

Electronic Acknowledgement Receipt

EFS ID:	3500428
Application Number:	12144396
International Application Number:	
Confirmation Number:	1445
Title of Invention:	Memory Subsystem And Method
First Named Inventor/Applicant Name:	David T. Wang
Customer Number:	68399
Filer:	John Clarence Carey/Kimberly Keeler
Filer Authorized By:	John Clarence Carey
Attorney Docket Number:	META/0006
Receipt Date:	23-JUN-2008
Filing Date:	
Time Stamp:	17:46:20
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Credit Card
Payment was successfully received in RAM	\$ 1130
RAM confirmation Number	4302
Deposit Account	200782
Authorized User	CAREY,JOHN

The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:

Charge any Additional Fees required under 37 C.F.R. Section 1.16 (National application filing, search, and examination fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.21 (Miscellaneous fees and charges)

Samsung Electronics Co., Ltd.

File Listing:					
Document Number	Document Description	File Name	File Size(Bytes) /Message Digest	Multi Part /.zip	Pages (if appl.)
1		META_0006_EAPP.pdf	1735791 a78ad13719fb01dba3df5802079544588b7df028	yes	47
	Multipart Description/PDF files in .zip description				
	Document Description		Start		End
	Specification		1		22
	Claims		23		27
	Abstract		28		28
	Drawings-only black and white line drawings		29		41
	Oath or Declaration filed		42		45
	Authorization to access Appl. by Trilateral Office		46		46
	Nonpublication request from applicant.		47		47
Warnings:					
Information:					
2	Fee Worksheet (PTO-06)	fee-info.pdf	8461 613c4e01279fddb6ba892e7477934a80a97c78f	no	2
Warnings:					
Information:					
Total Files Size (in bytes):			1744252		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

UNITED STATES PATENT APPLICATION FOR:

MEMORY SUBSYSTEM AND METHOD

INVENTORS:

DAVID T. WANG

SURESH NATARAJAN RAJAN

ATTORNEY DOCKET NUMBER: META/0006

PATTERSON & SHERIDAN, L.L.P.
3040 Post Oak Blvd. Suite 1500
Houston, TX 77056

MEMORY SUBSYSTEM AND METHOD

BACKGROUND OF THE INVENTION

Field of the Invention

[0001] Embodiments of the present invention generally relate to memory subsystems and, more specifically, to improvements to such memory subsystems.

Description of the Related Art

[0002] Memory circuit speeds remain relatively constant, but the required data transfer speeds and bandwidth of memory systems are increasing, currently doubling every three years. The result is that more commands must be scheduled, issued and pipelined in a memory system to increase bandwidth. However, command scheduling constraints that exist in the memory systems limit the command issue rates, and consequently, limit the increase in bandwidth.

[0003] In general, there are two classes of command scheduling constraints that limit command scheduling and command issue rates in memory systems: inter-device command scheduling constraints, and intra-device command scheduling constraints. These command scheduling constraints and other timing constraints and timing parameters are defined by manufacturers in their memory device data sheets and by standards organizations such as JEDEC.

[0004] Examples of inter-device (between devices) command scheduling constraints include rank-to-rank data bus turnaround times, and on-die-termination (ODT) control switching times. The inter-device command scheduling constraints typically arise because the devices share a resource (for example a data bus) in the memory sub-system.

[0005] Examples of intra-device (inside devices) command-scheduling constraints include column-to-column delay time (tCCD), row-to-row activation delay time (tRRD), four-bank activation window time (tFAW), and write-to-read turn-around time (tWTR). The intra-device command-scheduling constraints typically arise

because parts of the memory device (e.g. column, row, bank, etc.) share a resource inside the memory device.

[0006] In implementations involving more than one memory device, some technique must be employed to assemble the various contributions from each memory device into a word or command or protocol as may be processed by the memory controller. Various conventional implementations, in particular designs within the classification of Fully Buffered DIMMs (FBDIMMs, a type of industry standard memory module) are designed to be capable of such assembly. However, there are several problems associated with such an approach. One problem is that the FBDIMM approach introduces significant latency (see description, below). Another problem is that the FBDIMM approach requires a specialized memory controller capable of processing the assembly.

[0007] As memory speed increases, the introduction of latency becomes more and more of a detriment to the operation of the memory system. Even modern FBDIMM-type memory systems introduce 10s of nanoseconds of delay as the packet is assembled. As will be shown in the disclosure to follow, the latency introduced need not be so severe.

[0008] Moreover, the implementation of the FBDIMM-type memory devices required corresponding changes in the behavior of the memory controller, and this FBDIMMS are not backward compatible among industry-standard memory system. As will be shown in the disclosure to follow, various embodiments of the present invention may be used with previously existing memory controllers, without modification to their logic or interfacing requirements.

[0009] In order to appreciate the extent of the introduction of latency in an FBDIMM-type memory system, one needs to refer to Figure 1. Figure 1 shows an FBDIMM-type memory system 100 wherein multiple DRAMS (D0, D1,...D7, D8) are in communication via a daisy-chained interconnect. The buffer 105 is situated between two memory circuits (e.g. D1 and D2). In the READ path, the buffer 105 is

capable to present to memory D_N the data retrieved from D_M ($M > N$). Of course in a conventional FBDIMM-type system, the READ data from each successively higher memory D_M must be merged with the data of memory D_N , and such function is implemented via pass-through and merging logic 106. As can be seen, such an operation occurs sequentially at each buffer 105, and latency is thus cumulatively introduced.

[0010] As the foregoing illustrates, what is needed in the art is a memory subsystem and method that overcome the shortcomings of prior art systems.

SUMMARY OF THE INVENTION

[0011] One embodiment of the present invention sets forth an interface circuit configured to combine a plurality of data bursts returned by a plurality of memory devices into a contiguous data burst. The interface circuit includes a system control signal interface adapted to receive a first command from a memory controller and emulation and command translation logic adapted to translate a first address associated with the first command, issue the first command to a first memory device within the plurality of memory devices corresponding to the first address, and determine that the first command is a read command. The emulation and command translation logic is further adapted to select a memory data signal interface corresponding to the first memory device, receive a first data burst from the first memory device, delay the first data burst to eliminate a first clock-to-data phase between the first memory device and the interface circuit, and re-drive the first data burst to the memory controller.

[0012] One advantage of the disclosed interface circuit is that it can provide higher memory performance by not requiring idle bus cycles to turnaround the data bus when switching from reading from one memory device to reading from another memory device, or from writing to one memory device to writing to another memory device.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] So that the manner in which the above recited features of the present invention can be understood in detail, a more particular description of the invention, briefly summarized above, may be had by reference to embodiments, some of which are illustrated in the appended drawings. It is to be noted, however, that the appended drawings illustrate only typical embodiments of this invention and are therefore not to be considered limiting of its scope, for the invention may admit to other equally effective embodiments.

[0014] Figure 1 illustrates an FBDIMM-type memory system, according to prior art;

[0015] Figure 2A illustrates major logical components of a computer platform, according to prior art;

[0016] Figure 2B illustrates major logical components of a computer platform, according to one embodiment of the present invention;

[0017] Figure 2C illustrates a hierarchical view of the major logical components of a computer platform shown in Figure 2B, according to one embodiment of the present invention;

[0018] Figure 3A illustrates a timing diagram for multiple memory devices in a low data rate memory system, according to prior art;

[0019] Figure 3B illustrates a timing diagram for multiple memory devices in a higher data rate memory system, according to prior art;

[0020] Figure 3C illustrates a timing diagram for multiple memory devices in a high data rate memory system, according to prior art;

[0021] Figure 4A illustrates a data flow diagram showing how time separated bursts are combined into a larger contiguous burst, according to one embodiment of

the present invention;

[0022] Figure 4B illustrates a waveform corresponding to Figure 4A showing how time separated bursts are combined into a larger contiguous burst, according to one embodiment of the present invention;

[0023] Figure 4C illustrates a flow diagram of method steps showing how the interface circuit can optionally make use of a training or clock-to-data phase calibration sequence to independently track the clock-to-data phase relationship between the memory components and the interface circuit, according to one embodiment of the present invention;

[0024] Figure 4D illustrates a flow diagram showing the operations of the interface circuit in response to the various commands, according to one embodiment of the present invention;

[0025] Figures 5A through 5F illustrates a computer platform that includes at least one processing element and at least one memory module, according to various embodiments of the present invention.

DETAILED DESCRIPTION

[0026] Figure 2A illustrates major logical components of a computer platform 200, according to prior art. As shown, the computer platform 200 includes a system 220 and an array of memory components 210 interconnected via a parallel interface bus 240. As also shown, the system 220 further includes a memory controller 225.

[0027] Figure 2B illustrates major logical components of a computer platform 201, according to one embodiment of the present invention. As shown, the computer platform 201 includes the system 220 (e.g., a processing unit) that further includes the memory controller 225. The computer platform 201 also includes an array of memory components 210 interconnected to an interface circuit 250, which is connected to the system 220 via the parallel interface bus 240. In various embodiments, the memory components 210 may include logical or physical

components. In one embodiment, the memory components 210 may include DRAM devices. In such a case, commands from the memory controller 225 that are directed to the DRAM devices respect all of the command-scheduling constraints (e.g. tRRD, tCCD, tFAW, tWTR, etc.). In the embodiment of Figure 2B, none of the memory components 210 is in direct communication with the memory controller 225. Instead, all communication to/from the memory controller 225 and the memory components 210 is carried out through the interface circuit 250. In other embodiments, only some of the communication to/from the memory controller 225 and the memory components 210 is carried out through the interface circuit 250.

[0028] Figure 2C illustrates a hierarchical view of the major logical components of the computer platform 201 shown in Figure 2B, according to one embodiment of the present invention. Figure 2C depicts the computer platform 201 being comprised of wholly separate components, namely the system 220 (e.g. a motherboard), and the memory components 210 (e.g. logical or physical memory circuits).

[0029] In the embodiment shown, the system 220 further comprises a memory interface 221, logic for retrieval and storage of external memory attribute expectations 222, memory interaction attributes 223, a data processing engine 224 (e.g., a CPU), and various mechanisms to facilitate a user interface 225. In various embodiments, the system 220 is designed to the specifics of various standards, in particular the standard defining the interfaces to JEDEC-compliant semiconductor memory (e.g. DRAM, SDRAM, DDR2, DDR3, etc.). The specifics of these standards address physical interconnection and logical capabilities. In different embodiments, the system 220 may include a system BIOS program capable of interrogating the memory components 210 (e.g. DIMMs) as a way to retrieve and store memory attributes. Further, various external memory embodiments, including JEDEC-compliant DIMMs, include an EEPROM device known as a serial presence detect (SPD) where the DIMM's memory attributes are stored. It is through the interaction of the BIOS with the SPD and the interaction of the BIOS with the physical memory circuits' physical attributes that the memory attribute expectations and memory

interaction attributes become known to the system 220.

[0030] As also shown, the computer platform 201 includes one or more interface circuits 250 electrically disposed between the system 220 and the memory components 210. The interface circuit 250 further includes several system-facing interfaces, for example, a system address signal interface 271, a system control signal interface 272, a system clock signal interface 273, and a system data signal interface 274. Similarly, the interface circuit 250 includes several memory-facing interfaces, for example, a memory address signal interface 275, a memory control signal interface 276, a memory clock signal interface 277, and a memory data signal interface 278.

[0031] In Figure 2C, the memory data signal interface 278 is specifically illustrated as separate, independent interface. This illustration is specifically designed to demonstrate the functional operation of the seamless burst merging capability of the interface circuit 250, and should not be construed as a limitation on the implementation of the interface circuit. In other embodiments, the memory data signal interface 278 may be composed of more than one independent interfaces. Furthermore, specific implementations of the interface circuit 250 may have a memory address signal interface 275 that is similarly composed of more than one independently operable memory address signal interfaces, and multiple, independent interfaces may exist for each of the signal interfaces included within the interface circuit 250.

[0032] An additional characteristic of the interface circuit 250 is the presence of emulation and command translation logic 280, data path logic 281, and initialization and configuration logic 282. The emulation and command translation logic 280 is configured to receive and, optionally, store electrical signals (e.g. logic levels, commands, signals, protocol sequences, communications) from or through the system-facing interfaces, and process those signals. In various embodiments, the emulation and command translation logic 280 may respond to signals from the system-facing interfaces by responding back to the system 220 by presenting

signals to the system 220, process those signals with other information previously stored, present signals to the memory components 210, or perform any of the aforementioned operations in any order.

[0033] The emulation and command translation logic 280 is capable of adopting a personality, and such personality defines the physical memory component attributes. In various embodiments of the emulation and command translation logic 280, the personality can be set via any combination of bonding options, strapping, programmable strapping, the wiring between the interface circuit 250 and the memory components 210, and actual physical attributes (e.g. value of mode register, value of extended mode register) of the physical memory connected to the interface circuit 250 as determined at some moment when the interface circuit 250 and memory components 210 are powered up.

[0034] The data path logic 281 is configured to receive internally generated control and command signals from the emulation and command translation logic 280, and use the signals to direct the flow of data through the interface circuit 250. The data path logic 281 may alter the burst length, burst ordering, data-to-clock phase-relationship, or other attributes of data movement through the interface circuit 250.

[0035] The initialization and configuration logic 282 is capable of using internally stored initialization and configuration logic to optionally configure all other logic blocks and signal interfaces in the interface circuit 250. In one embodiment, the emulation and command translation logic 280 is able to receive configuration request from the system control signal interface 272, and configure the emulation and command translation logic 280 to adopt different personalities.

[0036] More illustrative information will now be set forth regarding various optional architectures and features of different embodiments with which the foregoing frameworks may or may not be implemented, per the desires of the user. It should be noted that the following information is set forth for illustrative purposes

and should not be construed as limiting in any manner. Any of the following features may be optionally incorporated with or without the other features described.

Industry-Standard Operation

[0037] In order to discuss specific techniques for inter- and intra-device delays, some discussion of access commands and how they are used is foundational.

[0038] Typically, access commands directed to industry-standard memory systems such as DDR2 and DDR3 SDRAM memory systems may be required to respect command-scheduling constraints that limit the available memory bandwidth. Note: the use of DDR2 and DDR3 in this discussion is purely illustrative examples, and is not to be construed as limiting in scope.

[0039] In modern DRAM devices, the memory storage cells are arranged into multiple banks, each bank having multiple rows, and each row having multiple columns. The memory storage capacity of the DRAM device is equal to the number of banks times the number of rows per bank times the number of column per row times the number of storage bits per column. In industry-standard DRAM devices (e.g. SDRAM, DDR, DDR2, DDR3, and DDR4 SDRAM, GDDR2, GDDR3 and GDDR4 SGRAM, etc.), the number of banks per device, the number of rows per bank, the number of columns per row, and the column sizes are determined by a standards-setting organization such as JEDEC. For example, the JEDEC standards require that a 1 Gb DDR2 or DDR3 SDRAM device with a four-bit wide data bus have eight banks per device, 8192 rows per bank, 2048 columns per row, and four bits per column. Similarly, a 2 Gb device with a four-bit wide data bus must have eight banks per device, 16384 rows per bank, 2048 columns per row, and four bits per column. A 4 Gb device with four-bit wide data bus must have eight banks per device, 32768 rows per bank, 2048 columns per row, and four bits per column. In the 1 Gb, 2Gb and 4Gb devices, the row size is constant, and the number of rows doubles with each doubling of device capacity. Thus, a 2 Gb or a 4 Gb device may be emulated by using multiple 1 Gb and 2 Gb devices, and by directly translating

row-activation commands to row-activation commands and column-access commands to column-access commands. This emulation is possible because the 1 Gb, 2 Gb, and 4 Gb devices all have the same row size.

[0040] The JEDEC standards require that an 8 Gb device with a four-bit wide data bus interface must have eight banks per device, 32768 rows per bank, 4096 columns per row, and four bits per column – thus doubling the row size of the 4 Gb device. Consequently, an 8 Gb device cannot necessarily be emulated by using multiple 1 Gb, 2 Gb or 4 Gb devices and simply translating row-activation commands to row-activation commands and column-access commands to column-access commands.

[0041] Now, with an understanding of how access commands are used, presented as follows are various additional optional techniques that may optionally be employed in different embodiments to address various possible issues.

[0042] Figure 3A illustrates a timing diagram for multiple memory devices (e.g., SDRAM devices) in a low data rate memory system, according to prior art. Figure 3A illustrates that multiple SDRAM devices in a low data rate memory system can share the data bus without needing idle cycles between data bursts. That is, in a low data rate system, the inter-device delays involved are small relative to a clock cycle. Therefore, multiple devices may share the same bus and even though there may be some timing uncertainty when one device stops being the bus master and another device becomes the bus master, the data cycle is not delayed or corrupted. This scheme using time division access to the bus has been shown to work for time multiplexed bus masters in a low data rate memory systems – without the requirement to include idle cycles to switch between the different bus masters.

[0043] As the speed of the clock increases, the inter- and intra-device delays comprise successively more and more of a clock cycle (as a ratio). At some point, the inter- and intra-device delays are sufficiently large (relative to a clock cycle) that the multiple devices on a shared bus must be managed. In particular, and as shown

in Figure 3B, as the speed of the clock increases, the inter- and intra-device delays comprise successively more and more of a clock cycle (as a ratio). Consequently, a one cycle delay is needed between the end of a read data burst of a first device on a shared device and the beginning of a read data burst of a second device on the same bus. Figure 3B illustrates that, at the clock rate shown, multiple memory devices (e.g., DDR SDRAM, DDR2 SDRAM, DDR3 SDRAM devices) sharing the data bus must necessarily incur minimally a one cycle penalty when switching from one memory device driving the data bus to another memory device driving the data bus.

[0044] Figure 3C illustrates a timing diagram for multiple memory devices in a high data rate memory system, according to prior art. Figure 3C shows command cycles, timing constraints 310 and 320, and idle cycles of memory. As the clock rate further increases, the inter- and intra-device delay may become as long as one or more clock cycles. In such a case, switching between a first memory device and a second memory device would introduce one or more idle cycles 330. Embodiments of the invention herein might be advantageously applied to reduce or eliminate idle time 330 between the data transfers 328 and 329.

[0045] Continuing the discussion of Figure 3C, the timing diagram shows a limitation preventing full bandwidth utilization in a DDR3 SDRAM memory system. For example, in an embodiment involving DDR3 SDRAM memory systems, any two row-access commands directed to a single DRAM device may not necessarily be scheduled closer than a period of time defined by the timing parameter of t_{RRD} . As another example, at most four row-access commands may be scheduled within a period of time defined by the timing parameter of t_{FAW} to a single DRAM device. Moreover, consecutive column-read access commands and consecutive column-write access commands cannot necessarily be scheduled to a given DRAM device any closer than t_{CCD} , where t_{CCD} equals four cycles (eight half-cycles of data) in DDR3 DRAM devices. This situation is shown in the left portion of the timing diagram of Figure 3C at 305. Row-access or row-activation commands are shown

as ACT in the figures. Column-access commands are shown as READ or WRITE in the figures. Thus, for example, in memory systems that require a data access in a data burst of four half-cycles as shown in Figure 3C, the tCCD constraint prevents column accesses from being scheduled consecutively. Figure 3C shows that the constraints 310 and 320 imposed on the DRAM commands sent to a given device restrict the command rate, resulting in idle cycles or bubbles 330 on the data bus and reducing the bandwidth. Again, embodiments of the invention herein might be advantageously applied to reduce or eliminate idle time 330 between the data transfers 328 and 329.

[0046] As illustrated in Figures 3A-3C, idle-cycle-less data bus switching was possible with slower speed DRAM memory systems such as SDRAM memory systems, but not possible with higher speed DRAM memory systems such as DDR SDRAM, DDR2 SDRAM, DDR3 SDRAM devices due to the fact that in any memory system where multiple memory devices share the same data bus, the skew and jitter characteristics of address, clock, and data signals introduce timing uncertainties into the access protocol of the memory system. In the case when the memory controller wishes to stop accessing one memory device to switch to accessing a different device, the differences in address, clock and data signal skew and jitter characteristics of the two difference memory devices reduce the amount of time that the memory controller can use to reliably capture data. In the case of the slow-speed SDRAM memory system, the SDRAM memory system is designed to operate at speeds no higher than 200 MHz, and data bus cycle times are longer than 5 nanoseconds (ns). Consequently, timing uncertainties introduced by inter-device skew and jitter characteristics may be tolerated as long as they are sufficiently smaller than the cycle time of the memory system – for example, 1 ns. However, in the case of higher speed memory systems, where data bus cycles times are comparable in duration to, or shorter than, one-nanosecond, a one-nanosecond uncertainty in skew or jitter between signal timing from different devices means that memory controllers can no longer reliably capture data from different devices without

accounting for the inter-device skew and jitter characteristics.

[0047] As illustrated in Figure 3B, DDR SDRAM, DDR2 and DDR3 SDRAM memory systems use the DQS signal to provide a source-synchronous timing reference between the DRAM devices and the memory controller. The use of the DQS signal provides accurate timing control at the cost of idle cycles that must be incurred when a first bus master (DRAM device) stops driving the DQS signal, and a second bus master (DRAM device) starts to drive the DQS signal for at least one cycle before the second bus master places the data burst on the shared data bus. The placement of multiple DRAM devices on the same shared data bus is a desirable configuration from the perspective of enabling a higher capacity memory system and providing a higher degree of parallelism to the memory controller. However, the required use of the DQS signal significantly lowers the sustainable bandwidth of the memory system.

[0048] The advantage of the infrastructure-compatible burst merging interface circuit 250 illustrated in Figures 2B and 2C and described in greater detail below is that it can provide the higher capacity, higher parallelism that the memory controller desires while retaining the use of the DQS signal in an infrastructure-compatible system to provide the accurate timing reference for data transmission that is critical for modern memory systems, without the cost of the idle cycles required for the multiple bus masters (DRAM devices) to switch from one DRAM device to another.

Elimination of Idle Data-Bus Cycles Using an Interface Circuit

[0049] Figure 4A illustrates a data flow diagram through the data signal interfaces 278, Data Path Logic 281 and System Data Signal Interface 274 of Figure 2C, showing how data bursts returned by multiple memory devices in response to multiple, independent read commands to different memory devices connected respectively to Data Path A, synchronized by Data Strobe A, Data Path B, synchronized by Data Strobe B, and Data Path C, synchronized by Data Strobe C are combined into a larger contiguous burst, according to one embodiment of the

present invention. In particular, data burst B (B0, B1, B2, B3) 4A20 is slightly overlapping with data burst A (A0, A1, A2, A3) 4A10. Also, data burst C 4A30 does not overlap with either the data burst A 4A10, nor the data burst B 4A20. As described in greater detail in Figures 4C and 4D, various logic components of the interface circuit 250 illustrated in Figure 2C are configured to re-time overlapping or non-overlapping bursts to obtain contiguous burst of data 4A40. In various embodiments, the logic required to implement the ordering and concatenation of overlapping or non-overlapping bursts may be implemented using registers, multiplexors, and combinational logic. As shown in Figure 4A, the assembled, contiguous burst of data 4A40 is indeed contiguous and properly ordered.

[0050] Figure 4A shows that the data returned by the memory devices can have different phase relationships relative to the clock signal of the interface circuit 250. Figure 4D shows how the interface circuit 250 may use the knowledge of the independent clock-to-data phase relationships to delay each data burst to the interface circuit 250 to the same clock domain, and re-drive the data bursts to the system interface as one single, contiguous, burst.

[0051] Figure 4B illustrates a waveform corresponding to Figure 4A showing how the three time separated bursts from three different memory devices are combined into a larger contiguous burst, according to one embodiment of the present invention. Figure 4B shows that, as viewed from the perspective of the interface circuit 250, the data burst A0-A1-A2-A3, arriving from one of the memory components 210 to memory data signal interface A as a response to command (Cmd) A issued by the memory controller 225, can have a data-to-clock relationship that is different from data burst B0-B1-B2-B3, arriving at memory signal interface B, and a data burst C0-C1-C2-C3 can have yet a third clock-to-data timing relationship with respect to the clock signal of the interface circuit 250. Figure 4B shows that once the respective data bursts are re-synchronized to the clocking domain of the interface circuit 250, the different data bursts can be driven out of the system data interface Z as a contiguous data burst.

[0052] Figure 4C illustrates a flow diagram of method steps showing how the interface circuit 250 can optionally make use of a training or clock-to-data phase calibration sequence to independently track the clock-to-data phase relationship between the memory components 210 and the interface circuit 250, according to one embodiment of the present invention. In implementations where the clock-to-data phase relationships are static, the training or calibration sequence is not needed to set the respective delays in the memory data signal interfaces. While the method steps are described with relation to the computer platform 201 illustrated in Figures 2B and 2C, any system performing the method steps, in any order, is within the scope of the present invention.

[0053] The training or calibration sequence is typically performed after the initialization and configuration logic 282 receives either an interface circuit initialization or calibration request. The goal of the training or calibration sequence is to establish the clock-to-data phase relationship between the data from a given memory device among the memory components 210 and a given memory data signal interface 278. The method begins in step 402, where the initialization and configuration logic 282 selects one of the memory data signal interfaces 278. As shown in Figure 4C, memory data signal interface A may be selected. Then, the initialization and configuration logic 282 may, optionally, issue one or more commands through the memory control signal interface 276 and optionally, memory address signal interface 275, to one or more of the memory components 210 connected to memory data signal interface A. The commands issued through the memory controller signal interface 276 and optionally, memory address signal interface 275, will have the effect of getting the memory components 210 to receive or return previously received data in a predictable pattern, sequence, and timing so that the interface circuit 250 can determine the clock-to-data phase relationships between the memory device and the specific memory data signal interface. In specific DRAM memory systems such as DDR2 and DDR3 SDRAM memory systems, multiple clocking relationships must all be tracked, including clock-to-data and clock-to-DQS. For the purposes of this application, the clock-to-data phase

relationship is taken to encompass all clocking relationships on a specific memory data interface, including and not limited to clock-to-data and clock-to-DQS.

[0054] In step 404, the initialization and configuration logic 282 performs training to determine clock-to-data phase relationship between the memory data interface A and data from memory components 210 connected to the memory data interface A. In step 406, the initialization and configuration logic 282 directs the memory data interface A to set the respective delay adjustments so that clock-to-data phase variances of each of the memory components 210 connected to the memory data interface A can be eliminated. In step 408, the initialization and configuration logic 282 determines whether all memory data signal interfaces 278 within the interface circuit 250 have been calibrated. If so, the method ends in step 410 with the interface circuit 250 entering normal operation regime. If, however, the initialization and configuration logic 282 determines that not all memory data signal interfaces 278 have been calibrated, then in step 412, the initialization and configuration logic 282 selects a memory data signal interface that has not yet been calibrated. The method then proceeds to step 402, described above.

[0055] The flow diagram of Figure 4C shows that the memory data signal interfaces 278 are trained sequentially, and after memory data interface A has been trained, memory data interface B is similarly trained, and respective delays set for data interface B. The process is then repeated until all of the memory data signal interfaces 278 have been trained and respective delays are set. In other embodiments, the respective memory data signal interfaces 278 may be trained in parallel. After the calibration sequence is complete, control returns to the normal flow diagram as illustrated in Figure 4D.

[0056] Figure 4D illustrates a flow diagram of method steps showing the operations of the interface circuit 250 in response to the various commands, according to one embodiment of the present invention. While the method steps are described with relation to the computer platform 201 illustrated in Figures 2B and 2C, any system performing the method steps, in any order, is within the scope of the

present invention.

[0057] The method begins in step 420, where the interface circuit 250 enters normal operation regime. In step 422, the system control signal interface 272 determines whether a new command has been received from the memory controller 225. If so, then, in step 424, the emulation and command translation logic 280 translates the address and issues the command to one or more memory components 210 through the memory address signal interface 275 and the memory control signal interface 276. Otherwise, the system control signal interface 272 waits for the new command (i.e., the method returns to step 422, described above).

[0058] In the general case, the emulation and command translation logic 280 may perform a series of complex actions to handle different commands. However, the description of all commands are not vital to the enablement of the seamless burst merging functionality of the interface circuit 250, and the flow diagram in Figure 4D describes only those commands that are vital to the enablement of the seamless burst merging functionality. Specifically, the READ command, the WRITE command and the CALIBRATION command are important commands for the seamless burst merging functionality.

[0059] In step 426, the emulation and command translation logic 280 determines whether the new command is a READ command. If so, then the method proceeds to step 428, where the emulation and command translation logic 280 receives data from the memory component 210 via the memory data signal interface 278. In step 430, the emulation and command translation logic 280 directs the data path logic 281 to select the memory data signal interface 278 that corresponds to one of the memory components 210 that the READ command was issued to. In step 432, the emulation and command translation logic 280 aligns the data received from the memory component 210 to match the clock-to-data phase with the interface circuit 250. In step 434, the emulation and command translation logic 280 directs the data path logic 281 to move the data from the selected memory data signal interface 278 to the system data signal interface 274 and re-drives the data out of the system data

signal interface 274. The method then returns to step 422, described above.

[0060] If, however, in step 426, the emulation and command translation logic determines that the new command is not a READ command, the method then proceeds to step 436, where the emulation and command translation logic determines whether the new command is a WRITE command. If so, then, in step 438, the emulation and command translation logic 280 directs the data path logic 281 to receive data from the memory controller 225 via the system data signal interface 274. In step 440, the emulation and command translation logic 280 selects the memory data signal interface 278 that corresponds to the memory component 210 that is the target of the WRITE commands and directs the data path logic 281 to move the data from the system data signal interface 274 to the selected memory data signal interface 278. In step 442, the selected memory data signal interface 278 aligns the data from system data signal interface 274 to match the clock-to-data phase relationship of the data with the target memory component 210. In step 444, the memory data signal interface 278 re-drives the data out to the memory component 210. The method then returns to step 422, described above.

[0061] If, however, in step 436, the emulation and command translation logic determines that the new command is not a WRITE command, the method then proceeds to step 446, where the emulation and command translation logic determines whether the new command is a CALIBRATION command. If so, then the method ends at step 448, where the emulation and command translation logic 280 issues a calibration request to the initialization and configuration logic 282. The calibration sequence has been described in Figure 4C.

[0062] The flow diagram in Figure 4D illustrates the functionality of the burst merging interface circuit 250 for individual commands. As an example, Figure 4A illustrates the functionality of the burst merging interface circuit for the case of three consecutive read commands. Figure 4A shows that data bursts A0, A1, A2 and A3 may be received by Data Path A, data bursts B0, B1, B2 and B3 may be received by Data Path B, and data bursts C0, C1, C2 and C3 may be received by Data Path C,

wherein the respective data bursts may all have different clock-to-data phase relationships and in fact part of the data bursts may overlap in time. However, through the mechanism illustrated in the flow diagram contained in Figure 4D, data bursts from Data Paths A, B, and C are all phase aligned to the clock signal of the interface circuit 250 before they are driven out of the system data signal interface 274 and appear as a single contiguous data burst with no idle cycles necessary between the bursts. Figure 4B shows that once the different data bursts from different memory circuits are time aligned to the same clock signal used by the interface circuit 250, the memory controller 225 can issue commands with minimum spacing – constrained only by the full utilization of the data bus - and the seamless burst merging functionality occur as a natural by-product of the clock-to-data phase alignment of data from the individual memory components 210 connected via parallel data paths to interface circuit 250.

[0063] Figure 5A illustrates a compute platform 500A that includes a platform chassis 510, and at least one processing element that consists of or contains one or more boards, including at least one motherboard 520. Of course the platform 500 as shown might comprise a single case and a single power supply and a single motherboard. However, it might also be implemented in other combinations where a single enclosure hosts a plurality of power supplies and a plurality of motherboards or blades.

[0064] The motherboard 520 in turn might be organized into several partitions, including one or more processor sections 526 consisting of one or more processors 525 and one or more memory controllers 524, and one or more memory sections 528. Of course, as is known in the art, the notion of any of the aforementioned sections is purely a logical partitioning, and the physical devices corresponding to any logical function or group of logical functions might be implemented fully within a single logical boundary, or one or more physical devices for implementing a particular logical function might span one or more logical partitions. For example, the function of the memory controller 524 might be implemented in one or more of

the physical devices associated with the processor section 526, or it might be implemented in one or more of the physical devices associated with the memory section 528.

[0065] Figure 5B illustrates one exemplary embodiment of a memory section, such as, for example, the memory section 528, in communication with a processor section 526. In particular, Figure 5B depicts embodiments of the invention as is possible in the context of the various physical partitions on structure 520. As shown, one or more memory modules 530_1 – 530_N each contain one or more interface circuits 550_1 – 550_N and one or more DRAMs 542_1 – 542_N positioned on (or within) a memory module 530_1 .

[0066] It must be emphasized that although the memory is labeled variously in the figures (e.g. memory, memory components, DRAM, etc), the memory may take any form including, but not limited to, DRAM, synchronous DRAM (SDRAM), double data rate synchronous DRAM (DDR SDRAM, DDR2 SDRAM, DDR3 SDRAM, etc.), graphics double data rate synchronous DRAM (GDDR SDRAM, GDDR2 SDRAM, GDDR3 SDRAM, etc.), quad data rate DRAM (QDR DRAM), RAMBUS XDR DRAM (XDR DRAM), fast page mode DRAM (FPM DRAM), video DRAM (VDRAM), extended data out DRAM (EDO DRAM), burst EDO RAM (BEDO DRAM), multibank DRAM (MDRAM), synchronous graphics RAM (SGRAM), phase-change memory, flash memory, and/or any other type of volatile or non-volatile memory.

[0067] Many other partition boundaries are possible and contemplated, including positioning one or more interface circuits 550 between a processor section 526 and a memory module 530 (see Figure 5C), or implementing the function of the one or more interface circuits 550 within the memory controller 524 (see Figure 5D), or positioning one or more interface circuits 550 in a one-to-one relationship with the DRAMs 542_1 – 542_N and a memory module 530 (see 5E), or implementing the one or more interface circuits 550 within a processor section 526 or even within a processor 525 (see Figure 5F). Furthermore, the system 220 illustrated in Figures 2B and 2C is analogous to the computer platform 500 and 510 illustrated in Figures 5A-5F, the

memory controller 225 illustrated in Figures 2B and 2C is analogous to the memory controller 524 illustrated in Figures 5A-5F, the interface circuit 250 illustrated in Figures 2B and 2C is analogous to the interface circuits 550 illustrated in Figures 5A-5F, and the memory components 210 illustrated in Figures 2B and 2C are analogous to the DRAMs 542 illustrated in Figures 5A-5F. Therefore, all discussions of Figures 2B, 2C, and 4A-4D apply with equal force to the systems illustrated in Figures 5A-5F.

[0068] One advantage of the disclosed interface circuit is that the idle cycles required to switch from one memory device to another memory device may be eliminated while still maintaining accurate timing reference for data transmission. As a result, memory system bandwidth may be increased, relative to the prior art approaches, without changes to the system interface or commands.

[0069] While the foregoing is directed to embodiments of the present invention, other and further embodiments of the invention may be devised without departing from the basic scope thereof. Therefore, the scope of the present invention is determined by the claims that follow.

We Claim:

1. An interface circuit configured to combine a plurality of data bursts returned by a plurality of memory devices into a contiguous data burst, the interface circuit comprising:
 - a system control signal interface adapted to receive a first command from a memory controller; and
 - emulation and command translation logic adapted to:
 - translate a first address associated with the first command,
 - issue the first command to a first memory device within the plurality of memory devices corresponding to the first address,
 - determine that the first command is a read command,
 - select a memory data signal interface corresponding to the first memory device,
 - receive a first data burst from the first memory device,
 - delay the first data burst to eliminate a first clock-to-data phase between the first memory device and the interface circuit, and
 - re-drive the first data burst to the memory controller.
2. The interface circuit of claim 1, wherein:
 - the system control signal interface is further adapted to receive a second command from the memory controller; and
 - the emulation and command translation logic is further adapted to:
 - translate a second address associated with the second command,
 - issue the second command to a second memory device within the plurality of memory devices corresponding to the second address,
 - determine that the second command is a read command,
 - select a memory data signal interface corresponding to the second memory device,
 - receive a second data burst from the second memory device,

delay the second data burst to eliminate a second clock-to-data phase
between the second memory device and the interface circuit,
and
re-drive the second data burst to the memory controller so that the first
data burst and the second data burst are combined into a third
data burst that is contiguous.

3. The interface circuit of claim 2, further comprising a means for determining optimized timing for re-driving the first data burst and the second data burst to eliminate idle cycles between the first data burst and the second data burst.
4. The interface circuit of claim 3, further comprising scheduling logic configured to order and concatenate the first and second data bursts.
5. The interface circuit of claim 4, wherein the scheduling logic is implemented using registers, multiplexors, and combination logic.
6. The interface circuit of claim 2, wherein:
the system control signal interface is further adapted to receive a third
command from the memory controller;
the emulation and command translation logic is further adapted to:
translate a third address associated with the third command,
issue the third command to a third memory device within the plurality
of memory devices corresponding to the third address,
determine that the third command is a write command,
select a memory data signal interface corresponding to the third
memory device,
receive a third data burst from the memory controller; and
the memory data signal interface corresponding to the third memory device is
adapted to:

delay the third data burst to eliminate a third clock-to-data phase between the third memory device and the interface circuit, and re-drive the third data burst to the third memory device.

7. The interface circuit of claim 6, further comprising initialization and configuration logic, wherein:

the system control signal interface is further adapted to receive a fourth command from the memory controller;

the emulation and command translation logic is further adapted to determine that the fourth command is a calibration command; and

the initialization and calibration logic is adapted to:

perform calibration sequence to determine the first, second, and third clock-to-data phases,

based on the first clock-to-data phase, set a first delay adjustment within the memory data signal interface corresponding to the first memory device,

based on the second clock-to-data phase, set a second delay adjustment within the memory data signal interface corresponding to the second memory device, and

based on the third clock-to-data phase, set a third delay adjustment within the memory data signal interface corresponding to the third memory device.

8. The interface circuit of claim 7, wherein each of the first, second, and third clock-to-data phases includes and clock-to-data (clock-to-DQ) value and a clock-to-data strobe signal (clock-to-DQS) value.

9. The interface circuit of claim 1, further comprising a memory address signal interface, a memory control signal interface, a memory clock signal interface, a system address signal interface, a system clock signal interface, and a system data

signal interface.

10. The interface circuit of claim 1, wherein the emulation and command translation logic is capable of emulating an interface protocol associated with each memory device within the plurality of memory devices.

11. The interface circuit of claim 1, wherein one or more memory devices within the plurality of memory devices are arranged a stack.

12. The interface circuit of claim 1, wherein each memory device within the plurality of memory devices comprises a dynamic random access memory chip.

13. The interface circuit of claim 1, wherein the plurality of memory devices are arranged in one stack, and the interface circuit is integrated within the one stack.

14. The interface circuit of claim 13, wherein the interface circuit comprises an intelligent buffer chip.

15. The interface circuit of claim 14, wherein the two or more of the memory devices comprise dynamic random access memories, and the intelligent buffer chip and the dynamic random access memories are included in a dual in-line memory module.

16. The interface circuit of claim 1, wherein the interface circuit is coupled directly to a memory module.

17. The interface circuit of claim 16, wherein the memory module comprises a dual in-line memory module.

18. The interface circuit of claim 1, wherein the interface circuit is connected to a

circuit board.

19. The interface circuit of claim 1, wherein the interface circuit is included within a computing device.

20. An apparatus comprising:
a plurality of memory devices;
a memory controller; and
an interface circuit electrically connected to the plurality of memory devices and the memory controller and configured to combine a plurality of data bursts returned by the plurality of memory devices into a contiguous data burst, wherein the interface circuit comprises:
a system control signal interface adapted to receive a first command from the memory controller, and
emulation and command translation logic adapted to:
translate a first address associated with the first command,
issue the first command to a first memory device within the plurality of memory devices,
determine that the first command is a read command,
select a memory data signal interface corresponding to the first memory device,
receive a first data burst from the first memory device,
delay the first data burst to eliminate a first clock-to-data phase between the one of the first memory device and the interface circuit, and
re-drive the first data burst to the memory controller.

21. The apparatus of claim 20 further comprising a motherboard.

22. The apparatus of claim 21, further comprising a platform chassis.

ABSTRACT OF THE DISCLOSURE

One embodiment of the present invention sets forth an interface circuit configured to combine time staggered data bursts returned by multiple memory devices into a larger contiguous data burst. As a result, an accurate timing reference for data transmission that retains the use of data (DQ) and data strobe (DQS) signals in an infrastructure-compatible system while eliminating the cost of the idle cycles required for data bus turnarounds to switch from reading from one memory device to reading from another memory device, or from writing to one memory device to writing to another memory device may be obtained, thereby increasing memory system bandwidth relative to the prior art approaches.

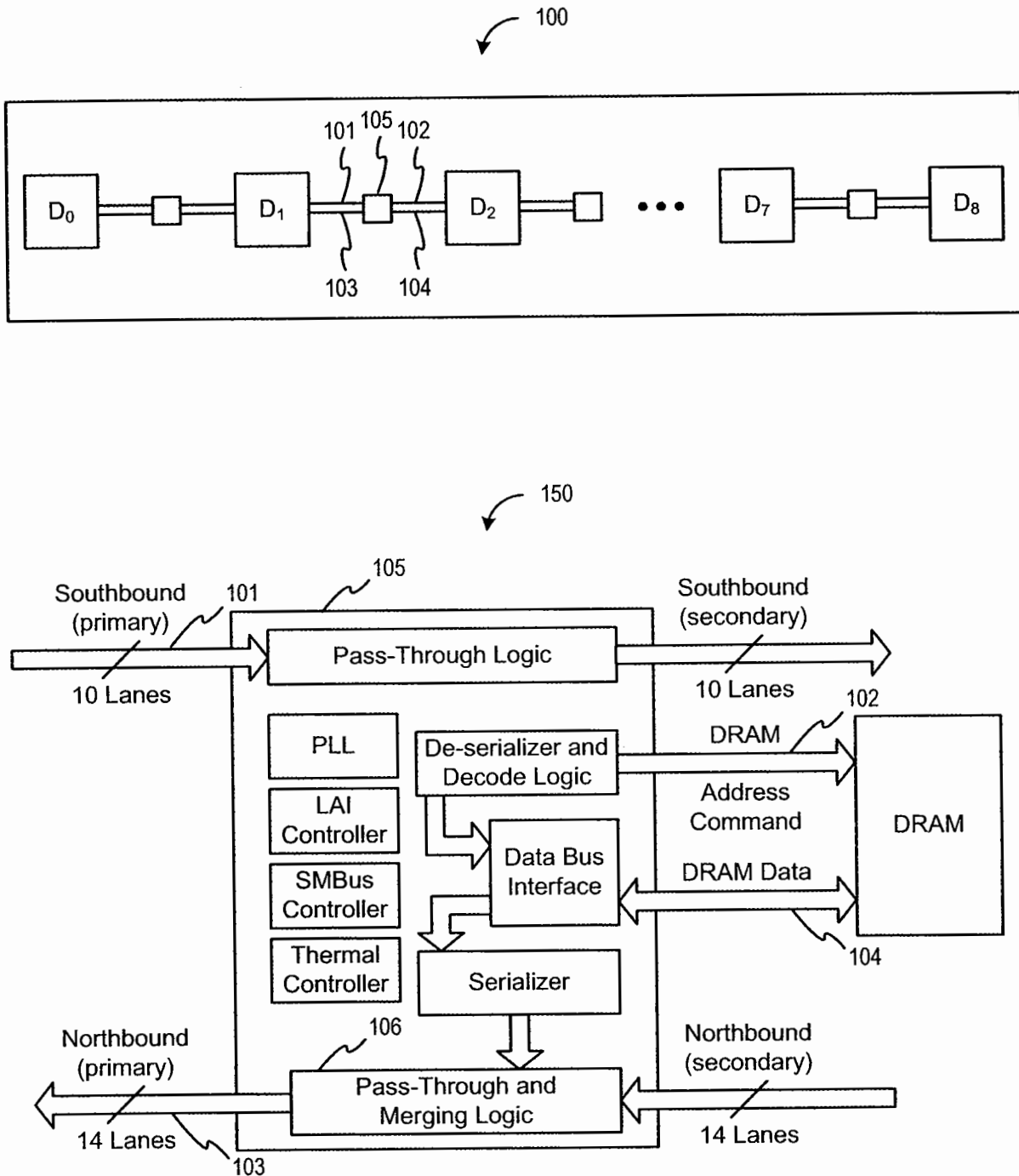


FIG. 1

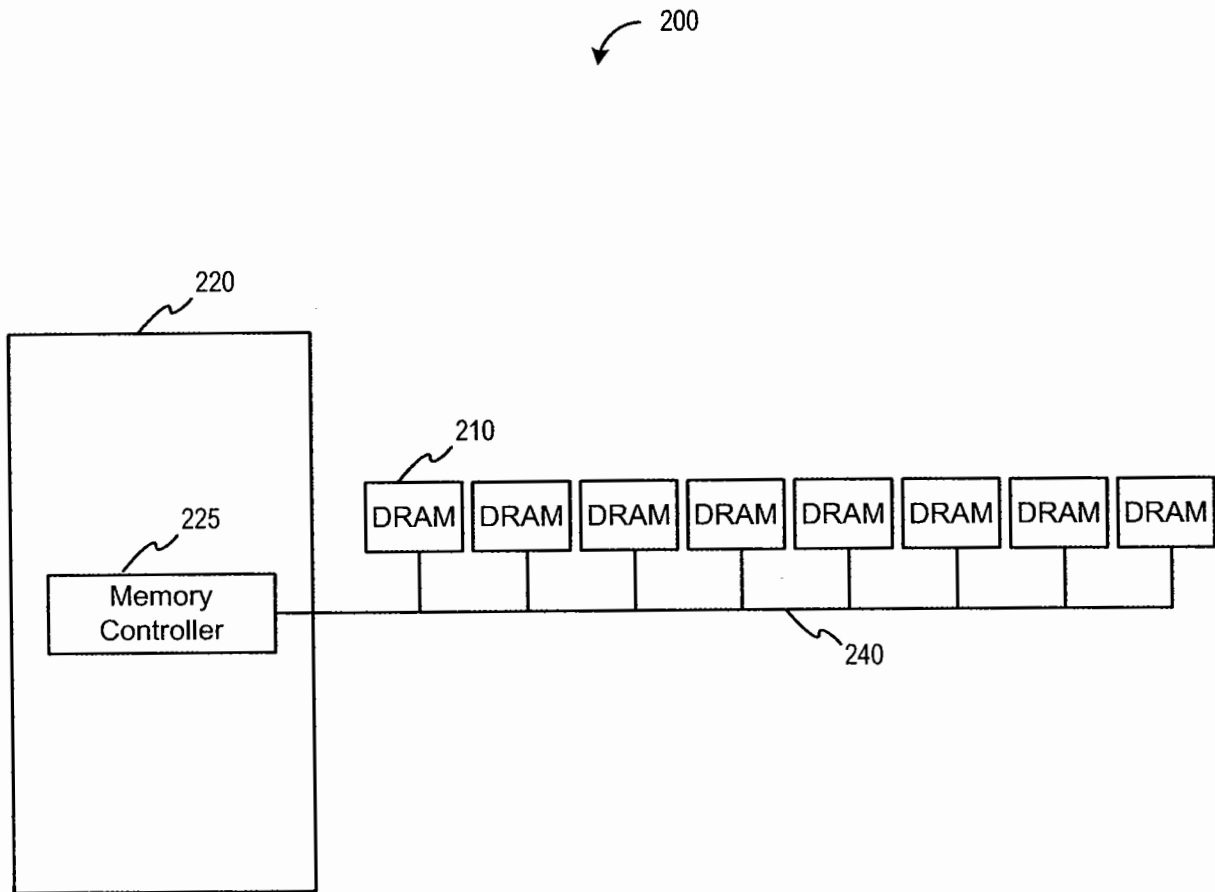


FIG. 2A

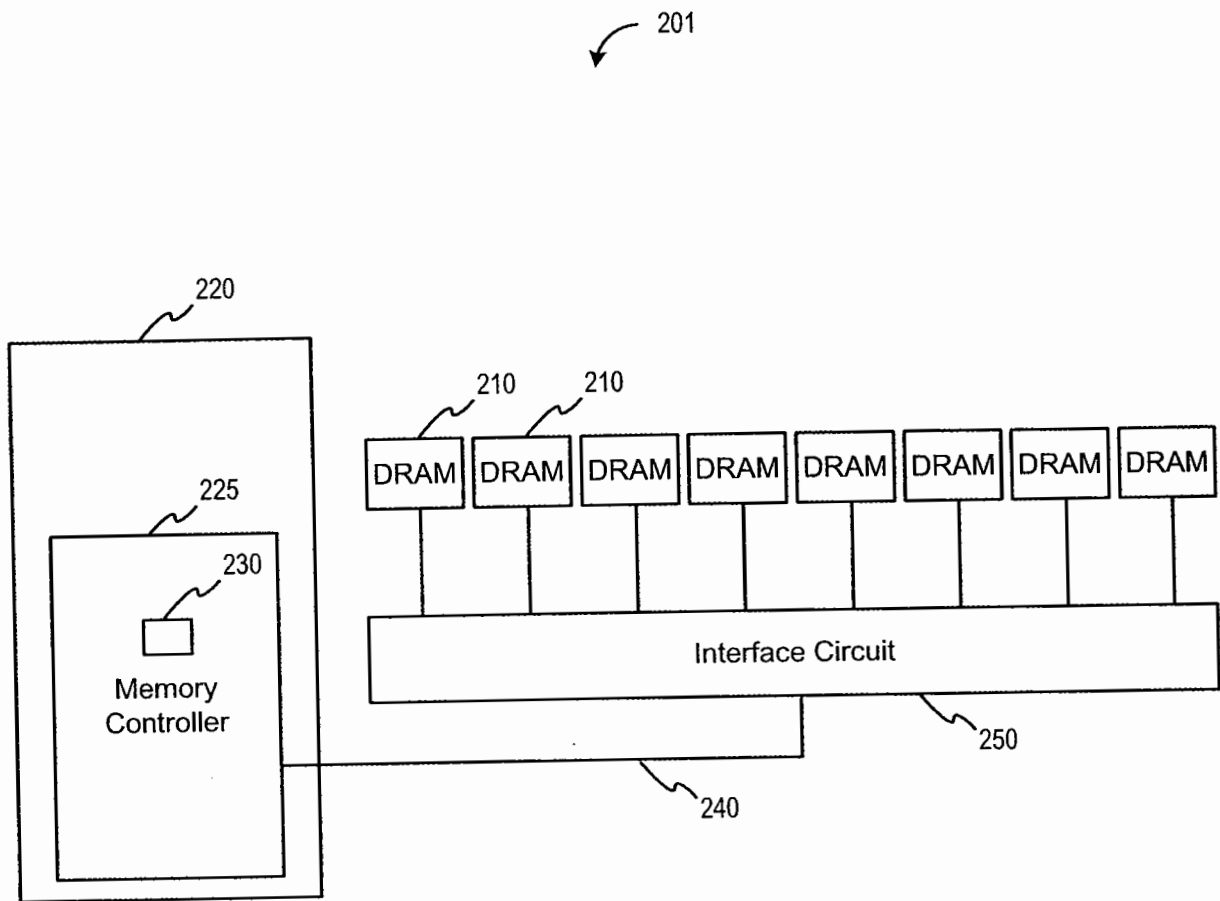


FIG. 2B

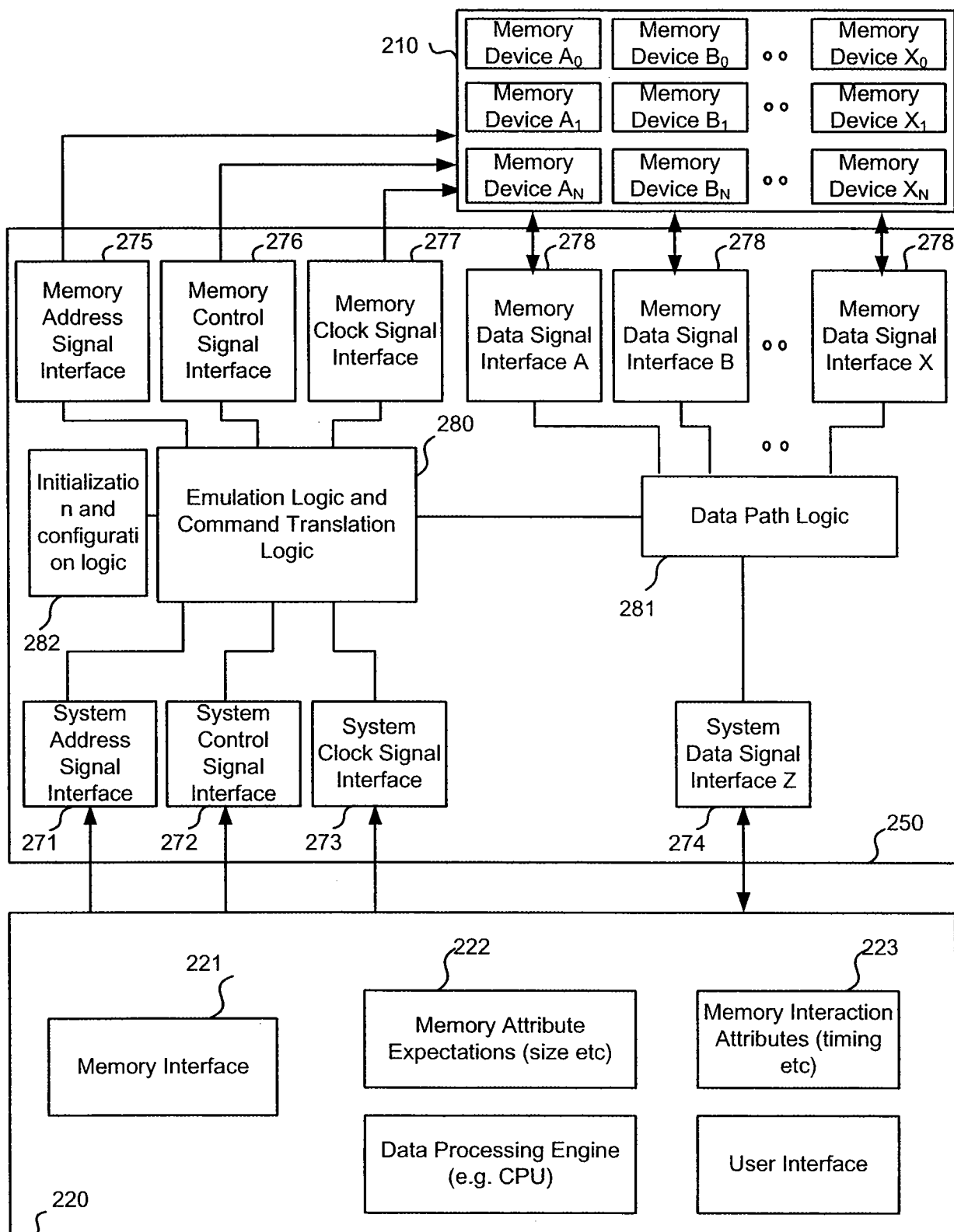


FIG. 2C

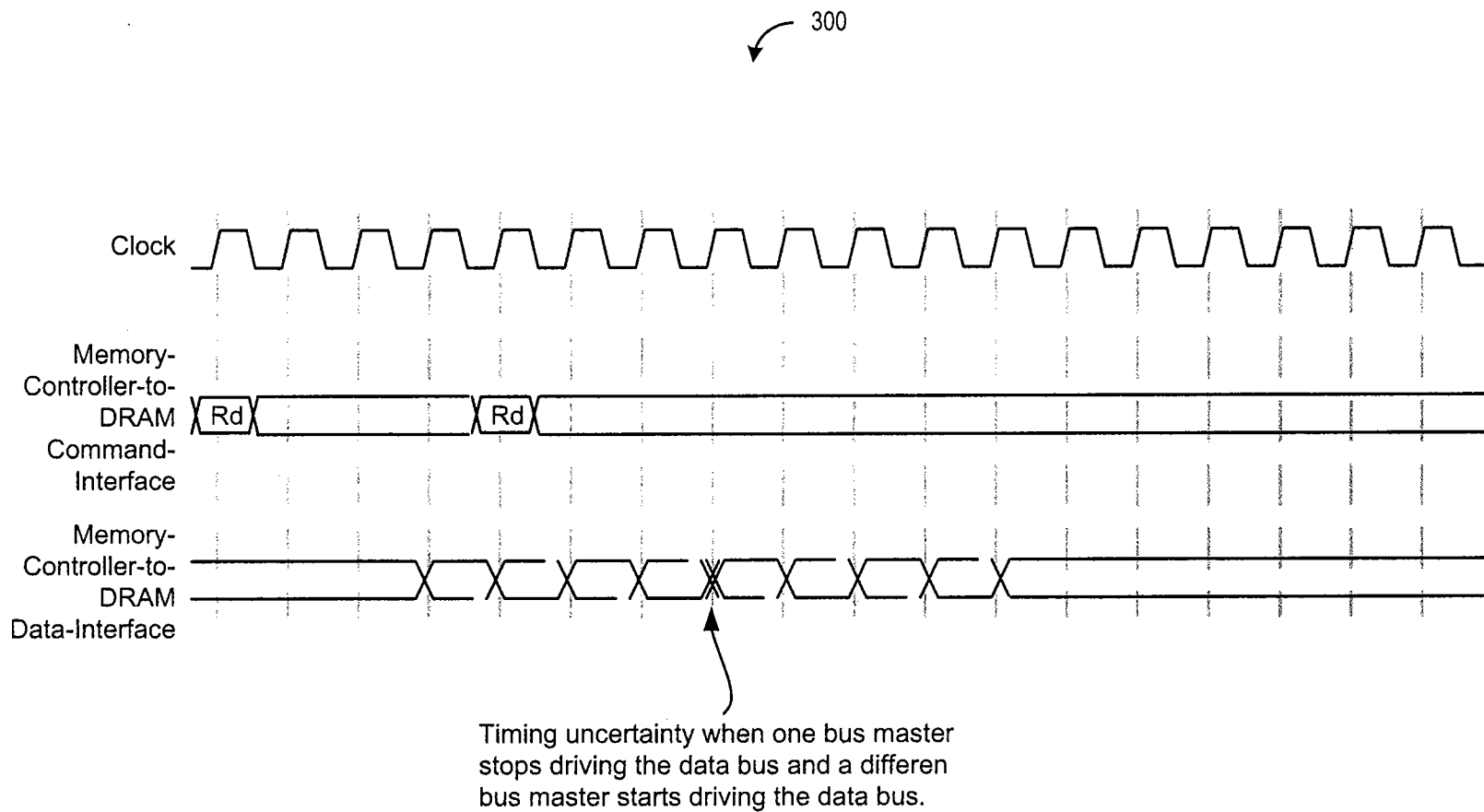


FIG. 3A

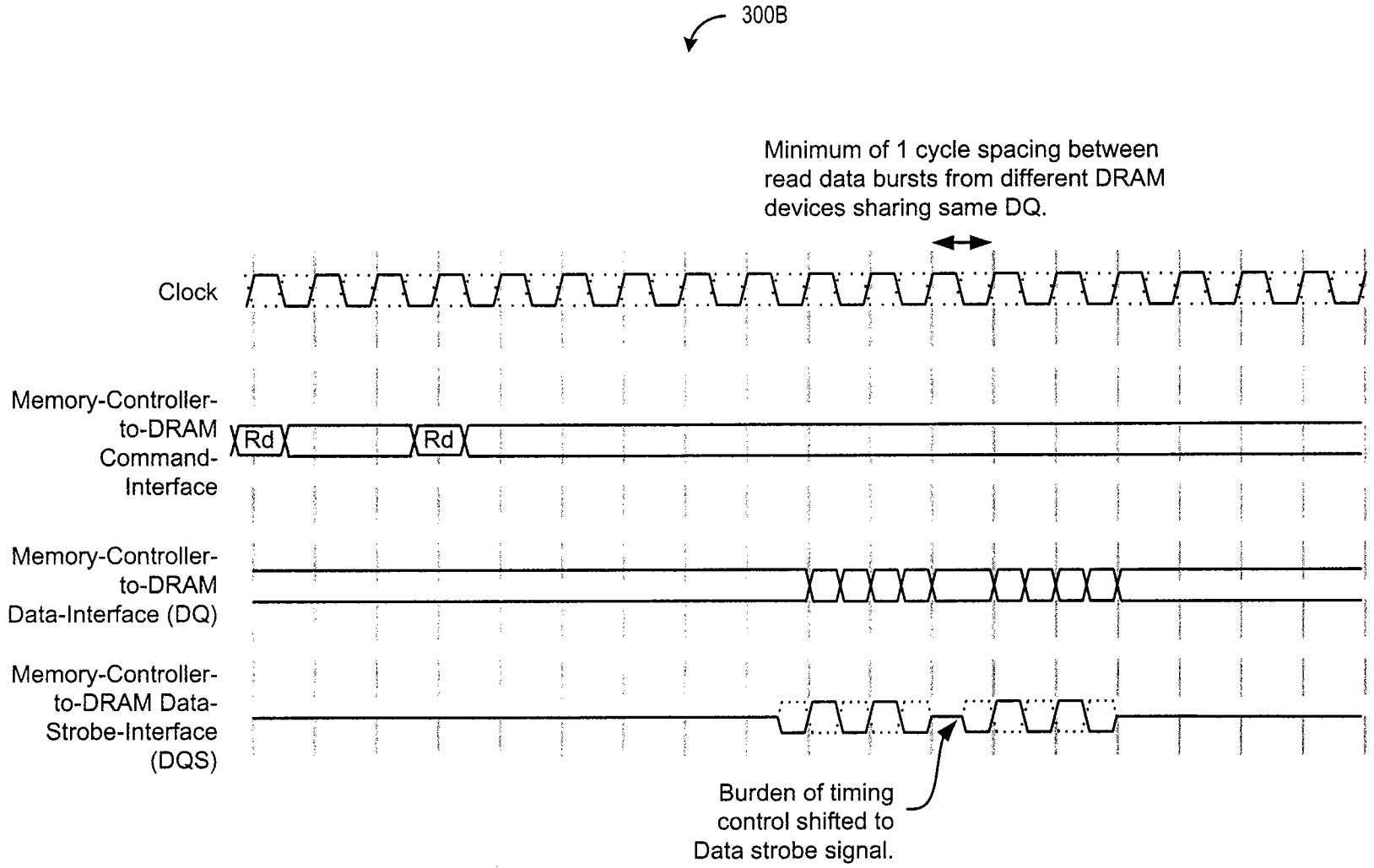


FIG. 3B

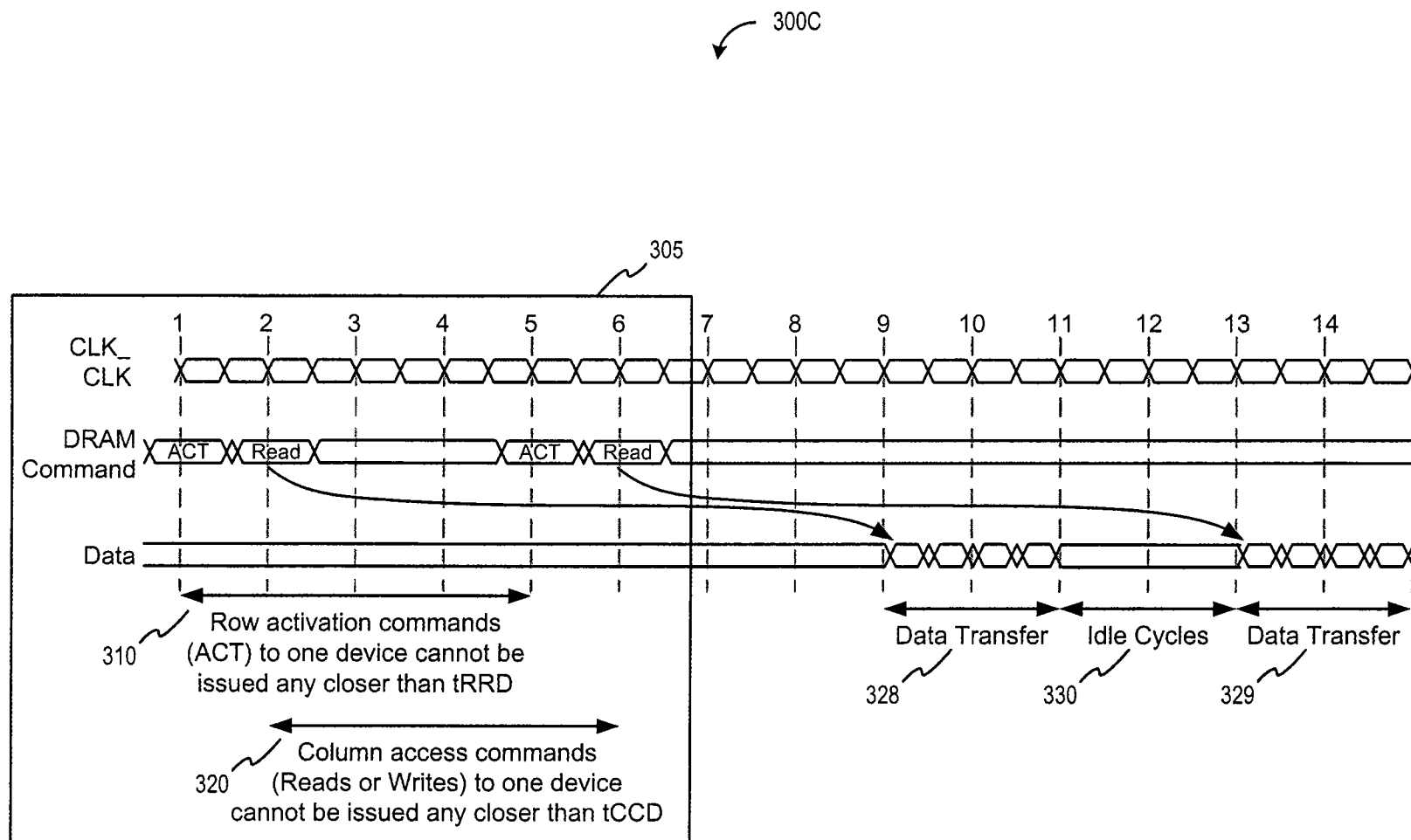


FIG. 3C

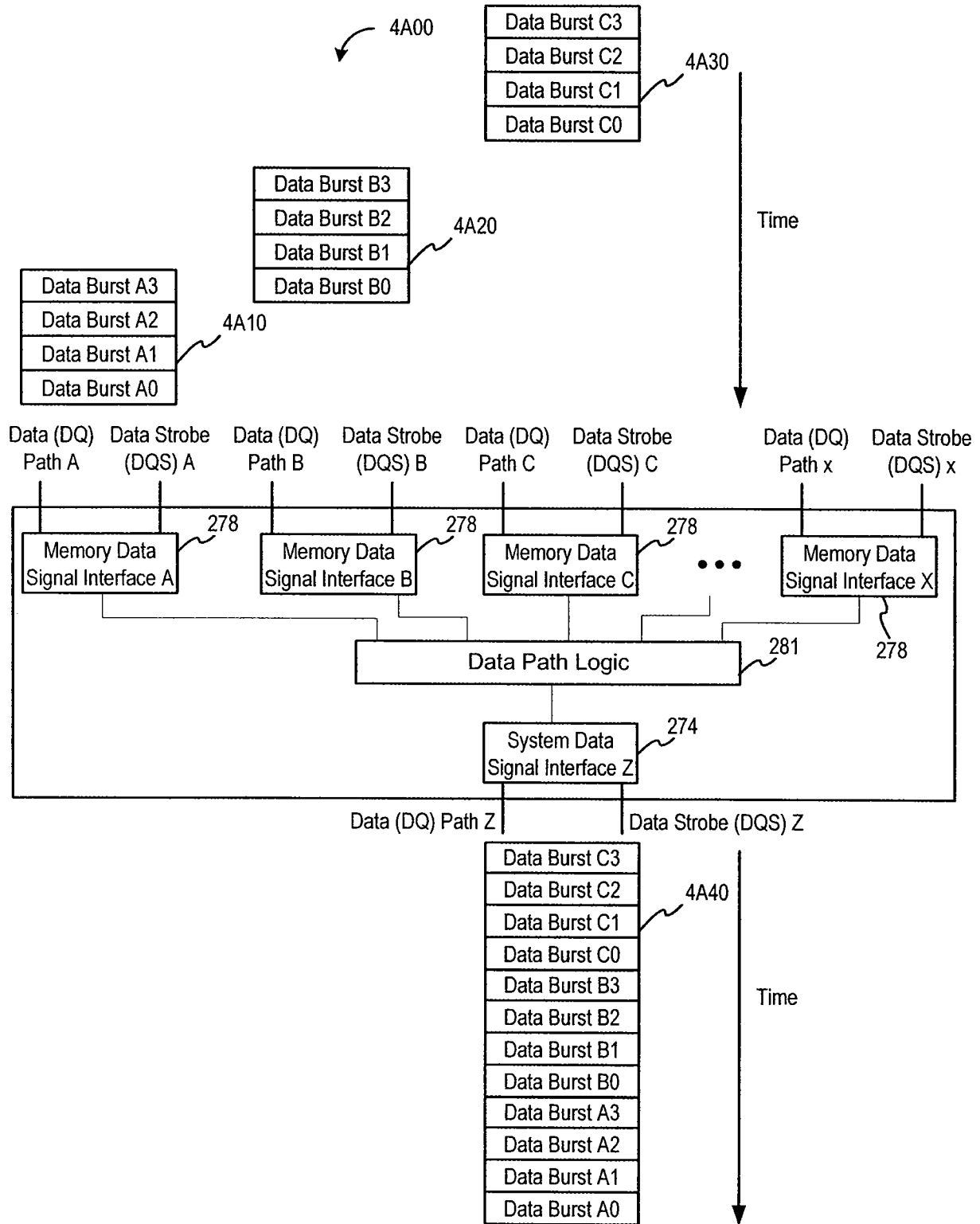


FIG. 4A

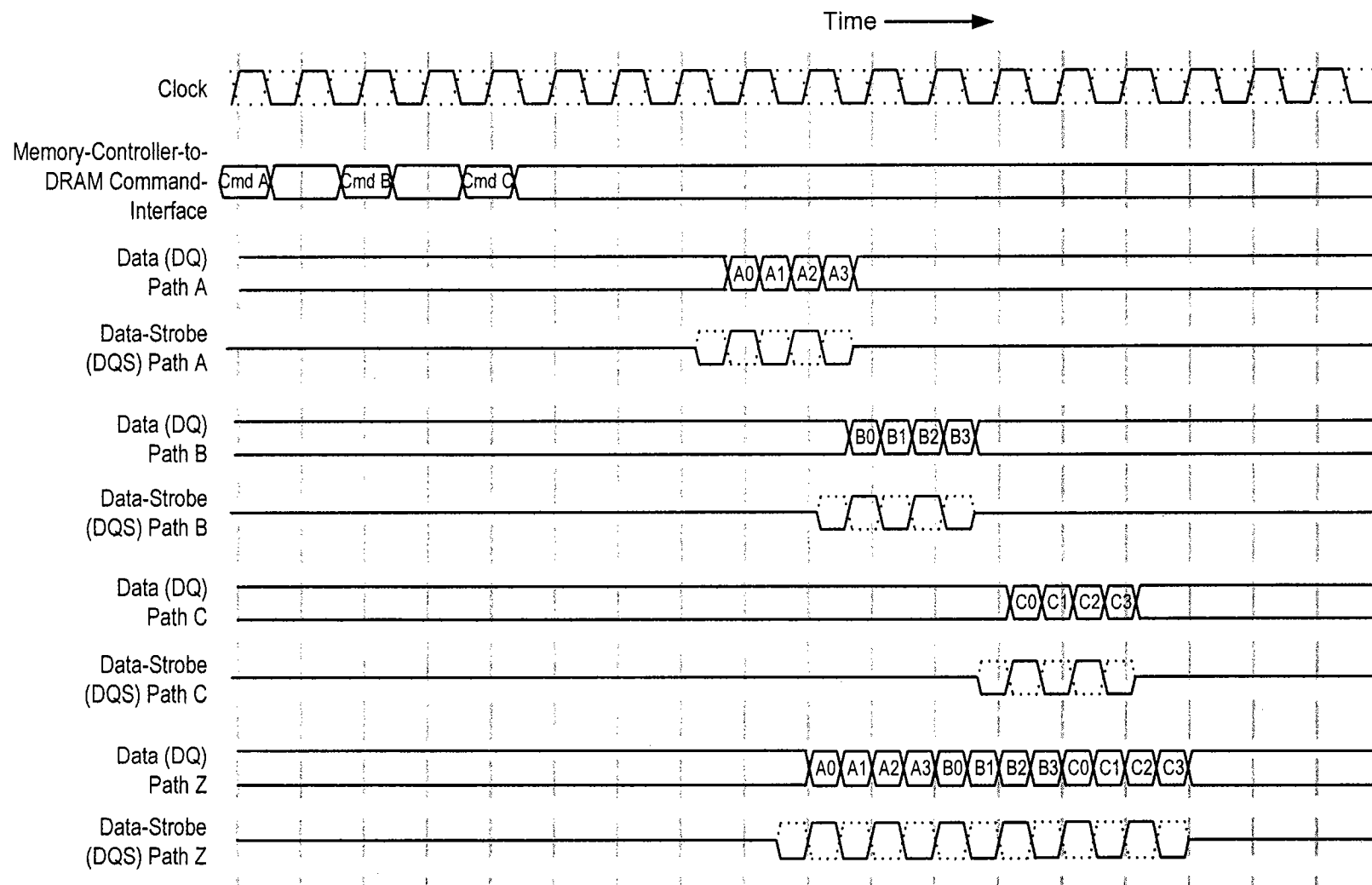


FIG. 4B

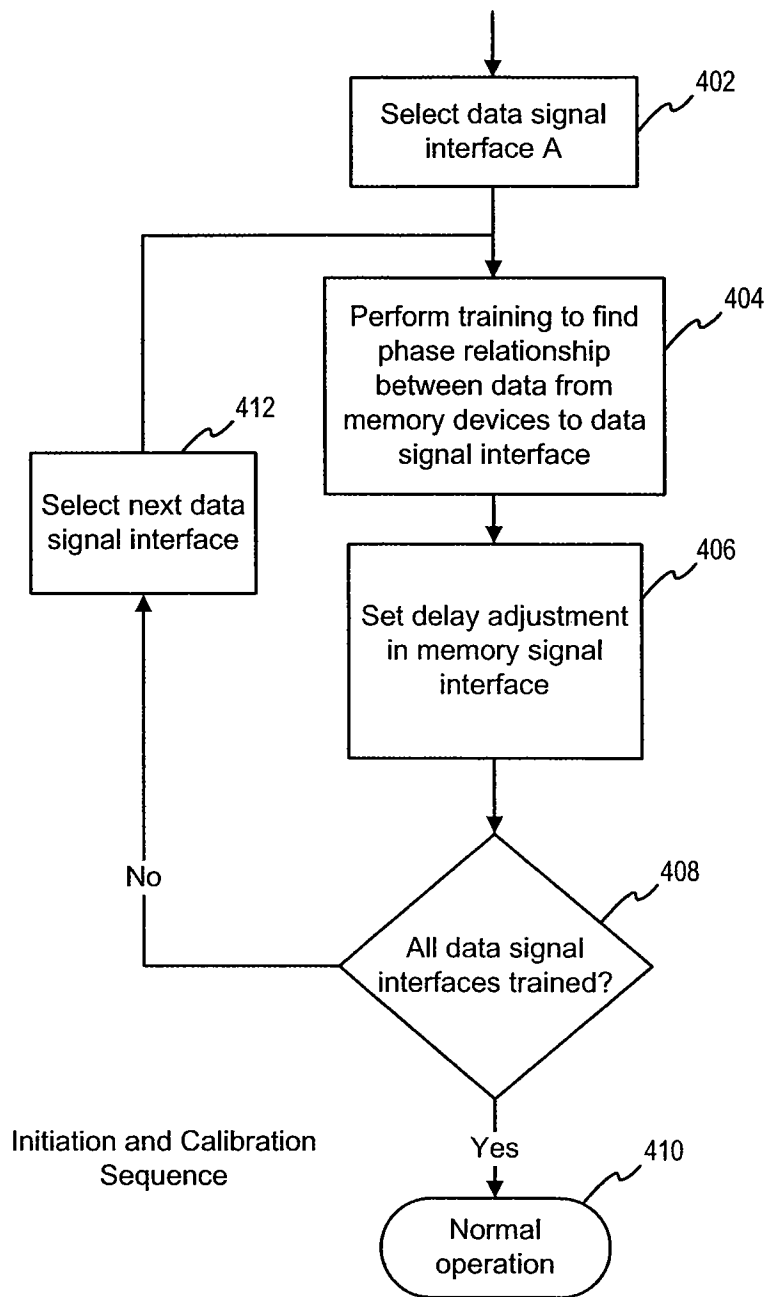
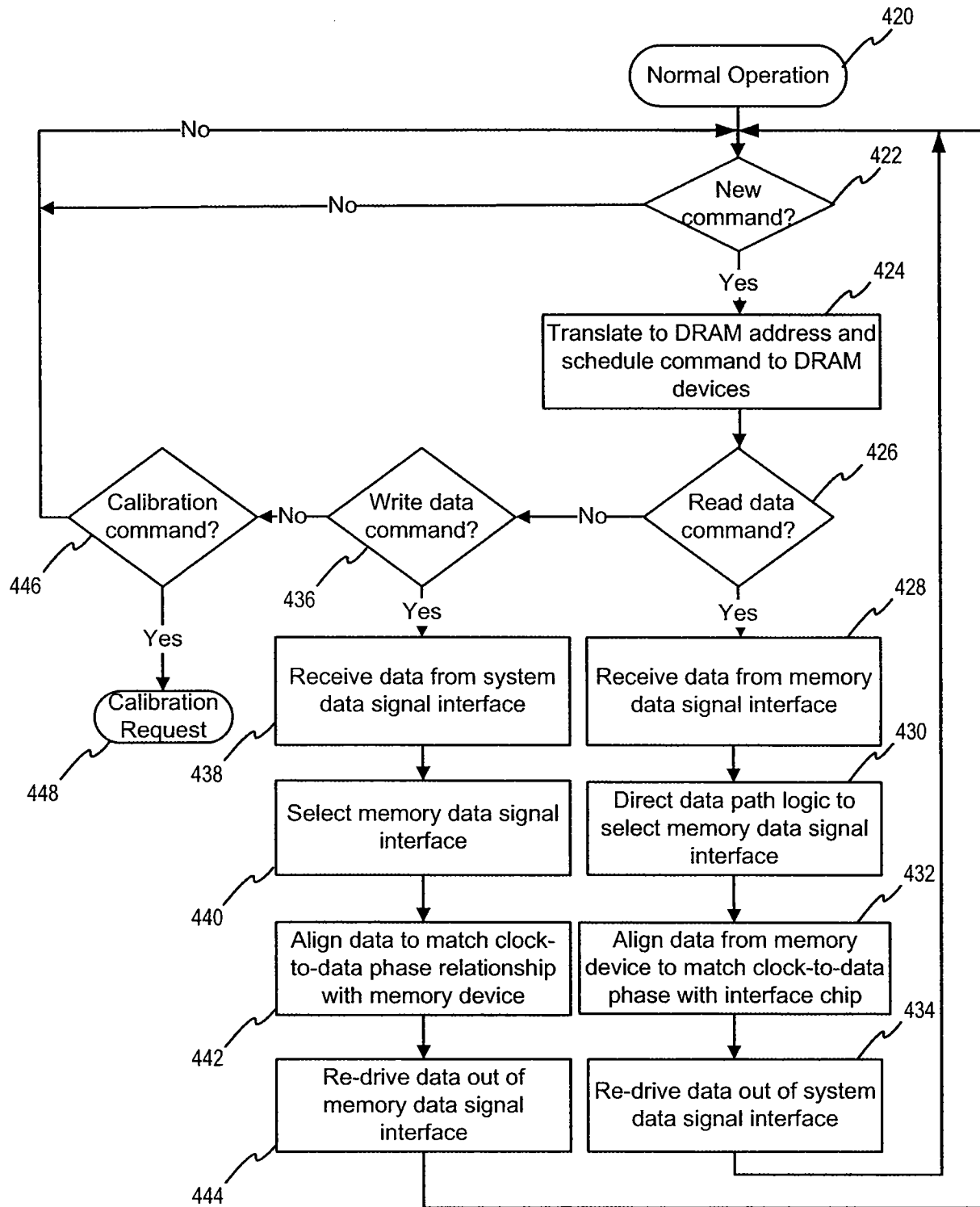
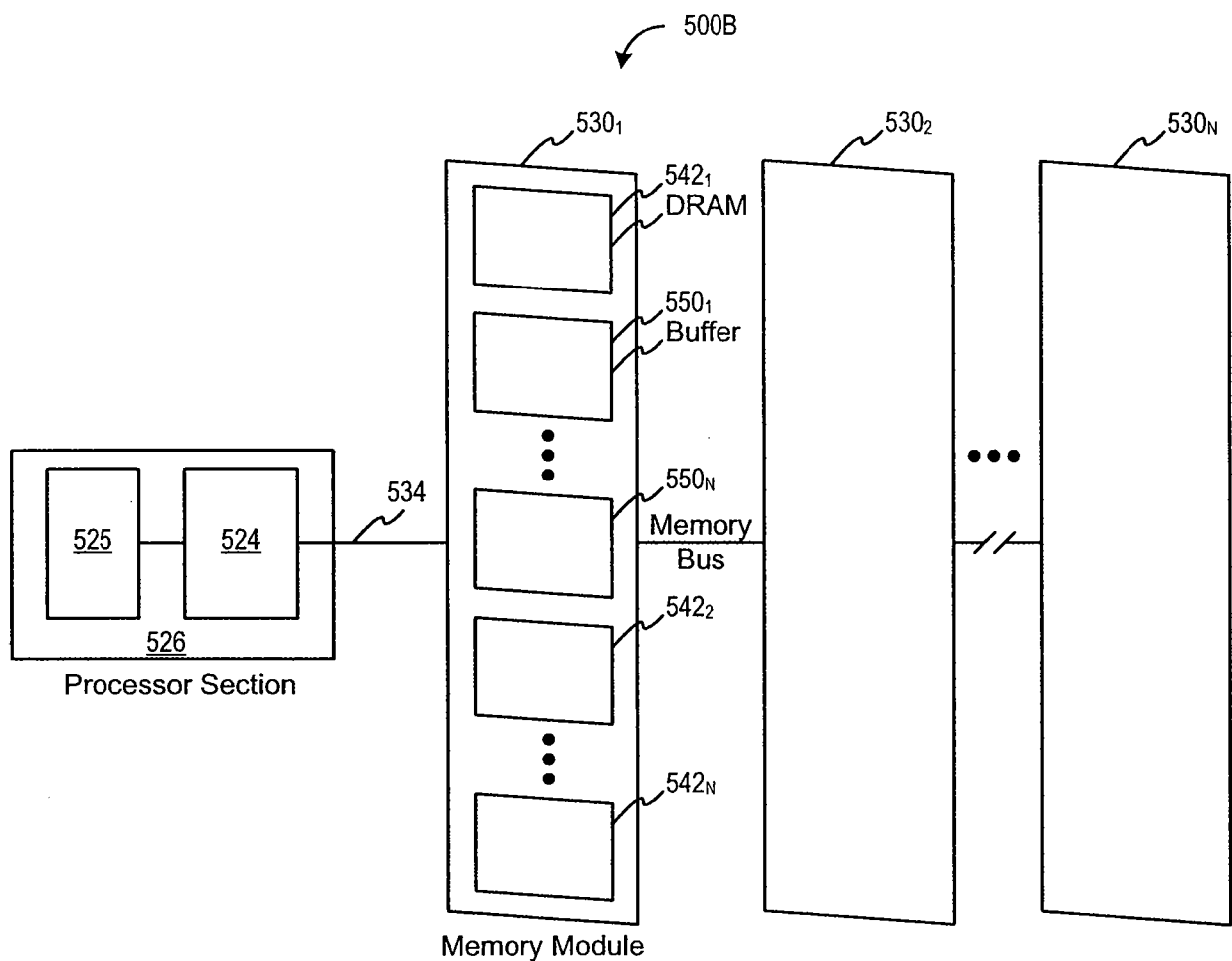
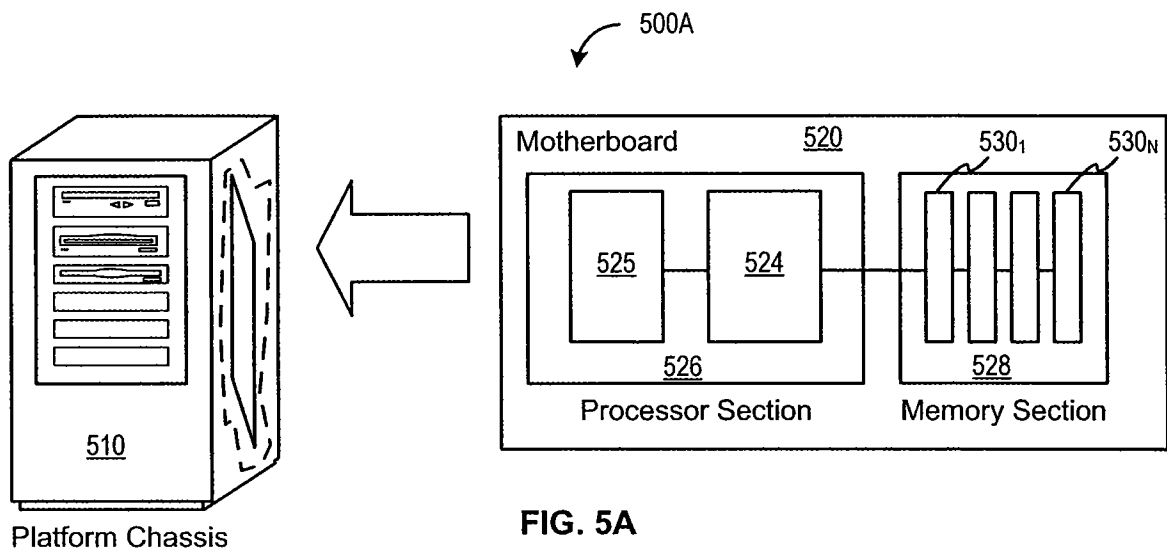


FIG. 4C



Normal Command Flow

FIG. 4D



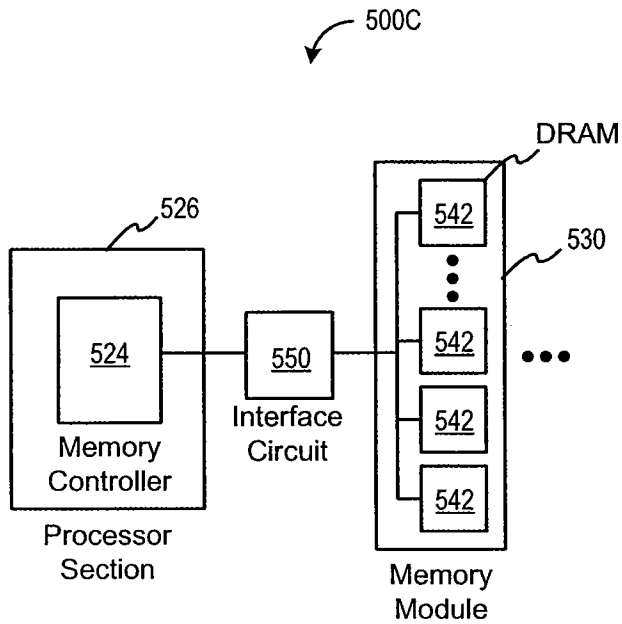


FIG. 5C

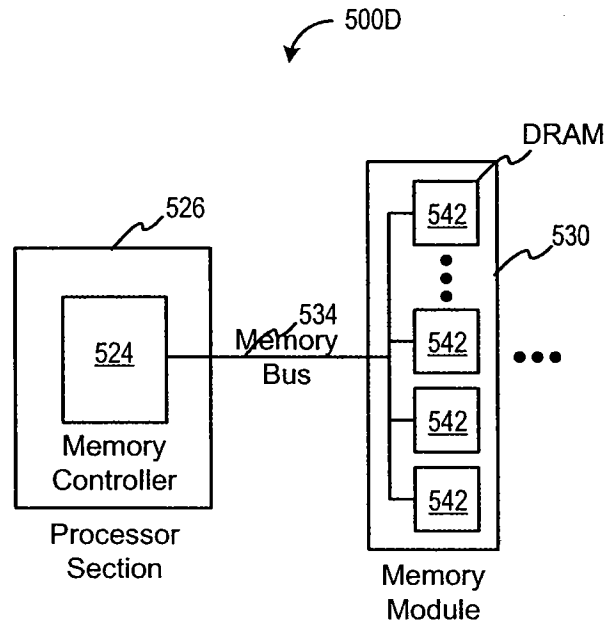


FIG. 5D

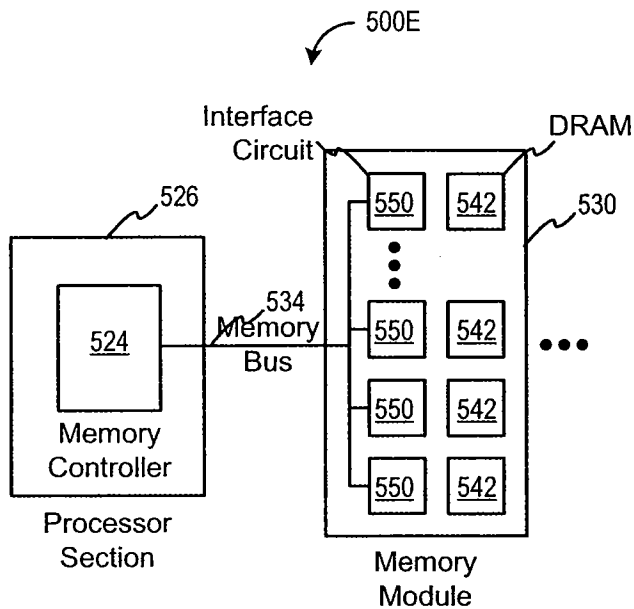


FIG. 5E

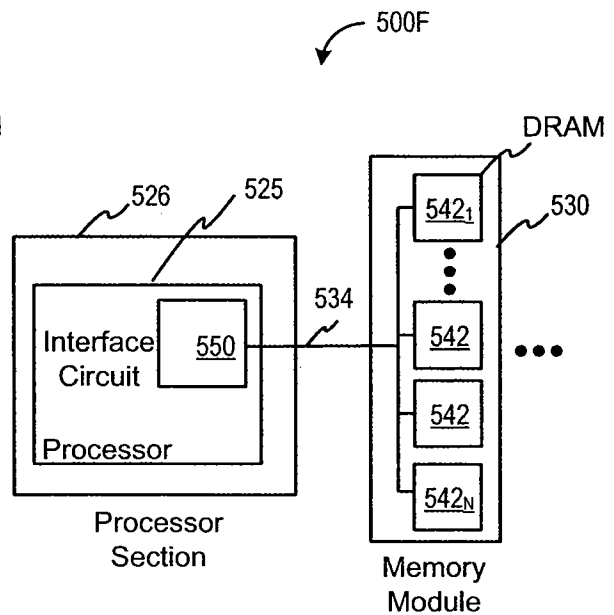


FIG. 5F