures), mixed analog and digital, etc.

30 Unless the context clearly requires otherwise, throughout the description and the claims, the words “comprise,” “comprising,” and the like are to be construed in an inclusive sense as opposed to an exclusive or exhaustive sense; that is to say, in a sense of “including, but not limited to.” Words using the singular or plural number also include the plural or singular number, respectively. Additionally, the words

“herein,” “hereunder,” “above,” “below,” and words of similar import, when used in this application, refer to this application as a whole and not to any particular portions of this application. When the word “or” is used in reference to a list of two or more items, that word covers all of the following interpretations of the word, any of the
5 items in the list, all of the items in the list, and any combination of the items in the list.

The above description of illustrated embodiments of the system and method is not intended to be exhaustive or to limit the invention to the precise forms disclosed. While specific embodiments of, and examples for, the system and method are
10 described herein for illustrative purposes, various equivalent modifications are possible within the scope of the invention, as those skilled in the relevant art will recognize. The teachings of the disclosure provided herein can be applied to other systems, not only for systems including graphics processing or video processing, as described above. The various operations described may be performed in a very wide
15 variety of architectures and distributed differently than described. In addition, though many configurations are described herein, none are intended to be limiting or exclusive.

In other embodiments, some or all of the hardware and software capability described herein may exist in a printer, a camera, television, a digital versatile disc
20 (DVD) player, a handheld device, a mobile telephone or some other device. The elements and acts of the various embodiments described above can be combined to provide further embodiments. These and other changes can be made to the system and method in light of the above detailed description.

In general, in the following claims, the terms used should not be construed to
25 limit the system and method to the specific embodiments disclosed in the specification and the claims, but should be construed to include any processing systems and methods that operate under the claims. Accordingly, the system and method is not limited by the disclosure, but instead the scope of the method and system is to be determined entirely by the claims.

30 While certain aspects of the method and system are presented below in certain claim forms, the inventors contemplate the various aspects of the method and system in any number of claim forms. For example, while only one aspect of the system and method may be recited as embodied in computer-readable medium, other aspects may likewise be embodied in computer-readable medium. Accordingly, the inventors

reserve the right to add additional claims after filing the application to pursue such additional claim forms for other aspects of the system and method.

CLAIMS

What is claimed is:

1 1. A method, comprising:
2 transmitting a data mask from a first system component to a second system
3 component, wherein the data mask indicates particular bytes of multiple bytes of data
4 to be written in response to a write command, wherein the data mask is transmitted
5 over address lines of an interface coupling the first system component and the second
6 system component.

1 2. The method of claim 1, further comprising transmitting a write
2 command from the first system component to the second system component, wherein
3 the write command indicates one of a plurality of data mask types being transmitted,
4 wherein the data mask types comprise a write_without_mask data mask type
5 indicating that there is no data mask for the write command, a write_multiple-byte
6 data mask type indicating that a mask bit of the data mask applies to a group of bytes
7 of write data, and a write_single_byte data mask type indicating that one bit of the
8 data mask applies to one byte of write data.

1 3. The method of claim 2, wherein each mask bit of the data mask may be
2 configured to apply to any two bytes of write data as specified by a client component
3 sending a request for a write operation to the first component.

1 4. The method of claim 2, wherein each mask bit of the data mask may be
2 configured to apply to two any bytes of write data as configured by the first
3 component.

1 5. The method of claim 2, further comprising the first component
2 determining a data mask type of the plurality of data mask types based on a write
3 request received from a client component.

1 6. The method of claim 2, wherein the first component and the second
2 component communicate using a protocol that specifies transmission of 32 bytes per

3 command, and wherein the write_without_mask data mask type indicates that 32
4 bytes of data are to be written in response to the write command.

1 7. The method of claim 2, wherein the first component and the second
2 component communicate using a protocol that specifies transmission of 32 bytes per
3 command, and wherein the write_multiple_byte data mask type indicates that each bit
4 of a 16-bit mask indicates whether two particular bytes of data are to be written in
5 response to the write command.

1 8. The method of claim 2, wherein the first component and the second
2 component communicate using a protocol that specifies transmission of 32 bytes per
3 command, and wherein the write_single-byte data mask type indicates that each bit of
4 a 16-bit mask indicates that a particular byte of data is to be written in response to the
5 write command.

1 9. A method comprising:
2 receiving a memory access request from a memory client;
3 determining whether the request is a write request;
4 if the request is a write request, determining a granularity of a received write
5 data mask, wherein the received write data mask indicates a subset of the write data
6 transmitted in the write request is to be written to a memory device, and wherein the
7 granularity indicates a quantity of write data to which each bit of the received write
8 data mask applies;
9 selecting a write command to issue to the memory device based on the
10 granularity of the received write data mask; and
11 generating a write data mask to be transmitted to the memory device on the
12 address lines.

1 10. The method of claim 9, further comprising:
2 if the request is a write request, determining whether the request includes a
3 write data mask; and
4 if the request does not include a write data mask, selecting a write command to
5 issue to the memory device based on the determination, wherein the write command
6 indicates that each byte of the write data is to be written to the memory device.

1 11. The method of claim 9, wherein the write command comprises a
2 write_multiple_byte command that indicates which bits of the write data mask apply
3 to particular groups of bytes of the write data.

1 12. The method of claim 9, wherein the write command comprises a
2 write_single_byte command that indicates which bits of the write data mask apply to
3 particular bytes of the write data.

1 13. The method of claim 11, wherein each bit of the generated write data
2 mask is configured to map to a particular group of bytes of the write data according to
3 the write_multiple_byte command.

1 14. The method of claim 12, wherein each bit of the generated write data
2 mask is configured to map to a particular byte of the write data according to the
3 write_single_byte command.

1 15. The method of claim 11, wherein each of the particular groups
2 comprises two bytes.

1 16. The method of claim 11, wherein 32 bytes may be transmitted in
2 response to a write command, and wherein two address bytes and two data mask bytes
3 are transmitted in response to the write_multiple_byte command.

1 17. The method of claim 12, wherein 32 bytes may be transmitted in
2 response to a write command, and wherein two address bytes and four data mask
3 bytes are transmitted in response to the write_single_byte command, and two no-
4 operation cycles are inserted on a command line to accommodate four additional
5 mask bytes transmitted in response to the write_single_byte command.

1 18. The method of claim 10, wherein the write command comprises a
2 write_without_mask command that is transmitted with a write address.

1 19. A method comprising:

2 receiving one of a plurality of write commands on command lines of a
3 memory interface;
4 receiving address data on address lines of the memory interface;
5 determining whether a write data mask is received on the address lines;
6 if a write data mask is received, determining a type of the write data mask; and
7 based on the type, interpreting each bit of the data mask to indicate whether
8 one or more bytes of write data are to be written to a memory device.

1 20. The method of claim 19, wherein determining whether a write data
2 mask is received comprises reading the received write command.

1 21. The method of claim 19, wherein determining the type of the write
2 data mask comprises reading the received write command.

1 22. The method of claim 19, wherein the type of the write data mask
2 comprises a write_multiple_byte type and a write_single_byte type.

1 23. The method of claim 22, further comprising:
2 in determining the type to be a write_multiple_byte type; and
3 interpreting each bit of the write data mask to indicate whether a particular
4 group of bytes of write data are to be written to the memory device.

1 24. The method of claim 22, further comprising:
2 in determining the type to be a write_single_byte type; and
3 interpreting each bit of the write data mask to indicate whether a particular
4 byte of write data is to be written to the memory device.

1 25. A system comprising:
2 a memory controller comprising data mask logic configured to,
3 generate a write command from a received write request, wherein the
4 write command includes information regarding a write data mask type;
5 generate a write data mask from the received write request; and
6 transmit the write data mask to a memory device on address lines of a
7 memory interface.

1 26. The system of claim 25, wherein the write command comprises:
2 write_multiple_byte command that indicates which bits of the write data mask
3 apply to particular groups of bytes of write data; and
4 a write_single_byte command that indicates which bits of the write data mask
5 apply to particular bytes of the write data.

1 27. The system of claim 25, wherein the write command comprises a
2 write_without_mask command that indicates all bytes of write data are to be written
3 to the memory device.

1 28. A system comprising:
2 a memory device comprising data mask logic configured to,
3 receive a write command via a memory interface from a memory
4 controller;
5 determine a type of the write command; and
6 based on the type of the write command, interpret a transmission on
7 address lines of the memory interface as a write data mask.

1 29. The system of claim 28, wherein the type of the write command
2 comprises:
3 write_multiple_byte command that indicates which bits of the write data mask
4 apply to particular groups of bytes of write data; and
5 a write_single_byte command that indicates which bits of the write data mask
6 apply to particular bytes of the write data.

1 30. The system of claim 28, wherein the write command comprises a
2 write_without_mask command that indicates all bytes of write data are to be written
3 to the memory device.

1 31. A computer-readable medium storing instructions that, when executed
2 in a system, cause a memory write method to be performed, the method comprising:
3 transmitting a data mask from a first system component to a second system
4 component, wherein the data mask indicates particular bytes of multiple bytes of data

5 to be written in response to a write command, wherein the data mask is transmitted
6 over address lines of an interface coupling the first system component and the second
7 system component.

1 32. The computer-readable medium of claim 31, the method further
2 comprising transmitting a write command from the first system component to the
3 second system component, wherein the write command indicates one of a plurality of
4 data mask types being transmitted, wherein the data mask types comprise a
5 write_without_mask data mask type indicating that there is no data mask for the write
6 command, a write_multiple_byte data mask type indicating that a mask bit of the data
7 mask applies to a group of bytes of write data, and a write_single_byte data mask type
8 indicating that one bit of the data mask applies to one byte of write data.

1 33. The computer-readable medium of claim 32, wherein each mask bit of
2 the data mask may be configured to apply to any two bytes of write data as specified
3 by a client component sending a request for a write operation to the first component.

1 34. The computer-readable medium of claim 32, wherein each mask bit of
2 the data mask may be configured to apply to two any bytes of write data as configured
3 by the first component.

1 35. The computer-readable medium of claim 32, further comprising the
2 first component determining a data mask type of the plurality of data mask types
3 based on a write request received from a client component.

100

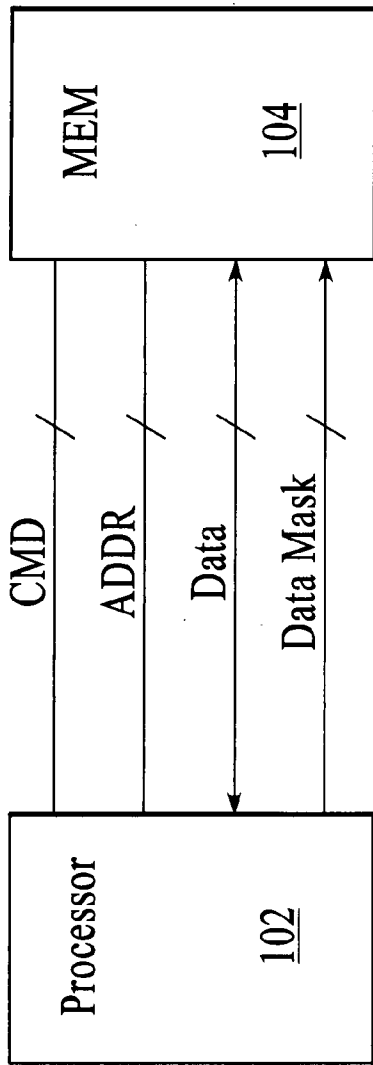


FIG.1A
(PRIOR ART)

burst length 8, 3 writes, middle with 10110111 data mask
current method: dedicated data mask pin

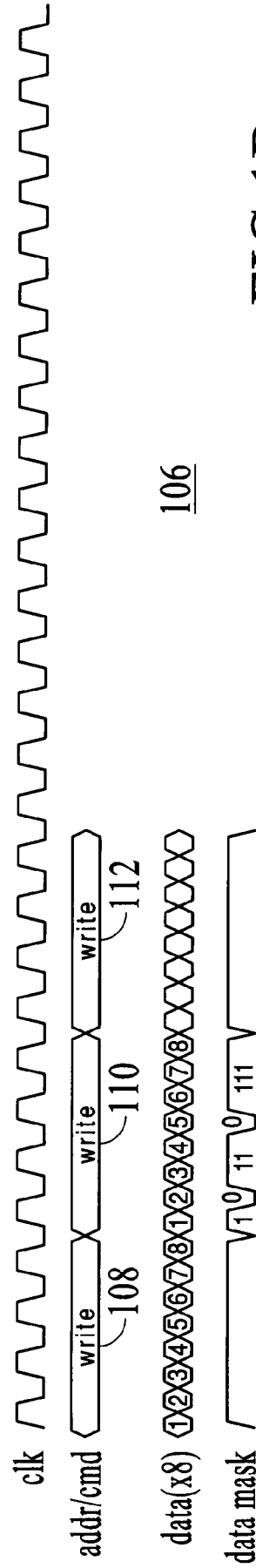


FIG.1B
(PRIOR ART)

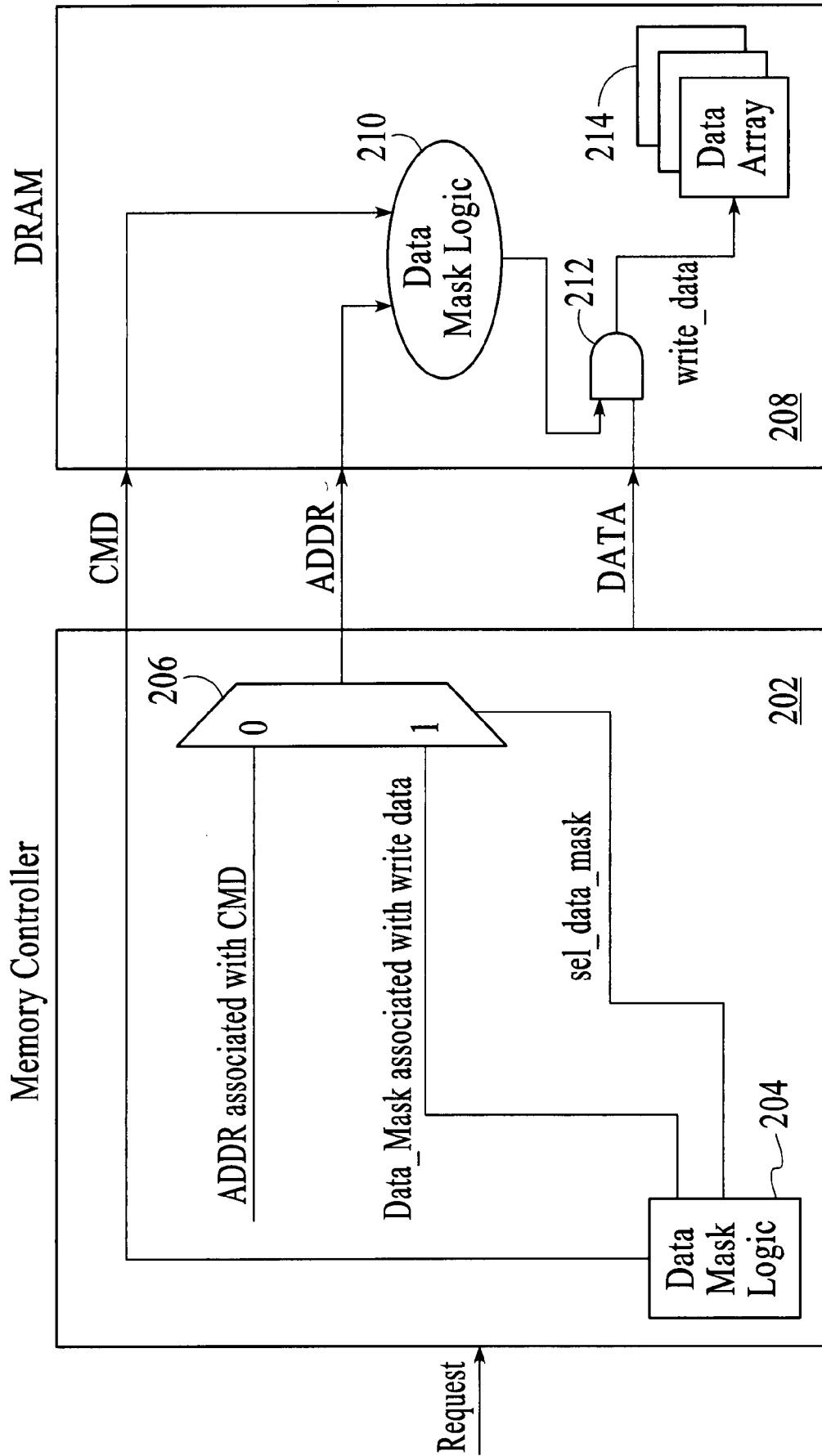


FIG.2

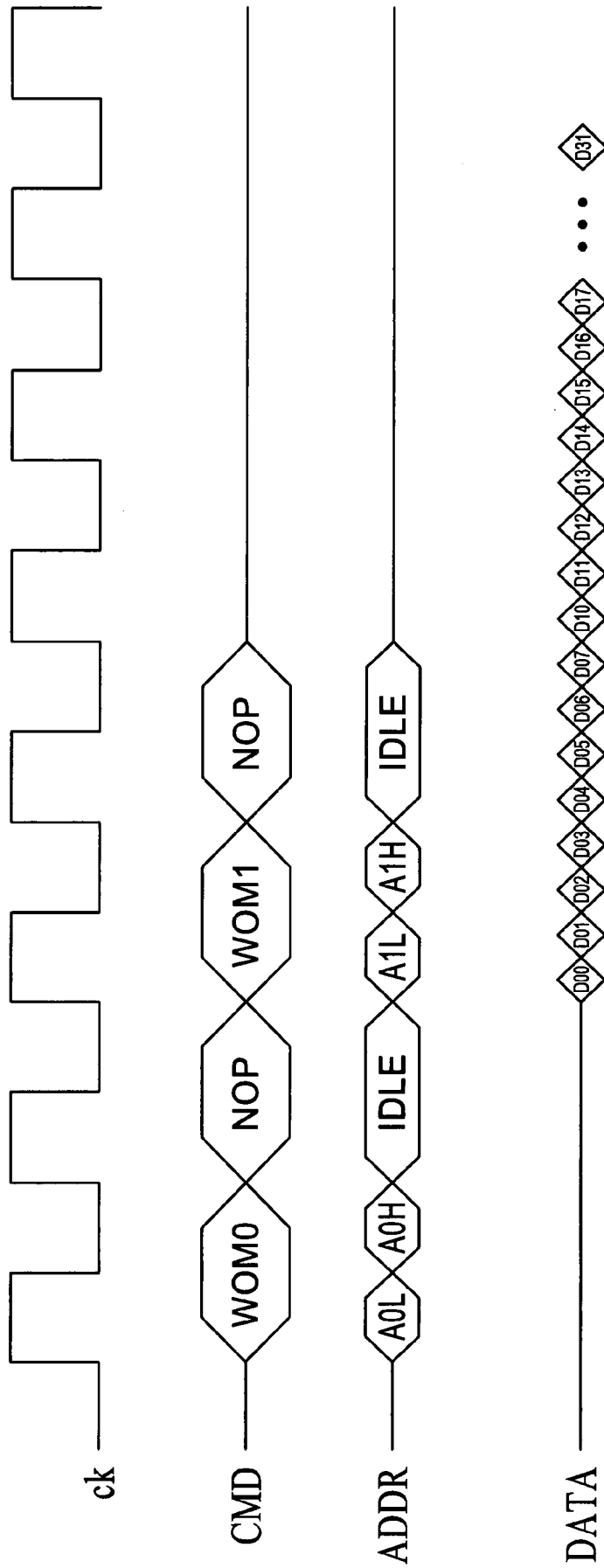


FIG.3

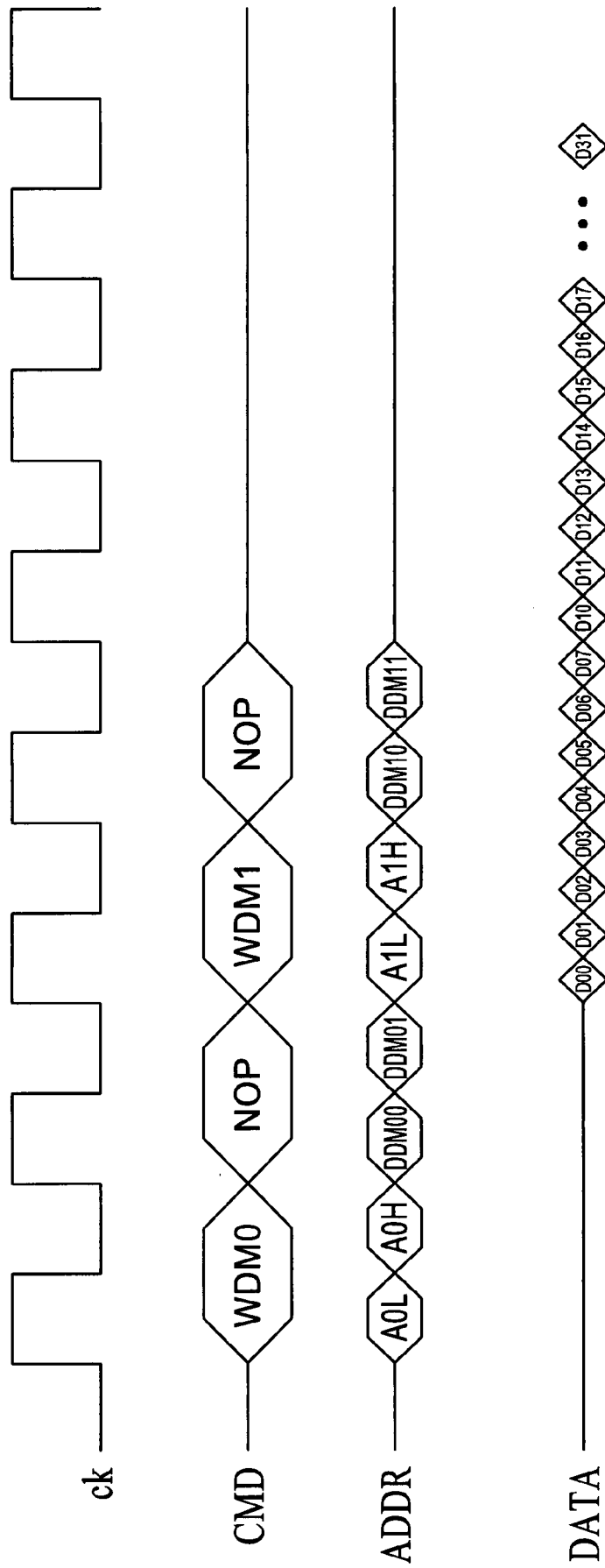


FIG.4

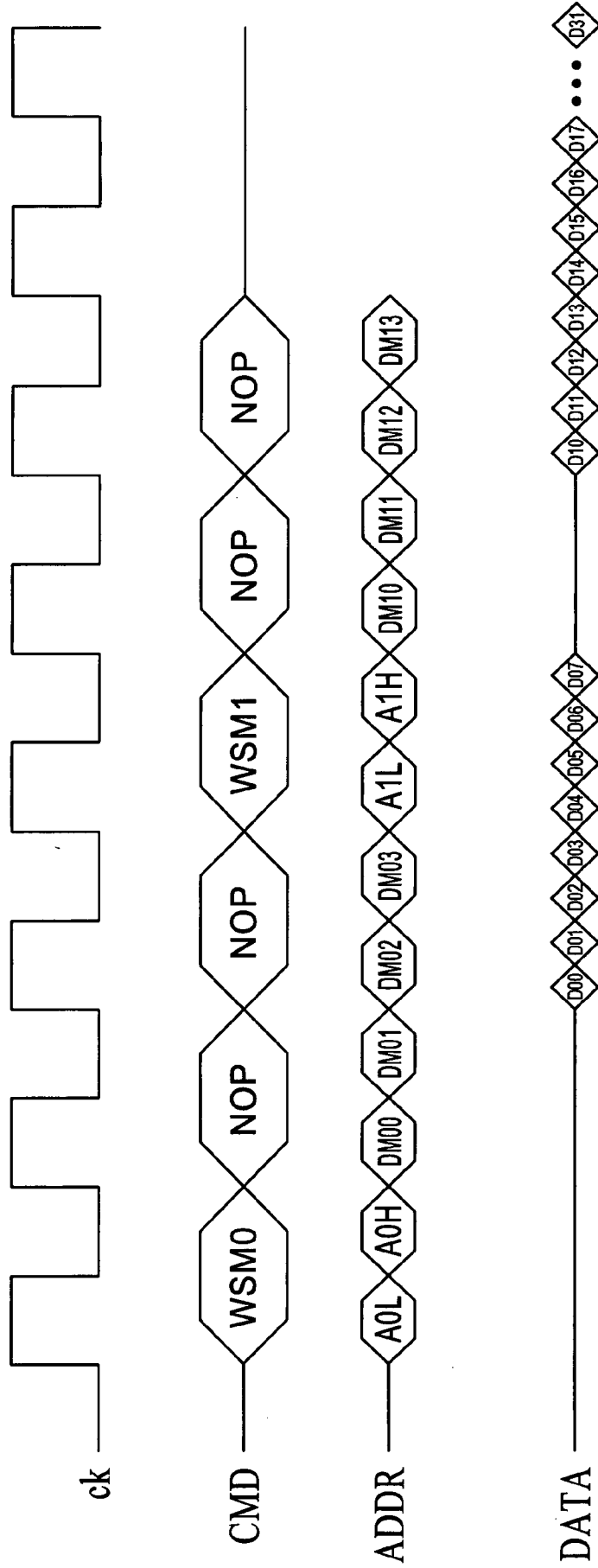
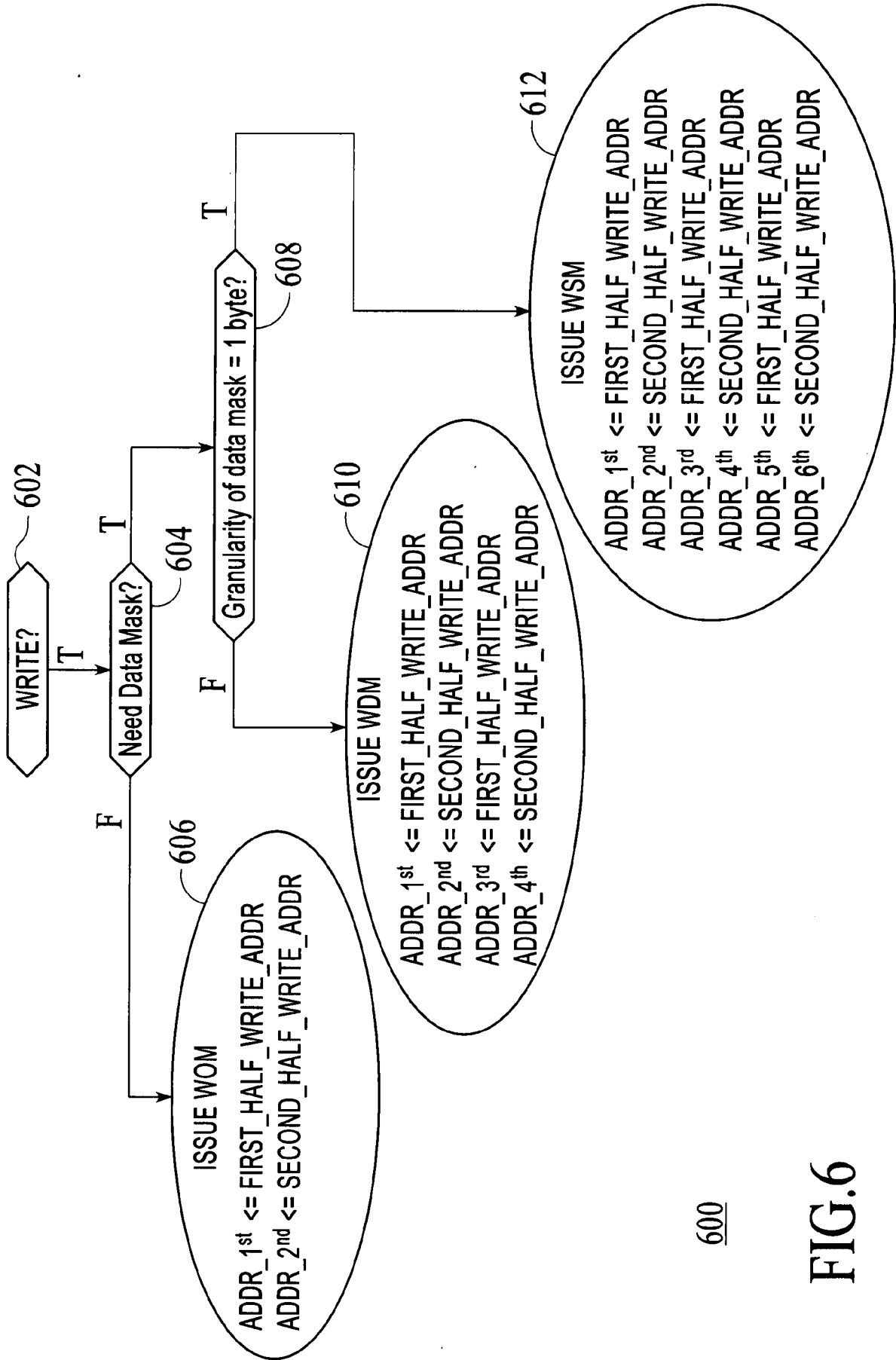


FIG.5



600

FIG.6

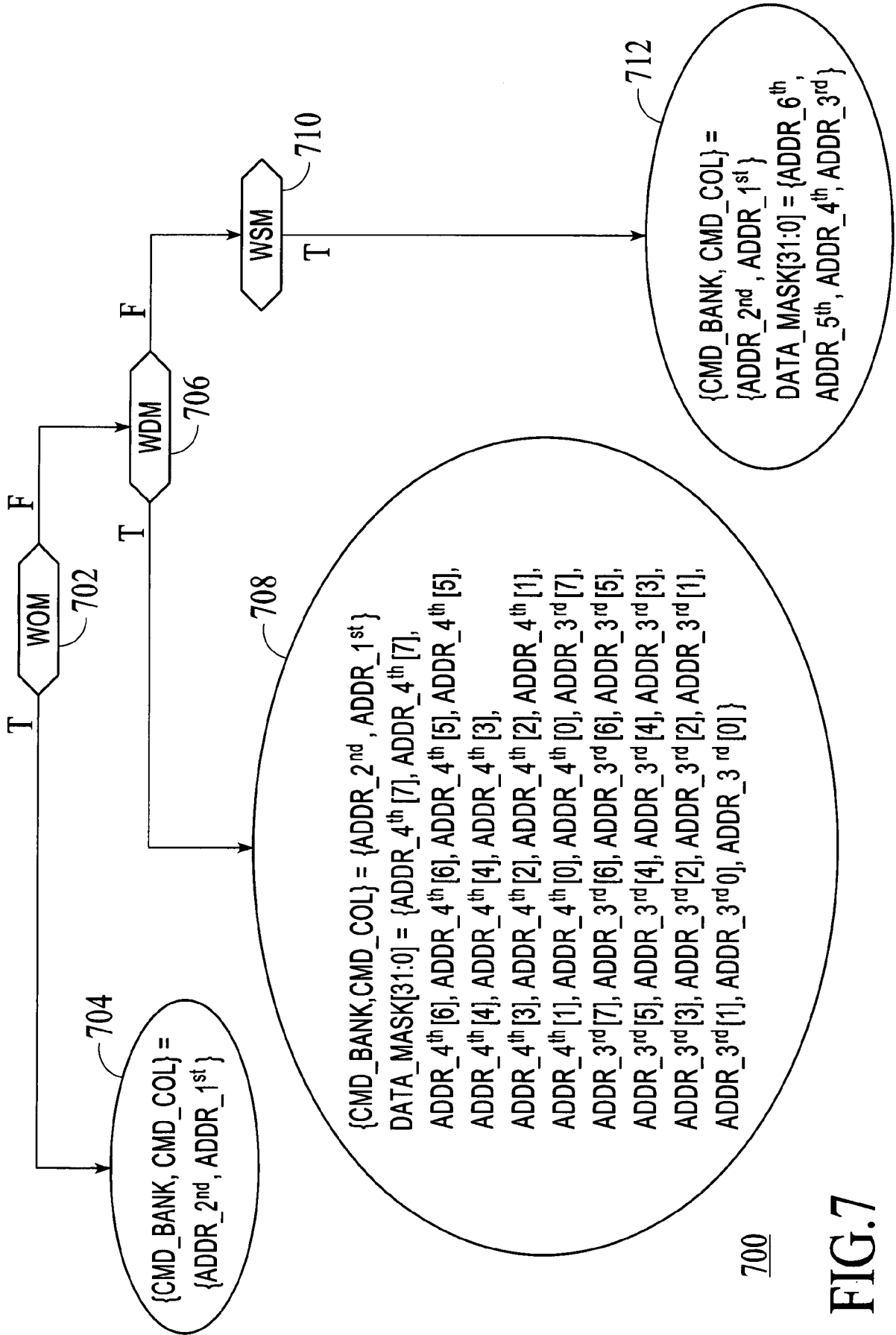


FIG. 7

