

DDR2SOFT DDR2 Memory Controller VHDL SOURCE CODE OVERVIEW

Overview

The DDR2SOFT is a generic DDR2 memory controller core (including the [VHDL source code](#)) designed to interface DDR2 memory modules and memory ICs with low-cost FPGAs.

This generic code does not use any hardware-based memory controller block (MCB). It is therefore slower but free of key MCB constraints (data bus width, number of instantiations). In particular, this component allows Xilinx Spartan-6 FPGAs to interface with 64-bit wide DDR2 DIMM/SODIMM, for streaming throughputs up to 16 Gbps.

The binary (.ngc) component is free for use on ComBlock FPGA development platforms. The VHDL source code is sold separately.

The component's very efficient implementation makes it suitable for multiple instantiations within a small FPGA.

The object-oriented code structure provides two levels of abstraction:

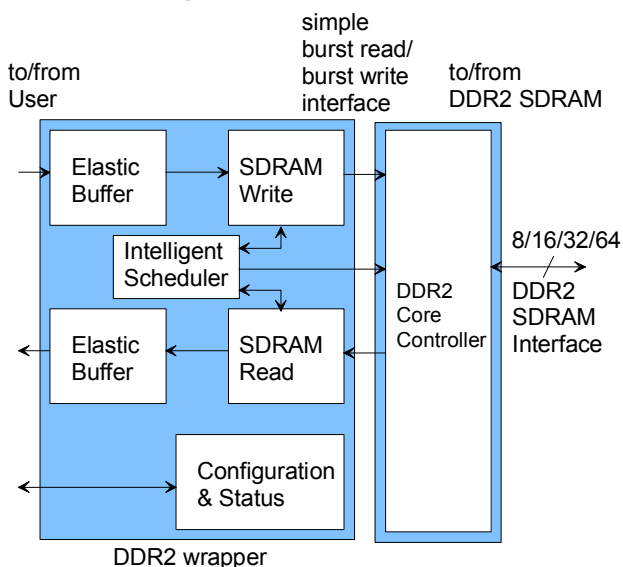
- The **DDR2 core controller** makes many low-level tasks invisible to the user: refresh, initialization and most timing. Its simple interface accepts burst read and burst write commands.
- The **DDR2 wrapper** provides another layer of abstraction by optimizing memory access for maximum throughput or minimum latency. Users can interface with their own clock, different than the DDR2 controller clock.

Key features include:

- Concurrent read/write at the user interface.
- Independent clocking for user data and DDR2.
- Optimized for efficient data streaming:

- Address auto-increment
- Start/Stop address boundaries
- Continuous/Single-shot modes
- Compliant with DDR2 SDRAM standard JESD79-2F (Nov. 2009)
- Support for low-power self-refresh mode.
- Automatic banks management (multiple banks can be opened simultaneously).
- Automatic calibration to cancel the round-trip delay FPGA – SDRAM – FPGA and sample the read data at the center of the eye diagram.

Block Diagram



Speed

FPGA	Clock (max)	Read/Write streaming throughput
Spartan-6 -2	130 MHz = 260 MTransfers/s	16-bit IC: 4 Gbit/s 64-bit SODIMM 16 Gbits/s

Target Hardware

The code is written in generic VHDL so that it can be ported to a variety of FPGAs. The code was developed and tested on a Xilinx Spartan-6 XC6SLX FPGA.

It can be easily ported to any Xilinx Virtex-5, Virtex-6, Spartan-6 FPGAs and other FPGAs capable of running at or above the minimum DDR2 clock frequency of 125 MHz.

Please note that the target FPGA must be capable of

- Creating programmable input delays at the SDRAM interface.
- Generating a low-jitter SDRAM clock using a true PLL (DCM are not recommended), or using an external low-jitter clock.

Device Utilization Summary

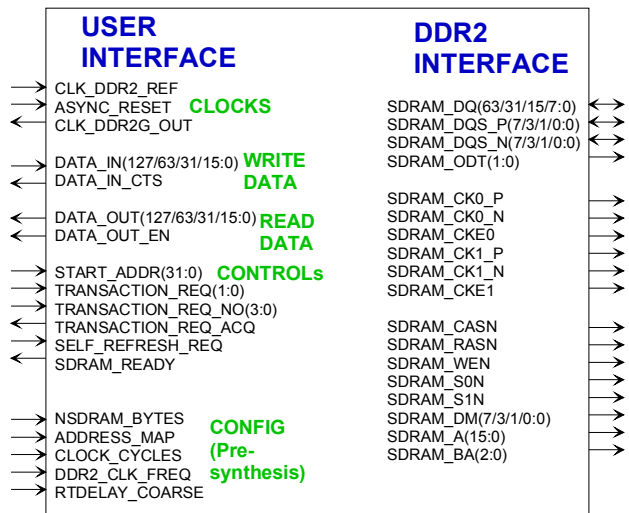
Device: Xilinx Spartan-6, Core DDR2 SDRAM controller only

FPGA resources	16-bit DDR2 data width	64-bit DDR2 DIMM /SODIMM data width
LUTs	665	820
Flip Flops (in slices)	350	741
RAMB16BWERs	0	0
DSP48A1s	0	0
GCLKs	4	4
DCMs	1	1

Device: Xilinx Spartan-6, Complete controller, including wrapper.

FPGA resources	16-bit DDR2 data width	64-bit DDR2 DIMM/SODIMM data width
Flip Flops		
LUTs		
RAMB16BWERs	2	8
DSP48A1s	0	0
GCLKs	2	2
DCMs	0	0

Core Controller Interface



On the controller DDR2 interface side, all signals connect directly to the external DDR2 SDRAM.

The controller user interface side comprises four signal groups:

- Clocks:** The user sets the DDR2 SDRAM speed by providing a reference frequency as CLK_DDR2_REF. The controller internally generates a global clock CLK_DDR2G_OUT for use by the user code.
- Controls for the two elementary transactions: burst read and burst write.** Each burst comprises 4 words, the word size being determined by the DDR2 SDRAM data width (8/16/32 or 64 bits).

User data is transferred two-words at a time to match the DDR2 SDRAM throughput.

- Configuration parameters are set at the time of VHDL synthesis (i.e. cannot be modified at run-time).**

All user-side signals are clock synchronous with the user-selected clock CLK_DDR2G_OUT.

The best way to generate the DDR2 SDRAM reference clock is by using a PLL as illustrated in the *clkgen_pll.vhd* example.

Core Controller Transactions

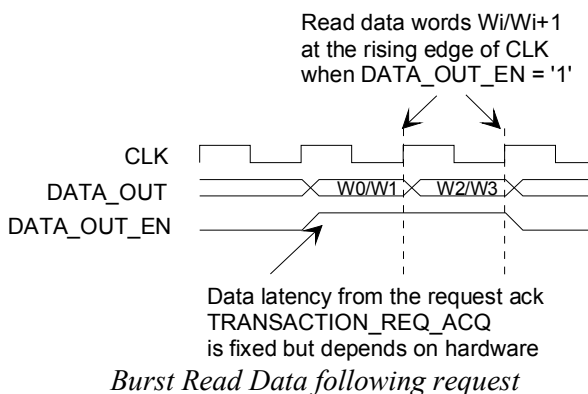
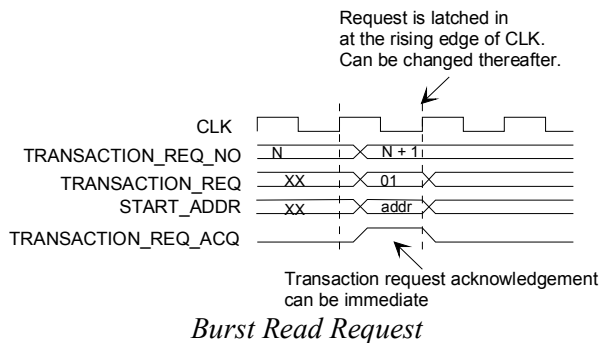
Burst Read

The user declares its intent to read a 4-word burst by simultaneously

- Changing the transaction request number TRANSACTION_REQ_NO
- Setting the transaction request to “burst read” TRANSACTION_REQ = “01”
- Setting the burst start address START_ADDR, expressed as integer multiple of words, not bytes.

The controller will send a 1-CLK wide pulse acknowledgement TRANSACTION_REQ_ACK as soon as it can process the transaction request. The acknowledgement indicates that this transaction was accepted and processing has just started. The user can now send the next request (even while the current transaction is in progress).

Note: When possible, the controller will send the acknowledgement pulse very quickly, within the same clock period as the user request.



The 4 words read from the SDRAM are returned, packed two words at a time in DATA_OUT while DATA_OUT_EN = '1'.

Burst Write

The user declares its intent to write a 4-word burst by simultaneously

- Changing the transaction request number TRANSACTION_REQ_NO
- Setting the transaction request to “burst write” TRANSACTION_REQ = “10”
- Setting the burst start address START_ADDR, expressed as integer multiple of words, not bytes.

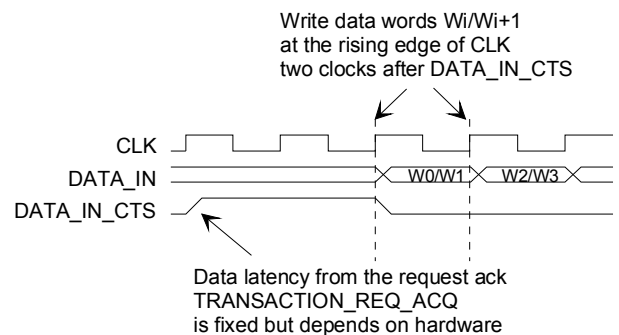
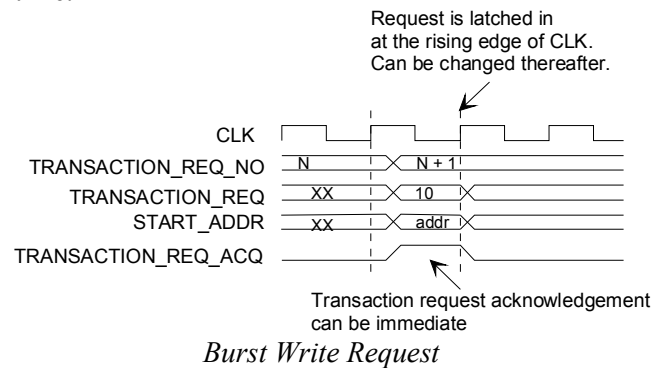
The controller will send a 1-CLK wide pulse acknowledgement TRANSACTION_REQ_ACK as soon as it can process the transaction request. The acknowledgement indicates that this transaction was accepted and processing has just started.

Following the transaction acknowledgement, the controller will tell the user when to provide the data by setting DATA_IN_CTS to '1' exactly two clocks before the data is due at the DATA_IN input.

IMPORTANT: input data must be provided exactly at the right time: 2 CLKs after the DATA_IN_CTS pulse.

This interface is perfect for interfacing with an external dual-port block RAM (since it takes up to 2 CLKs to read data from a RAMB).

The user must pack DATA_IN with two words at a time.



Configuration

The user can set the following controls [at run-time](#):

<i>DDR2 Controller Configuration</i>	<i>Description</i>
SELF_REFRESH_REQ	<p>1 = User requests the controller to enter the Self refresh mode to reduce power consumption. The SDRAM data is retained.</p> <p>This low-power mode is typically used for long-term idle periods (for example when the memory is in standby mode). Exiting from this mode takes 200 CLKs.</p> <p>0 = Normal mode. Read and Write is allowed.</p>

The user can set the following controls [prior to synthesis](#) in the “generic” section of the DDR2 component instantiation.

<i>Physical Configuration</i>	<i>Description</i>
NSDRAM_BYTES	<p>DDR2 interface data width, expressed in bytes. Also defines the user interface data width (which is always twice as large).</p> <p>1 = 8-bit word (IC)</p> <p>2 = 16-bit word (IC, e.g. COM-1600)</p> <p>4 = 32-bit word (ICs or partial DIMM/SODIMM)</p> <p>8 = 64-bit word (DIMM/SODIMM, e.g. COM-1500)</p>
ADDRESS_MAP	<p>The logical to physical address mapping can vary depending on the hardware. See the DDR2 SDRAM hardware specifications.</p> <p>The most common mappings are described below:</p> <p>00000: 1Gb x8 IC, column A[9:0], 8 banks BA[2:0], 1 rank S[1:0], row A[13:0]</p> <p>00001: 1Gb x16 IC, column A[9:0], 8 banks BA[2:0], 1 rank S[1:0], row A[12:0]</p> <p>00010: 2Gb x8 IC, column A[9:0], 8 banks BA[2:0], 1 rank S[1:0], row A[14:0]</p> <p>00011: 2Gb x16 IC, column A[9:0], 8 banks BA[2:0], 1 rank S[1:0], row A[13:0]</p> <p>0xxxx: other ICs, reserved for future use</p> <p>10000: 1GB DIMM/SODIMM, column A[9:0], 8 banks BA[2:0], 2</p>

	<p>ranks S[1:0], row A[12:0]</p> <p>10001: 2GB DIMM/SODIMM, column A[9:0], 8 banks BA[2:0], 2 ranks S[1:0], row A[13:0]</p> <p>10010: 4GB DIMM/SODIMM, column A[9:0], 8 banks BA[2:0], 2 ranks S[1:0], row A[14:0]</p> <p>1xxxx: other DIMM/SODIMMs, reserved for future use</p>
CLOCK_CYCLES	<p>CL - tRCD - tRP (CAS latency) - (RAS-CAS-Delay time) - (RAS precharge time) See DDR2 SDRAM hardware specifications.</p> <p>3 = 3-3-3 DDR2-400</p> <p>4 = 4-4-4 DDR2-533</p> <p>5 = 5-5-5 DDR2-667</p> <p>6 = 6-6-6 DDR2-800</p>
RTDELAY_COARSE	<p>Round-trip delay in the path FPGA -> SDRAM -> FPGA, rounded up to the nearest integer multiple of clock periods.</p> <p>Example Spartan-6 125 MHz: Output delay for SDRAM_CKx: 9ns Input delay for SDRAM_DQSy: 4ns RTDELAY_COARSE = 13ns*2*125MHz = 3.25 rounded up to 4.</p>

In addition, less frequently used timing parameters can be customized as VHDL constants in the `--// TIMERS` section of the source code instead of using the default worst case. For example, the refresh to command timer `tRFC` can be shortened in the case of smaller memories. Another example is the adjusting of the periodic refresh time `tREFI` for extended temperature (85-95C) operation. The list of customizable timers, using the JEDEC standard notation, is as follows:
`tRFC`, `tREFI`, `tWR`

Other DDR2 configuration parameters are fixed in the source code:

- Normal mode
- Burst length `BL = 4`
- ODT enabled, 75 Ohm `Rtt`
- `DQS#` enable
- `RDQS` disable
- Output buffer enabled
- Optional duty cycle corrector disabled
- `OCD` calibration is not used

Wrapper Interface

Exclusions

This software does not support the following features:

- Serial Presence-Detect (best handled by a microprocessor)

Software Licensing

The DDR2SOFT is supplied under the following key licensing terms:

1. A nonexclusive, nontransferable license to use the VHDL source code internally, and
2. An unlimited, royalty-free, nonexclusive transferable license to make and use products incorporating the licensed materials, solely in bitstream format, on a worldwide basis.

The complete VHDL/IP Software License Agreement can be downloaded from <http://www.comblock.com/download/softwarelicense.pdf>

Reference documents

[1] DDR2 SDRAM Specification, JEDEC Standard No. 79-2F

[2] ComBlock COM-1600 FPGA + ARM + USB2.0+ DDR2 + NAND development platform www.comblock.com/com1600.html

[3] COM-1600 development platform schematics

Configuration Management

The current software revision is 1.

[a] VHDL source code in directory
DDR2_rev\src

[b] Xilinx ISE project files
DDR2_rev\DDR2_ISE114.xise
DDR2_rev\DDR2_ISE122.xise

[c] .ucf constraint file
DDR2_rev\src\DDR2.ucf

[d] VHDL testbenches in directory
DDR2_rev\sim

[e] .ngc binary for use in conjunction with the
COM-1600 module
DDR2_rev\bin\DDR2.ngc

where *rev* is the current revision number.

VHDL development environment

The VHDL software was developed using the following development environment:

- Xilinx ISE 11.4 with XST as synthesis tool
- Xilinx ISE 12.2 with XST as synthesis tool
- Xilinx ISE Isim as VHDL simulation tool

Important: The FPGA – SDRAM interface is time-critical. It is therefore important to ask the tools (Synthesis, Map) to pack the flip-flops in IOBs.

Ready-to-use Hardware

The binary component (.ngc) is freely available for use on the following ComBlock hardware modules:

- COM-1600 FPGA + ARM + DDR2 + NAND + USB2 development platform
- COM-1500 FPGA + DDR2 SODIMM socket + ARM + NAND + USB2 development platform

See the following code templates:
TBD

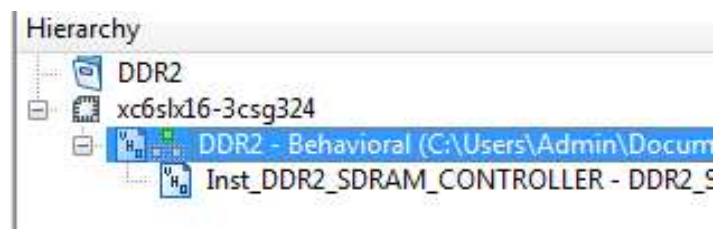
The schematics are available in this CD.

Xilinx-specific code

The VHDL source code is written in generic VHDL with few Xilinx primitives. No Xilinx CORE is used. The Xilinx primitives are:

- IBUF
- IBUFG
- BUFG (global clocks)
- IDDR2
- ODDR2
- IODELAY2 to delay the read data and data strobe
- IOBUFDS differential input/output buffer
- Dual-port RAM block: RAMB16BWERs

Top-Level VHDL hierarchy



The design consists primarily of two components:

- A core controller component (*DDR2_SDRAM_CONTROLLER.vhd*) and
- A wrapper (*DDR2_WRAPPER.vhd*)

The code is stored with one, and only one, component per file.

The core component *DDR2_SDRAM_CONTROLLER.vhd* operates at the DDR2 SDRAM clock. It performs the SDRAM power-up initialization, refresh, low-power self-refresh mode, round-trip delay fine calibration, active banks/rows management and burst read/write transactions. The user can request a burst read or a burst write transaction. All transactions transfer exactly $4 * NSDRAM_BYTES$ bytes. The core component's usage is described in the operation section above.

An optional wrapper (*DDR2_WRAPPER.vhd*) comprises the transmit and receive elastic buffers, user-clock to memory-clock domains translation, read/write sequencing and data segmentation into bursts.

DDR2_SDRAM_CONTROLLER_TEST.vhd is an optional component aimed at hardware verification such as board post-assembly tests. The components

writes and reads-back pseudo-random data at addresses within the specified address range.

Additional components are used for simulation purposes:

- *tbDDR2_SDRAM_CONTROLLER_TEST.v* *hd* is a simulation testbench for the test component. It simulates propagation delay between external SDRAM and FPGA.
- *DDR2_MODULE.v* is a SODIMM Verilog simulation module (from Micron). Verilog 'define statements at the top of file help select the memory type.

Calibration

The FPGA - DDR2 SDRAM interface timing can be quite challenging at higher clock speeds. The core controller code is written with two timing principles:

- (a) FPGA to DDR2 SDRAM direction: signals and synchronous clocks propagate in the same direction, with little timing variation through the IOBs. No problem is expected here.
- (b) DDR2 SDRAM to FPGA direction: the controller automatically calibrates out the round-trip delays so as to sample the input signals in the FPGA at the optimum time.

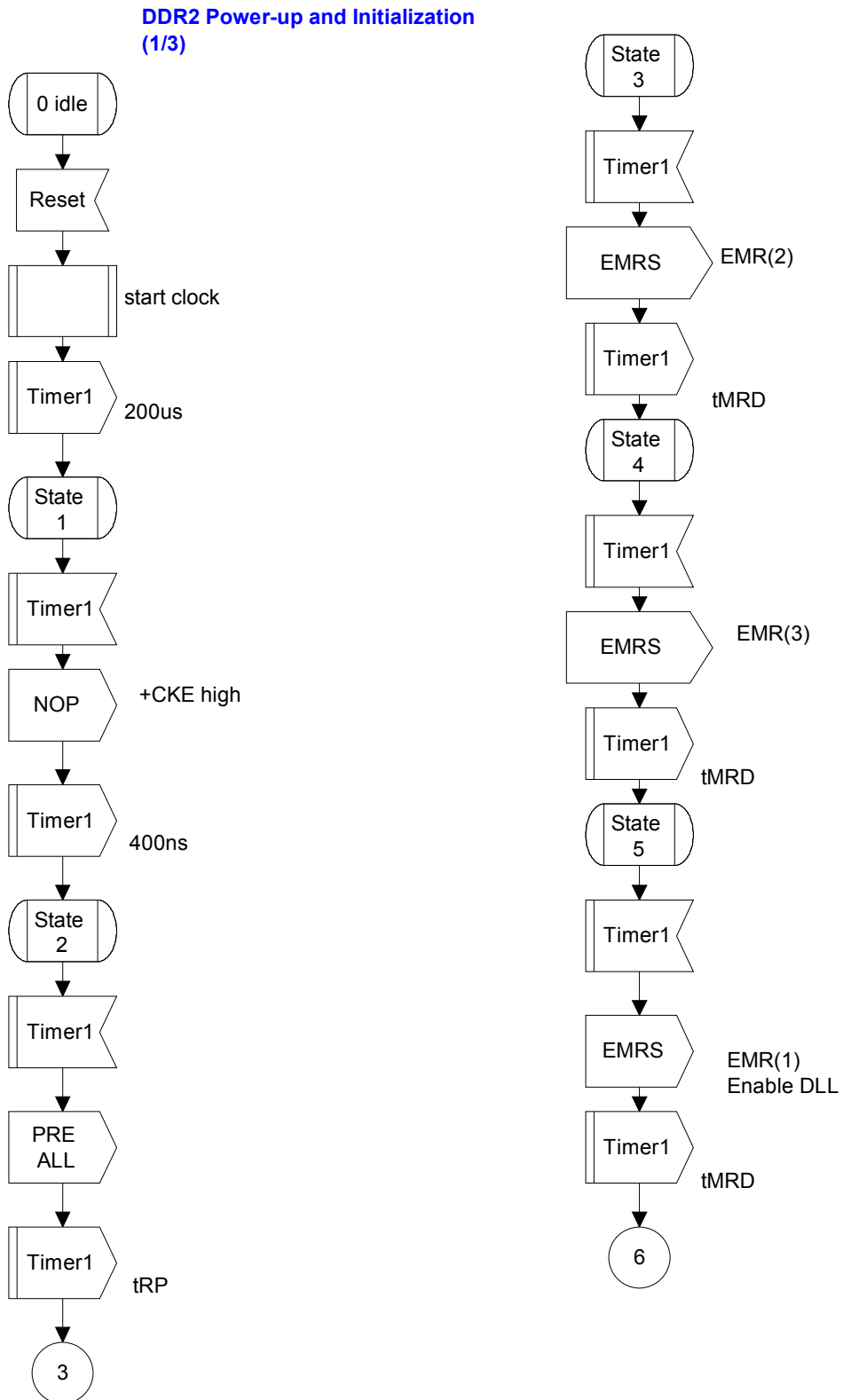
The automatic calibration is only capable of performing fine delay compensation in the range 0 to $\frac{1}{2}$ clock period. It is up to the user to specify the coarse delay `RTDELAY_COARSE`, expressed as an integer number of $\frac{1}{2}$ clock periods rounded up. Since the signal round-trip delay (FPGA->SDRAM->FPGA) is design-dependent, the user must find out the round-trip delay for its design.

The fine delay (fraction of a $\frac{1}{2}$ clock period) is removed automatically by the built-in calibration circuit.

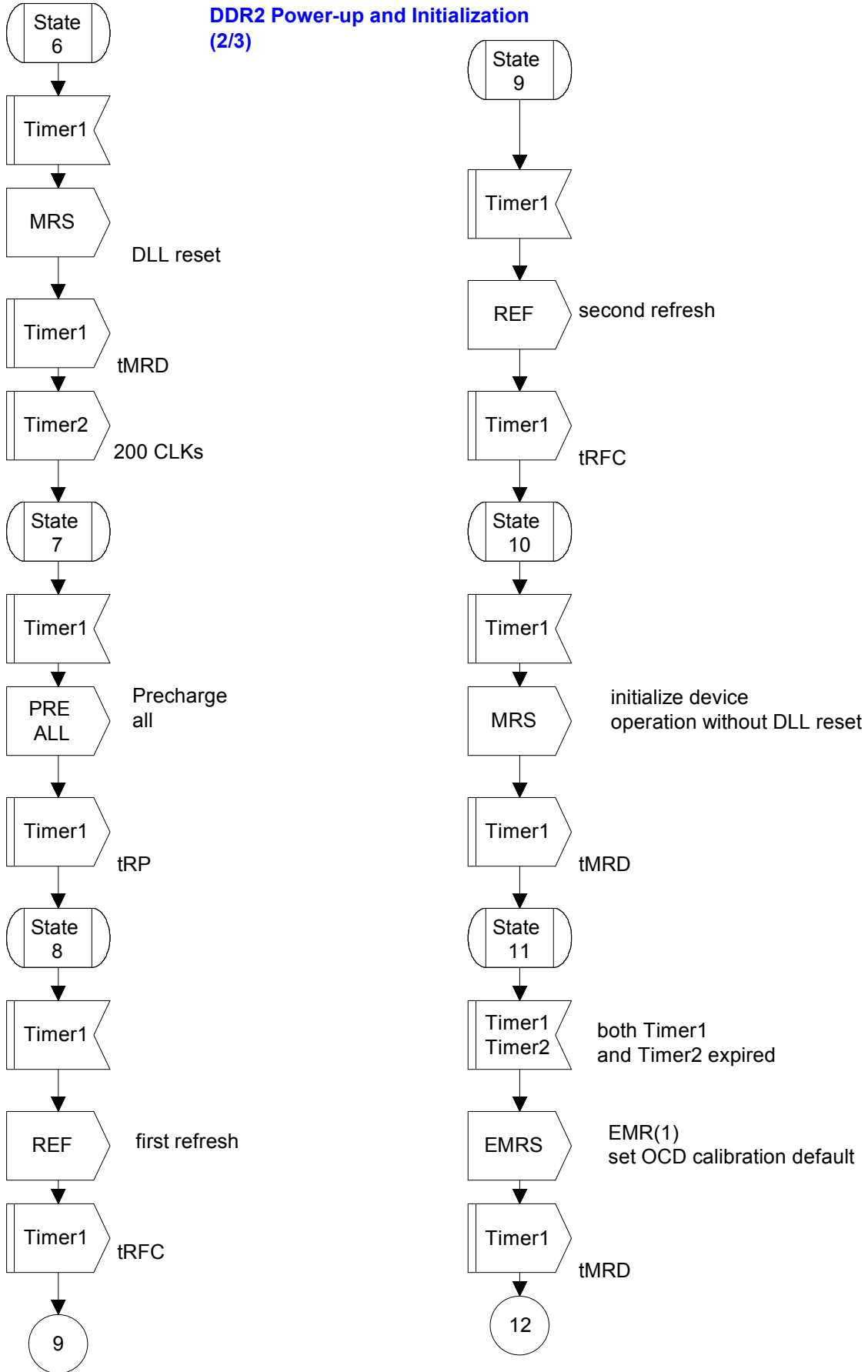
Calibration is triggered automatically at power up or after reset. Once the calibration is complete, the controller will set the `SDRAM_READY` flag high. The controller ignores user-generated Read/Write transaction requests issued before the SDRAM is ready.

State Machine

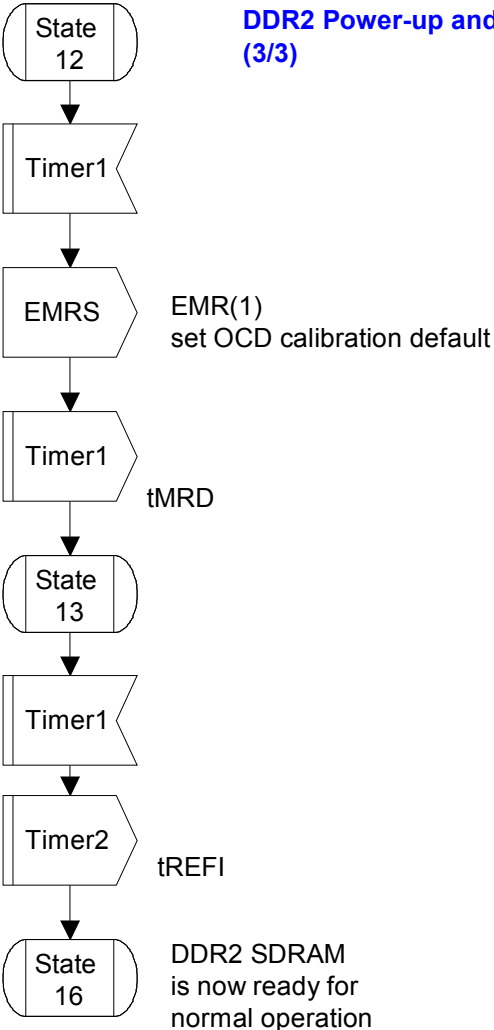
The DDR2 memory controller state machine is described by the SDL flowchart below:

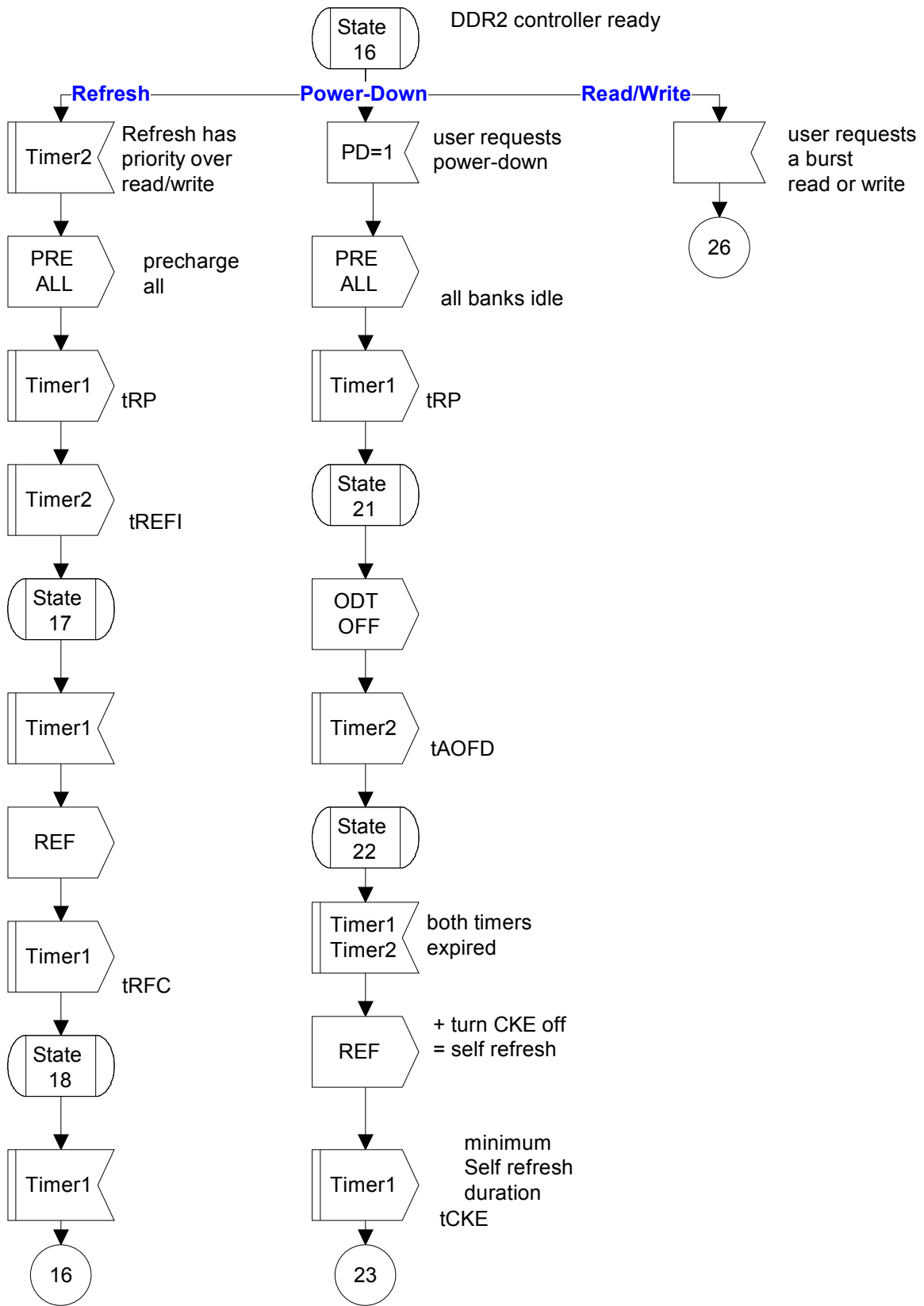


**DDR2 Power-up and Initialization
(2/3)**



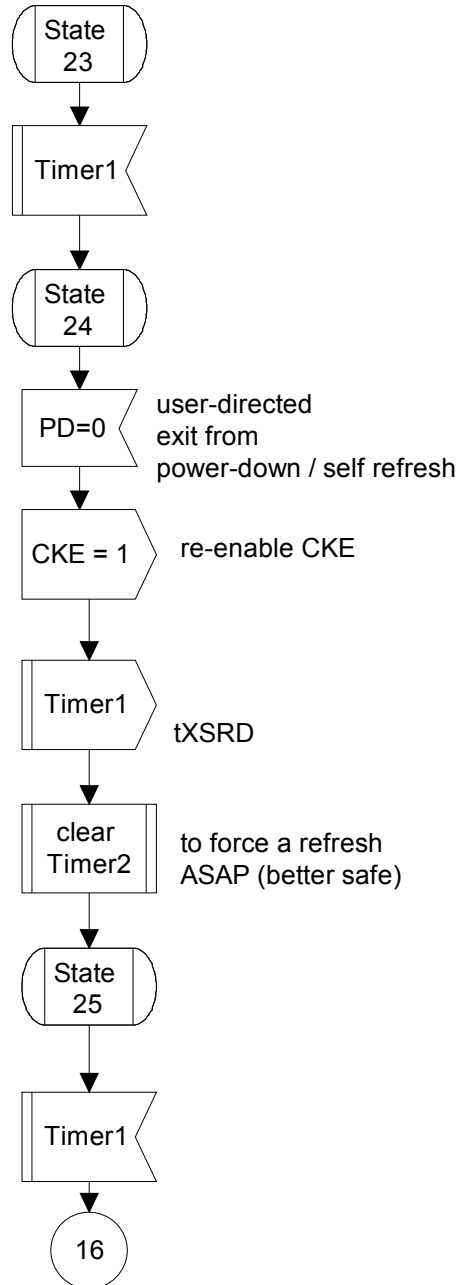
**DDR2 Power-up and Initialization
(3/3)**



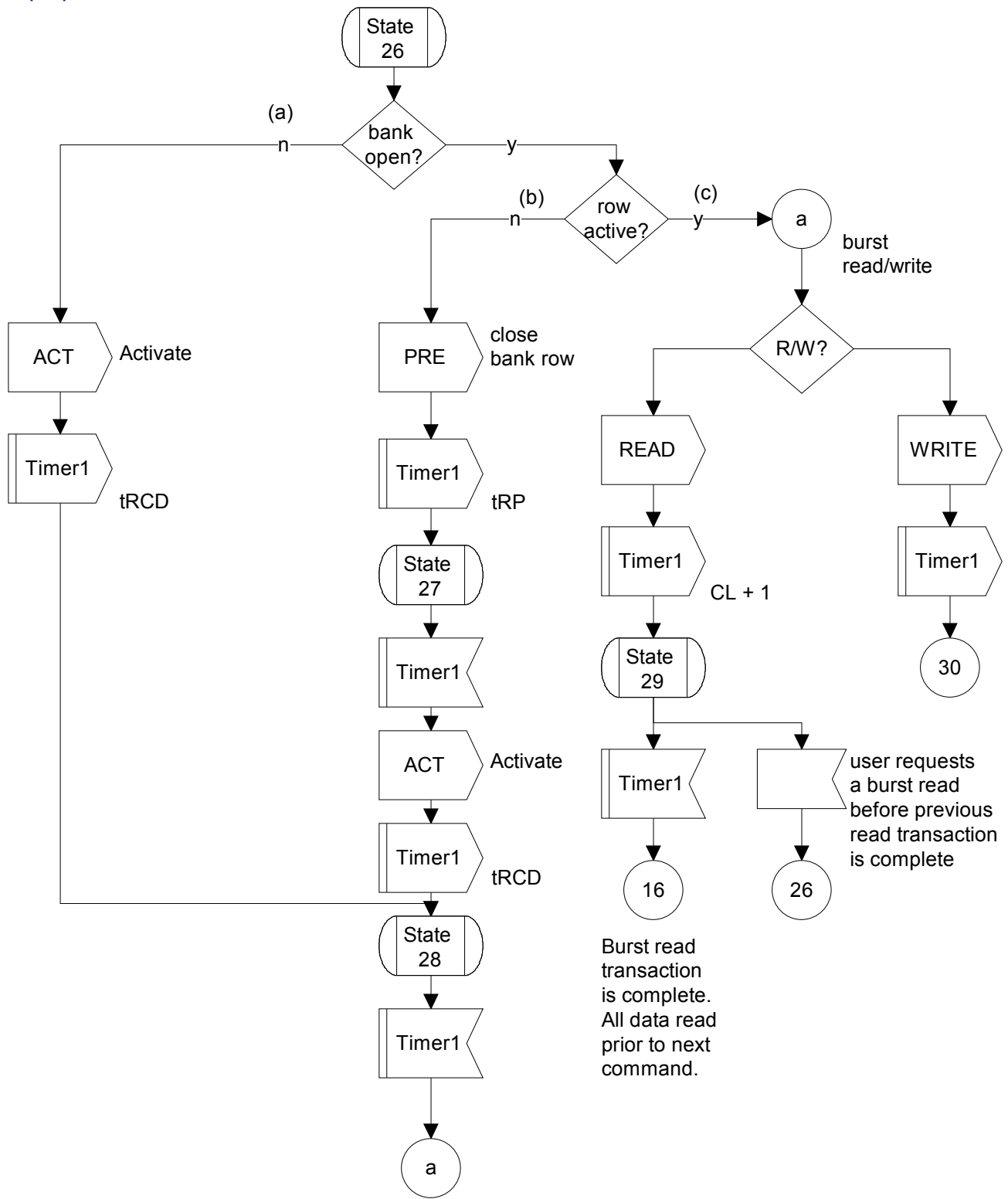


**DDR2 Refresh / Read / Write / Power-Down
(2/3)**

Exit Self-Refresh



DDR2 Burst Read / Write, Banks management
(3/3)



Simulations

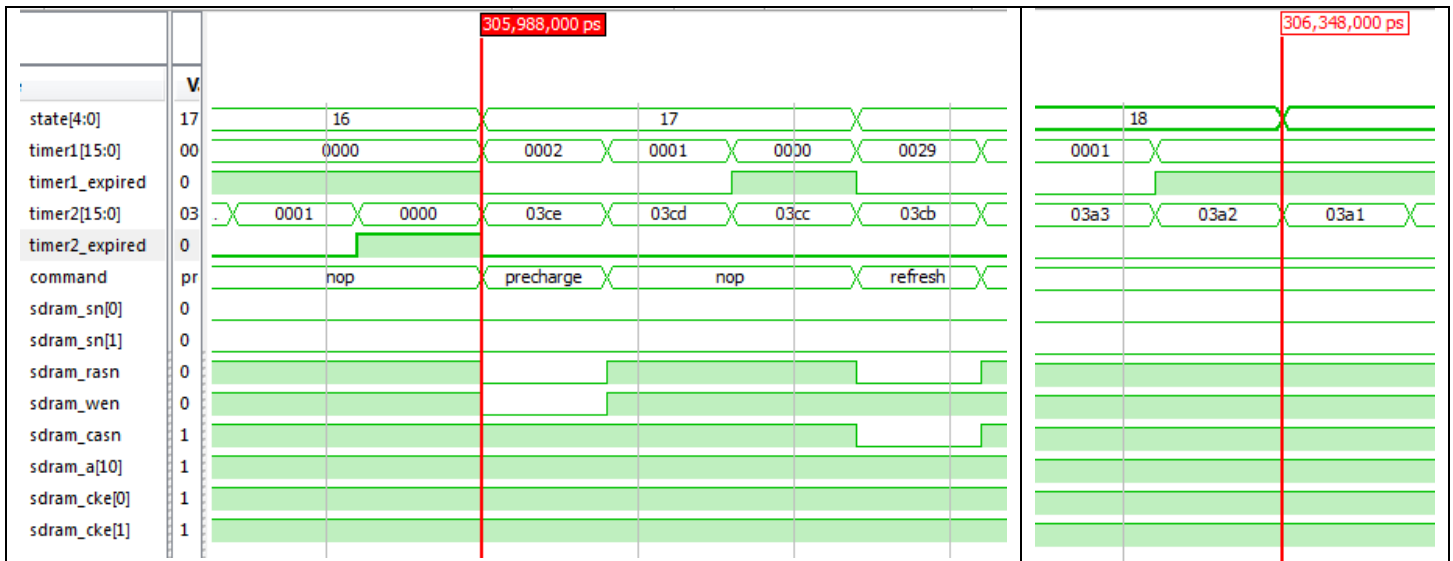
All simulations below are conducted in the following environment: DDR2-400, 200 MHz clock, burst length = 4

Auto-Refresh

Timer 2 schedules a refresh cycle once every $tREFI = 7.8\mu s$.

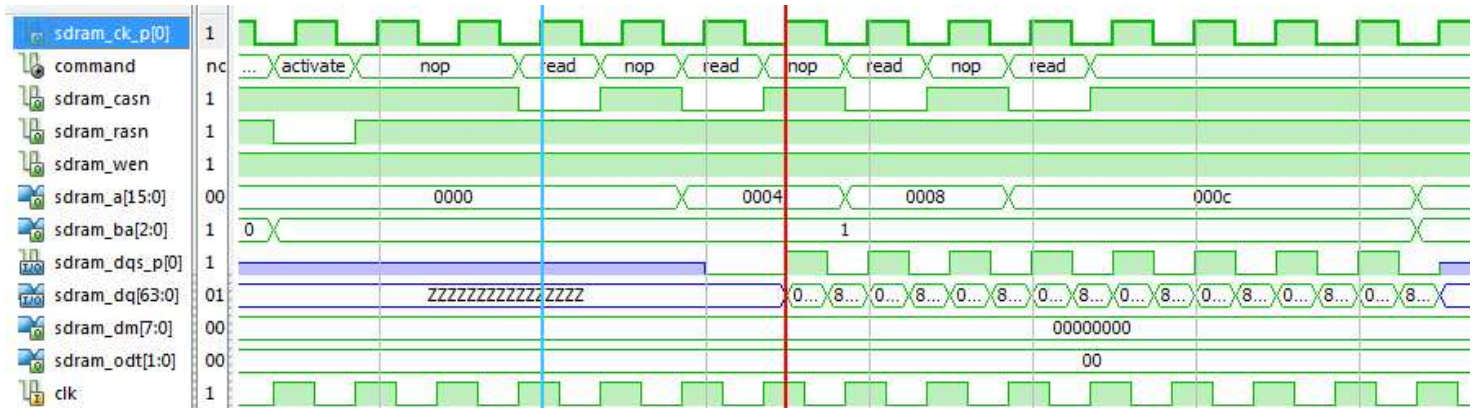


Refresh transaction details

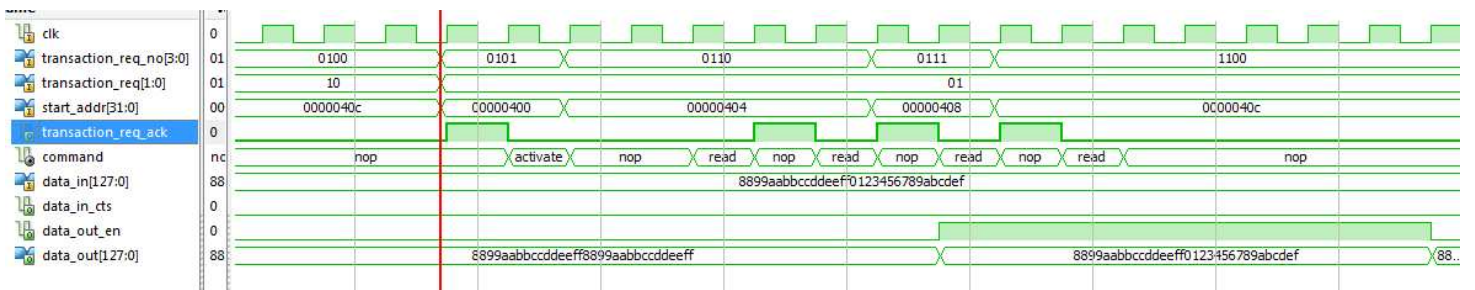


Burst read

4 successive burst read requests, burst length = 4, CL = 3



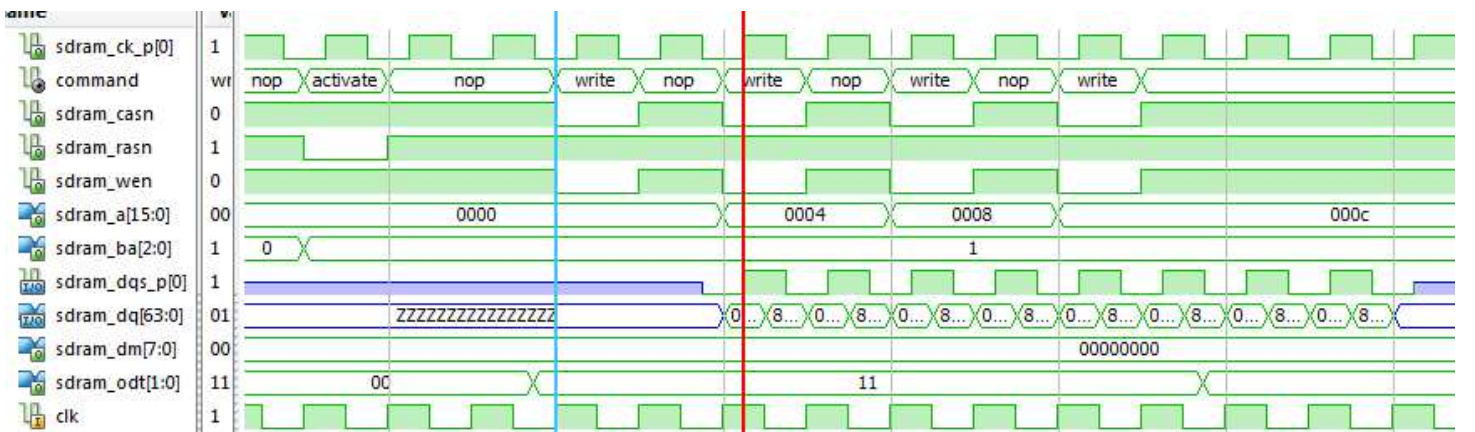
Burst read, DDR2 SDRAM side



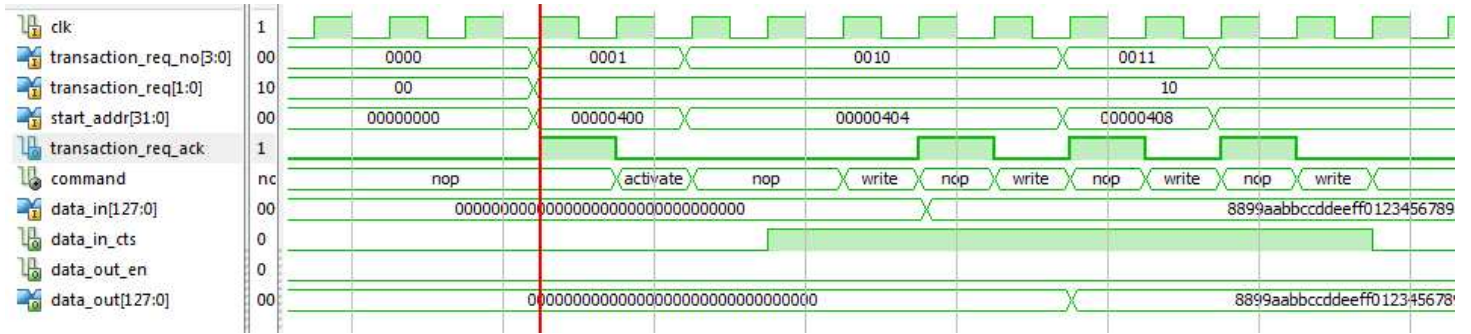
Burst read, user side.

Burst write

4 successive burst write requests, burst length = 4, CL = 3



Burst write, DDR2 SDRAM side



Burst write, user side

ComBlock Compatibility List

FPGA development platform	
COM-1600	FPGA + ARM + DDR2 + USB2 + NAND development platform
COM-1500	FPGA + DDR2 SODIMM socket + ARM + USB2 + NAND development platform

ComBlock Ordering Information

DDR2SOFT DDR2 MEMORY CONTROLLER, VHDL SOURCE CODE

Contact Information

MSS • 18221-A Flower Hill Way •
Gaithersburg, Maryland 20879 • U.S.A.
Telephone: (240) 631-1111
Facsimile: (240) 631-1676
E-mail: info@comblock.com