US Patent & Trademark Office
Patent Public Search | Text View

United States Patent Application Publication	20250266081
Kind Code	A1
Publication Date	August 21, 2025
Inventor(s)	Nygren; Aaron John et al.

READ CLOCK START AND STOP FOR SYNCHRONOUS MEMORIES

Abstract

A data processor comprises data processor comprising a memory controller, wherein the data processor is configured to control the memory controller to program at least one mode register of a memory to set a read clock mode to one of a plurality of settings, wherein the plurality of settings includes a read-only mode in which the memory starts a read clock signal in response to a command, and subsequently perform a read cycle by receiving data during the read cycle using the read clock signal.

Inventors: **Nygren; Aaron John (Boise, ID), Gopalakrishnan; Karthik (Cupertino, CA), Liu; Tsun Ho (Boston, MA)**

Applicant: **Advanced Micro Devices, Inc.** (Santa Clara, CA)

Family ID: **1000008577976**

Assignee: **Advanced Micro Devices, Inc. (Santa Clara, CA)**

Appl. No.: **19/195264**

Filed: **April 30, 2025**

Related U.S. Application Data

parent US continuation 17850299 20220627 parent-grant-document US 12315551 child US 19195264
us-provisional-application US 63287151 20211208

Publication Classification

Int. Cl.: **G11C11/4076** (20060101); **G06F1/08** (20060101); **G06F1/10** (20060101); **G06F1/3234** (20190101); **G06F1/3237** (20190101); **G06F3/06** (20060101); **G06F12/00** (20060101);

G06F13/00 (20060101)

U.S. Cl.:

CPC **G11C11/4076** (20130101); **G06F1/08** (20130101); **G06F1/10** (20130101); **G06F3/0604** (20130101); **G06F3/0659** (20130101); **G06F3/0671** (20130101); G06F1/3237 (20130101); G06F1/3275 (20130101); G06F12/00 (20130101); G06F13/00 (20130101)

Background/Summary

[0001] This application claims priority to provisional application U.S. 63/287,151, filed Dec. 8, 2021, and to U.S. application Ser. No. 17/850,299, filed Jun. 27, 2022, the entire contents of which are incorporated herein by reference.

CROSS REFERENCE TO RELATED APPLICATION

[0002] Related subjected matter is found in U.S. patent application Ser. No. 17/850,499, filed Jun. 27, 2022, invented by Aaron John Nygren, Karthik Gopalakrishnan, and Tsun Ho Liu and assigned to the assignee hereof.

BACKGROUND

[0003] Modern dynamic random-access memory (DRAM) provides high memory bandwidth by increasing the speed of data transmission on the bus connecting the DRAM and one or more data processors, such as graphics processing units (GPUs), central processing units (CPUs), and the like. DRAM is typically inexpensive and high density, thereby enabling large amounts of DRAM to be integrated per device. Most DRAM chips sold today are compatible with various double data rate (DDR) DRAM standards promulgated by the Joint Electron Devices Engineering Council (JEDEC). Typically, several DDR DRAM chips are combined onto a single printed circuit board substrate to form a memory module that can provide not only relatively high speed but also scalability.

[0004] DDR DRAMs are synchronous because they operate in response to a free-running clock signal that synchronizes the issuance of commands from the host processor to the memory and therefore the exchange of data between the host processor and the memory. DDR DRAMs are responsive to the clock signal to synchronize commands and can be used to generate read data strobe signals. For example, DDR DRAMs receive write data using a center-aligned data strobe signal known as “DQS” provided by the host processor, in which the memory captures data on both the rising and falling edges of DQS. Similarly, DDR DRAMs provide read data synchronously with an edge-aligned DQS in which the DDR DRAMs provide the DQS signal. During read cycles, the host processor delays the DQS signal internally to align it with the center portion of the DQ signals generally by an amount determined at startup by performing data eye training. Some DDR DRAMs, such as graphics DDR, version six (GDDR6) DRAMs receive both a main clock signal and a separate write clock signal and programmably generate a read data strobe signal.

[0005] However, while these enhancements have improved the speed of DDR memory used for computer systems' main memory, further improvements are desirable.

Description

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] FIG. **1** illustrates in block diagram for a data processing system according to some embodiments;

[0007] FIG. **2** illustrates in block diagram form the GDDR DRAM of FIG. **1** according to some embodiments;

[0008] FIG. **3** illustrates in table showing a mode register setting for the receive clock modes of the memory of FIG. **2**;
[0009] FIG. **4** illustrates a flow chart useful in understanding the operation of the memory of FIG. **2** according to some embodiments;
[0010] FIG. **5** is a timing diagram showing properties of the receive clock timing of the memory of FIG. **2** according to some embodiments;
[0011] FIG. **6** is a timing diagram showing further properties of the receive clock timing of the memory of FIG. **2** according to some embodiments; and
[0012] FIG. **7** is a timing diagram showing yet further properties of the receive clock timing of the memory of FIG. **2** according to some embodiments.
[0013] In the following description, the use of the same reference numerals in different drawings indicates similar or identical items. Unless otherwise noted, the word "coupled" and its associated verb forms include both direct connection and indirect electrical connection by means known in the art, and unless otherwise noted any description of direct connection implies alternate embodiments using suitable forms of indirect electrical connection as well. The following Detailed Description is directed to electronic circuitry, and the description of a block shown in a drawing figure implies the implementation of the described function using suitable electronic circuitry, unless otherwise noted.
DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS
[0014] A data processor comprises data processor comprising a memory controller, wherein the data processor is configured to control the memory controller to program at least one mode register of a memory to set a read clock mode to one of a plurality of settings, wherein the plurality of settings includes a read-only mode in which the memory starts a read clock signal in response to a command, and subsequently perform a read cycle by receiving data during the read cycle using the read clock signal.
[0015] A method includes programming at least one mode register of a memory, the programming comprising setting a read clock mode to one of a plurality of settings. The plurality of settings includes an always running mode and a read-only mode in which the memory starts a read clock signal in response to a command. A read cycle is subsequently performed by a memory controller by receiving data during the read cycle using the read clock signal.
[0016] According to various embodiments disclosed herein, a memory provides the capability to start and stop the read clock (RCK) that the memory provides to the memory controller based on the commands provided to the memory. Moreover, this behavior can be programmably enabled and disabled based on the value of one or more bits of a mode register.
[0017] FIG. **1** illustrates in block diagram for a data processing system **100** according to some embodiments. Data processing system **100** includes generally a data processor in the form of a graphics processing unit (GPU) **110**, a host central processing unit (CPU) **120**, a double data rate (DDR) memory **130**, and a graphics DDR (GDDR) memory **140**.
[0018] GPU **110** is a discrete graphics processor that has extremely high performance for optimized graphics processing, rendering, and display, but requires a high memory bandwidth for performing these tasks. GPU **110** includes generally a set of command processors **111**, a graphics single instruction, multiple data (SIMD) core **112**, a set of caches **113**, a memory controller **114**, a DDR physical interface circuit (PHY) **115**, and a GDDR PHY **116**.
[0019] Command processors **111** are used to interpret high-level graphics instructions such as those specified in the OpenGL programming language. Command processors **111** have a bidirectional connection to memory controller **114** for receiving the high-level graphics instructions, a bidirectional connection to caches **113**, and a bidirectional connection to graphics SIMD core **112**. In response to receiving the high-level instructions, command processors **111** issue SIMD instructions for rendering, geometric processing, shading, and rasterizing of data, such as frame data, using caches **113** as temporary storage. In response to the graphics instructions, graphics SIMD core **112** executes the low-level instructions on a large data set in a massively parallel

fashion. Command processors **111** use caches **113** for temporary storage of input data and output (e.g., rendered and rasterized) data. Caches **113** also have a bidirectional connection to graphics SIMD core **112**, and a bidirectional connection to memory controller **114**.

[0020] Memory controller **114** has a first upstream port connected to command processors **111**, a second upstream port connected to caches **113**, a first downstream bidirectional port, and a second downstream bidirectional port. As used herein, "upstream" ports are on a side of a circuit toward a data processor and away from a memory, and "downstream" ports are on a side if the circuit away from the data processor and toward a memory. Memory controller **114** controls the timing and sequencing of data transfers to and from DDR memory **130** and GDDR memory **140**. DDR and GDDR memory support asymmetric accesses, that is, accesses to open pages in the memory are faster than accesses to closed pages. Memory controller **114** stores memory access commands and processes them out-of-order for efficiency by, e.g., favoring accesses to open pages, disfavoring frequent bus turnarounds from write to read and vice versa, while observing certain quality-of-service objectives.

[0021] DDR PHY **115** has an upstream port connected to the first downstream port of memory controller **114**, and a downstream port bidirectionally connected to DDR memory **130**. DDR PHY **115** meets all specified timing parameters of the implemented version or versions of DDR memory **130**, such as DDR version five (DDR5), and performs training operations at the direction of memory controller **114**. Likewise, GDDR PHY **116** has an upstream port connected to the second downstream port of memory controller **114**, and a downstream port bidirectionally connected to GDDR memory **200**. GDDR PHY **116** meets all specified timing parameters of the implemented version of GDDR memory **140**, such as GDDR version seven (GDDR7), and performs training operations at the direction of memory controller **114**.

[0022] The inventors have discovered that the read clock (RCK) that the memory, e.g., GDDR memory **200**, provides to GDDR PHY **116** can be programmed to operate in certain new and advantageous ways. According to some embodiments, the memory has a "read-only" mode. In the read-only mode, the memory provides the RCK signal with read commands in which it causes the RCK signal to start toggling during a read preamble period before a data transmission of a read command, and to continue to toggle at least to the end of a read postamble period following the read command. The read-only mode provides the ability to reduce power consumption during workloads in which read operations are or can be infrequent.

[0023] GDDR memory **200** also has an "always on" mode. In the always-on mode, GDDR memory **200** provides the RCK signal continuously as long as a write clock (WCK) is received from the host, e.g., the memory controller or memory PHY of a host processor chip. The always on mode provides the ability for the host processor PHY to stay locked and avoid the need for resynchronization during a preamble period.

[0024] According to some embodiments, the memory further has a disabled mode in which the memory does not provide any read clock signal.

[0025] FIG. **2** illustrates in block diagram form GDDR memory **200** of FIG. **1** according to some embodiments. GDDR memory **200** generally includes a control circuit **210**, an address path **220**, a memory array and page buffers **230**, and a data read path **240**, a set of bond pads **250**, and a data write path.

[0026] Control circuit **210** includes a command decoder **211**, mode registers **212**, and an RCK logic and state machine **213**. Command decoder **211** decodes commands received from command and address pins (not shown in FIG. **2**) into one of several supported commands defined by the memory's command truth table. One type of command decoded by command decoder **211** is a mode register set (MRS) command. The MRS command causes the command decoder to provide settings to the indicated mode register in which the settings are contained on the ADDRESS inputs. MRS commands have been known in the context of DRAMs for quite some time, and vary between different GDDR DRAM versions. Mode registers **212** store the programmed settings, and

in some cases, output information about the GDDR DRAM. RCK logic and state machine **213** has a first input connected to the output of command decoder **211**, a second input connected to certain outputs of mode registers **212**, and an output. As will be described further, RCK logic and state machine **213** further processes a read clock flag. The read clock flag indicates, on the fly, the read clock behavior after the read postamble period, i.e., during the "inter-amble" period. The read clock flag can be encoded with the command signals, with a separate signal, or in any other known way.
[0027] Address path **220** receives a multi-bit ADDRESS signal, and includes an input buffer **221** and an address latch **222** for each address signal, a set of row decoders **223**, and a set of column decoders **224**. Input buffer **221** receives and buffers the corresponding multi-bit ADDRESS signal, and provides a multi-bit buffered ADDRESS signal in response. Address latch **222** has an input connected to the output of input buffer **221**, an output, and a clock input receiving a signal labelled "WCK". Address latch **222** latches the bits of the buffered address on a certain clock edge, e.g., the rising edge, and functions not only as a write clock during write commands, but also as a main clock that is used to capture commands. Row decoders **223** have an input connected to the output of address latch **222**, and an output. Column decoders **224** have an input connected to the output of address latch **222**, and an output.
[0028] Memory arrays and page buffers **230** are organized into a set of individual memory arrays known as banks that are separately addressable. For example, GDDR memory **200** may have a total of **16** banks. Each bank can have only one "open" page at a time, in which the open page has its contents read into a corresponding page buffer for faster read and write accesses. Row decoders **223** select a row in the accessed bank during an activate command, and the contents of the indicated row are read into the page buffer and the row is ready for read and write accesses. Column decoders **224** select a column of the row in response to a column address.
[0029] Data read path **240** includes a read queue **241**, a read latch **242**, an output buffer **243**, a delay locked loop (DLL) **244**, and an RCK and RCK pins. Read queue **241** has an input connected to an output of memory arrays and page buffers **230**, and an output. Read latch **242** has an input connected to the output of read queue **241**, a clock input, and an output. Buffer **243** has an input connected to the output of read latch **242**, and an output connected to bond pads **250**. DLL **244** has an input receiving a write clock signal labelled "WCK", and an output connected to the clock input of read latch **242**. RCK driver circuit **245** has an input connected to the output of DLL **244**, a control input connected to the output of RCK logic and state machine **213**, and an output connected to the RCK and RCK pins.
[0030] Write data path **260** includes an input buffer **261**, a write latch **262**, and a write queue **263**. Input buffer **261** has an input connected to a set of bond pads **250** labelled "DQ", and an output. Write latch **262** has an input connected to the output of input buffer **261**, and an output. Write queue **263** has an input connected to the output of write latch **262**, and an output connected to memory arrays and page buffers **230**.
[0031] In operation, GDDR memory **200** allows concurrent operations in the memory banks and in one embodiment, GDDR memory **200** is compatible with one of the double data rate (DDR) standards published by the Joint Electron Device Engineering Council (JEDEC), such as the newly emerging graphics DDR, version 7 (GDDR7) standard. In order to access data, a memory accessing agent such as GPU **110** activates a row in a memory bank by issuing an activate ("ACT") command. In response to the ACT command, data from memory cells along the selected row are stored in a corresponding page buffer. In DRAMs, data reads are destructive to the contents of the memory cells, but a copy of the data is stored in the page buffer. After memory controller **114** finishes accessing data in the selected row of a bank, it closes the row by issuing a precharge ("PRE") command (or write or read command with auto-precharge, or a precharge all command). The PRE command causes the data in page buffer **124** to be rewritten to its row in the selected bank, allowing another row to then be activated. These operations are conventional in DDR memories and described in the various JEDEC standard documents and will not be described

further.
[0032] According to various embodiments disclosed herein, however, GDDR memory **200** includes a modified set of mode registers **212** that, compared to existing standards such as GDDR6, adds mode register fields that can be used to define the behavior of the RCK signal that memory **200** provides along with accessed data during a read cycle. In addition, memory **200** includes RCK logic and state machine **213** to control the output of the RCK (and optionally RCK) signals according to the behavior specified in mode registers **212**.
[0033] FIG. **3** illustrates a table **300** showing a mode register setting for the receive clock modes of the memory of FIG. **2**. Table **300** shows values of different bits or bit fields of a 12-bit mode register, in which the twelve bits correspond to address signals by which the mode registers are loaded. Table **300** has six columns, including an OP code (operational code) column corresponding to certain bit or bits of the mode register, a Function column identifying the function defined by the corresponding bits, an OP code Value column specifying the different values of the OP code, and a Description column identifying the meaning of the different OP code values.
[0034] Mode register bits [**1**:**0**] are labelled “RCKMODE” and identify the selected RCK mode. A value of 00b (binary) identifies the Disabled mode, in which the RCK is not provided by memory **200**. This mode is the default mode.
[0035] A value of 01b indicates the Read Only mode. As will be described further below, in the Read Only mode, RCK is provided during one or more read cycles and each read cycle contains both a preamble and a postamble. When Read Only mode is selected, an interamble behavior is defined when consecutive reads are separated by more than the minimum amount of spacing, i.e., by at least t.sub.CCD+1 RCK cycles, in which t.sub.CCD is the minimum command-to-command delay time. In general, during the Read Data mode, the RCK starts a preamble period before the transfer of data in a read cycle, and ends a preamble period after a read cycle. In particular, it starts toggling coincident with data transfer for a read command (RD), a read with auto-precharge command (RDA), and with a read training (RDTR) command. It stops with a clear condition. In some embodiments, the clear condition includes receipt of a write command (a write command (WR), a write with auto-precharge command (WRA), or a write training (WRTR) command), receipt of an all banks idle state indication, or entry into a power down state.
[0036] A value of 10b indicates an Always Running mode. In the Always Running mode, RCK runs continuously as long as WCK, used to generate RCK, is received by memory **200**.
[0037] A value of 11b is reserved (RSVD) but allows the definition of a new mode of providing the RCK signal to be added in the future using this mode register structure.
[0038] Mode register bit [**2**] defines a receive clock type (RCKTYPE). A value of 0b indicates that GDDR memory **200** provides the RCK signal as a single-ended signal, i.e., RCK does not toggle. A value of 1b indicates that both the RCK and the RCK signals toggle as a differential signal.
[0039] Mode register bits [**4**:**3**] define the length of the static preamble period. To allow a memory controller to lock to the preamble, each preamble period has a static period, a low-speed period, and a high-speed period. During the static period, the read clock signal is driven in its inactive state, i.e., RCK is driven low and RCK is driven high. A value of 00b indicates a static period of 0 clock cycles, i.e., no static period. Values of 01b, 10b, and 11b define static periods of 2, 4, and 6 cycles, respectively.
[0040] Mode register bit [**5**] is not defined and is reserved for future use (“RFU”).
[0041] Mode register bits [**7**:**6**] define the length of the high-speed preamble period. A value of 00b indicates a high-speed preamble period of 0 clock cycles, i.e., no high-speed preamble period. Values of 01b, 10b, and 11b define static periods of 2, 4, and 6 cycles, respectively.
[0042] Mode register bit [**8**] is not defined and is RFU.
[0043] Mode register bits [**10**:**9**] define the length of the low-speed preamble period. A value of 00b indicates a low-speed preamble period of 0 clock cycles, i.e., no low-speed preamble period. Values of 01b, 10b, and 11b define static periods of 1, 2, and 3 cycles. Note that while the high-speed and

low-speed preamble periods are independently programmable, if OP code bits [**7**:**6**] and [**10**:**9**] have the same values, then the high-speed and low-speed preambles are the same lengths of time.
[0044] Mode register bit [**11**] is not defined and is RFU.
[0045] It should be apparent that these mode register encodings are just one possible way to encode these values, and other encodings are possible. For example, instead of using a dedicated mode register, these bits can be distributed among multiple mode registers, for example in otherwise unused or reserved bit positions. Moreover, the choice of available values for the static, low-speed, and high-speed preamble are somewhat arbitrary and may be varied in different embodiments.
[0046] FIG. **4** illustrates a flow chart **400** useful in understanding the operation of memory **200** of FIG. **2** according to some embodiments. Flow chart **400** governs the start and stop behavior and the inter-amble behavior of RCK when RCKMODE is set to Read Only. Flow chart **400** defines a flag known as the RCKON flag.
[0047] Flow starts in action box **410** when a first command is received. A decision box **420** determines whether the command is a read command (such as one of a read command (RD), read with auto-precharge command (RDA), or read training (RDTR) command) and if so the state of the RCKON flag. If the command is not a read command, or if it is a read command and the RCKON flag is cleared, then flow proceeds to action box **430**. In action box **430**, RCK stops toggling after the read postamble period.
[0048] If the command is a read command and the RCKON state variable is set to 1, then flow proceeds to action box **440**. In action box **440**, memory **200** continues to toggle the RCK signal after the postamble period for the read command. From this point on, the state of RCKON becomes a don't-care. Flow proceeds to a decision box **450**, which determines whether a clear condition has been received. In some embodiments, the clear condition is one or more of a write command (e.g., any one or more of a write command (WR), a write with auto-precharge command (WRA), or a write read training (WRTR) command), an all-banks idle state condition, or entry into a power down state). If a clear condition is not received, then flow returns to decision box **450**. If a clear condition is received, then flow proceeds to action box **460**. In action box **460**, the state variable RCKON is cleared to 0, and RCK stops toggling after the read postamble, and flow returns to decision box **420**.
[0049] FIG. **5** illustrates a timing diagram **500** showing properties of the receive clock timing of memory **200** of FIG. **2** according to some embodiments. In timing diagram **500**, the horizontal axis represents time in picoseconds (ps), and the vertical axis represents the amplitude of various signals in volts. Shown along the vertical axis are three signals or signal groups of interest: a COMMAND signal, an RCK signal, and a DATA signal. Dashed lines show low-to-high and high-to-low transitions of the RCK signal and correspond to various time points.
[0050] In the example shown in timing diagram **500**, mode register **300** has been programmed for RCKMODE=Read Only, RCKTYPE=Single Ended, RCKPRE_Static=4, RCKPRE_LS=1, and RCKPRE_HS=2. Timing diagram **500** shows the issuance of a read command labelled “RD” at the second RCK transition, with the RCKON attribute set to 1. Because of the read latency, memory **200** does not provide the read data until the twenty-fifth clock cycle. Thus, prior to this RCK cycle, memory **200** provides a preamble as defined in table **300**.
[0051] In this example, the burst length is 16, and memory **200** can accept another command at t.sub.CCDMIN+1, but doesn't actually issue it until t.sub.CCDMIN+7, creating the need to define interamble behavior. As seen here, the interamble is a combination of the continuous toggling RCK after the last data transmission of the first cycle, followed by a low period of the low speed preamble of the second ready cycle, followed by high speed toggling of the preamble of the high-speed portion.
[0052] FIG. **6** is a timing diagram **600** showing further properties of the receive clock timing of memory **200** of FIG. **2** according to some embodiments. In timing diagram **600**, the horizontal axis represents time in picoseconds (ps), and the vertical axis represents the amplitude of various signals

in volts. Shown along the vertical axis are the COMMAND signal, the RCK signal, and the DATA signal as previously described. Dashed lines show low-to-high and high-to-low transitions of the RCK signal and correspond to time points designated “t.sub.1” through “t.sub.68”.

[0053] In the example shown in timing diagram **600**, mode register **300** has been programmed for RCKMODE=Read Only, RCKTYPE=Single Ended, RCKPRE_Static=4, RCKPRE_LS=1, and RCKPRE_HS=2. Timing diagram **600** shows the issuance of a read command RD at t.sub.2 with RCKON=0 and in which RCKON has not been previously set since a prior clear condition. In this case, the RD command causes memory **200** to issue a preamble as defined in the mode register, perform the read burst cycle with RCK toggling, and follow the read burst cycle by a postamble. In this case, the postamble includes a trailing static portion of two clock cycles to end the postamble period. Thus, when the RCKMODE=Read Only, the host processor can convert RCK and RCK into a read only toggling dynamically during operation according to the RCKON setting. If instead RCKTYPE=Differential, then before the preamble period, both RCK_t and RCK_c would be high due to not being driven by GDDR memory **200** but pulled high by GDDR PHY **116**, and RCK_t would be driven low during the preamble while RCK_c would remain high, until they subsequently started to toggle.

[0054] FIG. **7** is a timing diagram **700** showing yet further properties of the receive clock timing of memory **200** of FIG. **2** according to some embodiments. In timing diagram **700**, the horizontal axis represents time in picoseconds (ps), and the vertical axis represents the amplitude of various signals in volts. Shown along the vertical axis are the COMMAND signal, the RCK signal, and the DATA signal as previously described. Dashed lines show low-to-high and high-to-low transitions of the RCK signal and correspond to various time points.

[0055] In timing diagram **700**, memory **200** receives an RD command with an RCKON attribute set to 1 at a point in time before the times shown in timing diagram **700**. Memory **200** provides a preamble for the RCK signal in response to the RD command. However, after the end of the transfer of data, memory **200** continues to toggle the RCK signal which, in the example of timing diagram **700**, forms an extended interamble period. As shown in FIG. **7**, before a subsequent read command is received, memory **200** receives a WR command (WR, WRA, or WRTR). In this case, RCK logic and state machine **213** decodes the write command and after a write command latency time, stops toggling the RCK signal.

[0056] By providing the capability to start and stop the RCK toggling, memory **200** provides a read clock signal that is a hybrid of a strobe and a clock signal. Memory **200** also provides a mechanism for the memory to suppress outputting the RCK continuously after a clear condition. This capability allows the DLL in the memory controller to stay locked during a streak of read commands, yet to stop toggling and save power in response to a clear condition. Memory controllers re-order commands to improve the efficiency of usage of the memory bus, and group commands of the same type to lower the frequency of bus turn-arounds from reads to writes and from writes to reads. For example, efficiency of bus usage is especially important for discrete GPUs, which users often configure to push the limits of performance. This capability provides several benefits.

[0057] First, it allows the memory controllers to operate more efficiently by simplifying their design that would otherwise be required due to the complexities of the interamble. In particular, the interamble calculations can be simplified or eliminated.

[0058] Second, it preserves signal integrity of the signals that switch when RCK is not used. For example, this mechanism allows the user to avoid the continuous generation of RCK in response to receiving a write command. Thus, the write cycle is more robust, with larger timing margins to capture data in the memory because of less signal interference and cross-talk.

[0059] Third, by suppressing the switching of the RCK signal during streaks of write accesses, it saves the switching power of the high-speed signal switching of an external signal typically driven on a printed circuit board, when it is not needed.

[0060] Fourth, it provides flexibility in implementation because the host processor can set the RCKON flag used in the memory in a variety of ways. For example, the host processor can issue a mode register set command to program an unused or vender-specific RCKON bit in a mode register. It can issue an explicit RCKON command or include an RCKON attribute in a RD command encoding. It can also activate a new, dedicated signal line.
[0061] A memory or portions thereof described herein can be embodied one or more integrated circuits, any of which may be described or represented by a computer accessible data structure in the form of a database or other data structure which can be read by a program and used, directly or indirectly, to fabricate integrated circuits. For example, this data structure may be a behavioral-level description or register-transfer level (RTL) description of the hardware functionality in a high-level design language (HDL) such as Verilog or VHDL. The description may be read by a synthesis tool which may synthesize the description to produce a netlist including a list of gates from a synthesis library. The netlist includes a set of gates that also represent the functionality of the hardware including integrated circuits. The netlist may then be placed and routed to produce a data set describing geometric shapes to be applied to masks. The masks may then be used in various semiconductor fabrication steps to produce the integrated circuits. Alternatively, the database on the computer accessible storage medium may be the netlist (with or without the synthesis library) or the data set, as desired, or Graphic Data System (GDS) II data.
[0062] While particular embodiments have been described, various modifications to these embodiments will be apparent to those skilled in the art. For example, host processors can use various techniques of setting the read clock on attribute (RCKON) such as explicit commands, unused bits in the command encoding, setting a mode register, and the like. While the RCK programming was described in the context of a GDDR memory, another type of circuit, integrated or discrete, can use a clock signal with a hybrid behavior that was described for the RCK signal. The RCK signal can be used in other types of memory as well, including DDR and high-bandwidth (HDM) memory. Moreover, various levels of granularity of preamble and postamble behavior can also be supported. The bits that define the supported RCK mode can be defined in a dedicated mode register, or can be set in various bit positions in different mode registers. These bit positions may have been previously unused and reserved for future use, or dedicated to customer-specific use.
[0063] Accordingly, it is intended by the appended claims to cover all modifications of the disclosed embodiments that fall within the scope of the disclosed embodiments.

Claims

1. A data processor comprising a memory controller, wherein the data processor is configured to control the memory controller to: program at least one mode register of a memory to set a read clock mode to one of a plurality of settings, wherein the plurality of settings includes a read-only mode in which the memory starts a read clock signal in response to a command; and subsequently perform a read cycle by receiving data during the read cycle using the read clock signal.
2. The data processor of claim 1, wherein the plurality of settings of the at least one mode register includes a read clock type setting, wherein the read clock type includes a single ended mode and a differential mode.
3. The data processor of claim 1, wherein the plurality of settings of the at least one mode register includes a static preamble length setting.
4. The data processor of claim 1, wherein the plurality of settings of the at least one mode register includes a high-speed preamble length setting.
5. The data processor of claim 1, wherein the plurality of settings of the at least one mode register includes a low-speed preamble length setting.
6. The data processor of claim 1, wherein the memory controller is configured to selectively

provide a clear condition to the memory when the read clock mode setting is read-only.
7. The data processor of claim 6, wherein the memory controller provides the clear condition by providing an explicit read clock stop command.
8. The data processor of claim 6, wherein the memory controller provides the clear condition by providing a write command.
9. The data processor of claim 8, wherein providing the write command comprises the memory controller providing a write with auto-precharge command.
10. The data processor of claim 6, wherein the memory controller provides the clear condition by providing a mode register set command.
11. The data processor of claim 6, wherein the memory controller provides the clear condition by causing all banks of the memory to be idle.
12. The data processor of claim 6, wherein the memory controller provides the clear condition by sending a training command.
13. The data processor of claim 6, wherein the memory controller provides the clear condition by placing the memory into a power down state.
14. The data processor of claim 1, wherein the data processor is further configured to control the memory controller to: program the at least one mode register of the memory to set a preamble type; and subsequently perform the read cycle by receiving data during the read cycle using the preamble type.
15. The data processor of claim 1, wherein the plurality of settings further includes: an always running mode; and a disabled mode.
16. The data processor of claim 1, wherein the data processor further comprises: a graphics processing unit (GPU) core coupled to the memory controller.
17. The data processor of claim 1, wherein the data processor further comprises: a physical interface circuit coupled to the memory controller and configured to be coupled to the memory.
18. A method, comprising: programming at least one mode register of a memory, the programming comprising setting a read clock mode to one of a plurality of settings, wherein the plurality of settings includes an always running mode and a read-only mode in which the memory starts a read clock signal in response to a command; and subsequently performing a read cycle by a memory controller by receiving data during the read cycle using the read clock signal.
19. The method of claim 18, wherein the plurality of settings of the at least one mode register includes a read clock type setting, wherein the read clock type includes a single ended mode and a differential mode.
20. The method of claim 18, wherein the plurality of settings of the at least one mode register includes a static preamble length setting.
21. The method of claim 18, wherein the plurality of settings of the at least one mode register includes a high-speed preamble length setting.
22. The method of claim 18, wherein the plurality of settings of the at least one mode register includes a low-speed preamble length setting.
23. The method of claim 18, further comprising selectively providing a clear condition to the memory when the read clock mode setting is read-only.
24. The method of claim 23, wherein selectively providing the clear condition comprises providing an explicit read clock stop command by the memory controller.
25. The method of claim 24, wherein selectively providing the clear condition comprises providing a write command by the memory controller.
26. The method of claim 25, wherein selectively providing the write command comprises providing a write with auto-precharge command by the memory controller.
27. The method of claim 23, wherein selectively providing the clear condition comprises providing a mode register set command by the memory controller.
28. The method of claim 23, wherein selectively providing the clear condition comprises causing

all banks of the memory to be idle by the memory controller.

29. The method of claim 23, wherein selectively providing the clear condition comprises sending a training command by the memory controller.

30. The method of claim 23, wherein selectively providing the clear condition comprises placing the memory into a power down state by the memory controller.

31. The method of claim 18, further comprising: programming the at least one mode register of the memory to set a preamble type; and subsequently performing the read cycle according to the preamble type.

32. The method of claim 18, further comprising: generating memory access requests by a graphics processing unit (GPU) core coupled to the memory controller.

33. The method of claim 18, further comprising: coupling the memory controller to the memory using a physical interface circuit.