SDRAM CONTROLLER
Specification

Author: Dinesh Annayya
dinesha@opencores.org

Rev. 0.2
February 9, 2012

This page has been intentionally left blank

Revision History

Rev.	Date	Author	Description
0.0	01/17/2012	Dinesh Annayya	First draft release
0.1	01/30/2012	Dinesh Annayya	➢ Wish Bone Interface is added ➢ 8 Bit SDRAM Support is added ➢ Test bench is automated for Stand-alone SDRAM controller and Integrated SDRAM controller with wish bone interfaces
0.2	02/07/2012	Dinesh Annayya	➢ Frequently asked Section is added ➢ Block Diagram are upgraded

Contents

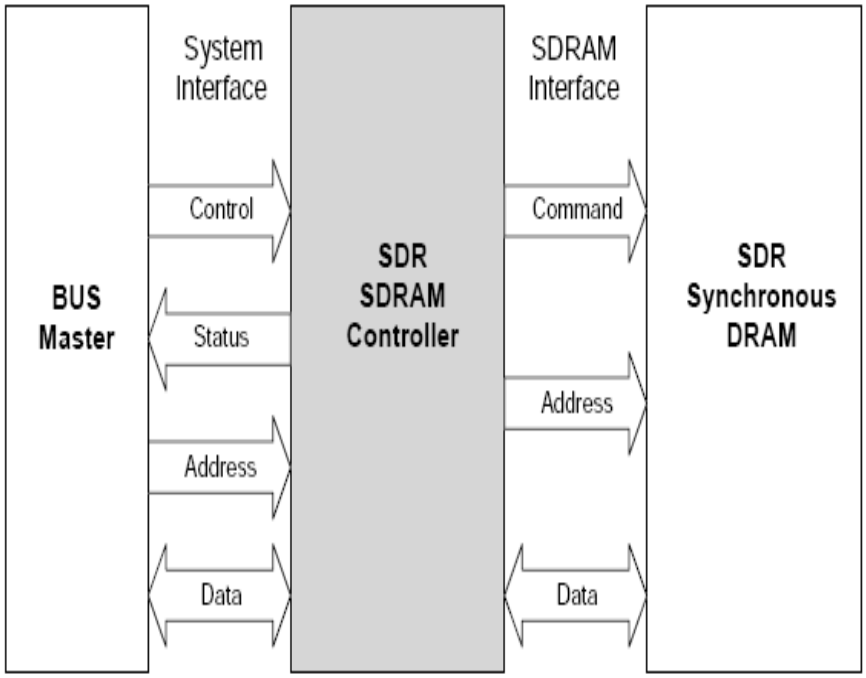
SDRAM CONTROLLER Specification 0
1 Introduction 2
1.1 FEATURES 3
2. IO ports 4
2.1 Core Parameters 4
2.2 WISHBONE interface signals 4
2.3 Application interface signals 4
2.4 SDRAM External connections 5
3 Configuration 6
4 SDRAM Overview 6
5 Functional Description 9
5. 1 Wish Bone Bus Handler: 10
5.2 SDRAM Controller 10
5.3 SDRAM Interface 12
6. SDRAM Data Path 16
7 Simulation 17
8 WAVEFORM 20
Appendix A : Timing 23
A.1 SDRAM Output signals parameter definition 23
A2. SDRAM Input signals parameter definition 23
Appendix A : Frequently Asked Q 24

1 Introduction

Synchronous DRAM (SDRAM) has become a mainstream memory of choice in embedded system memory design. For high-end applications using processors the interface to the SDRAM is supported by the processor's built-in peripheral module. However, for other applications, the system designer must design a controller to provide proper commands for SDRAM initialization, read/write accesses and memory refresh.

This SDRAM controller reference design, located between the SDRAM and the bus master, reduces the user's effort to deal with the SDRAM command interface by providing a simple generic system interface to the bus master. Figure 1 shows the relationship of the controller between the bus master and SDRAM. The bus master can be either a microprocessor or a user's proprietary module interface.

Figure 1. SDR SDRAM Controller System



1.1 FEATURES

- 8/16/32 Configurable SDRAM data width
- Support asynchronous application layer and SDRAM layer
- Wish bone compatible application layer
- Programmable column address
- Support for industry-standard SDRAM devices and modules
- Supports all standard SDRAM functions
- Fully Synchronous; All signals registered on positive edge of system clock
- One chip-select signals
- Support SDRAM with four bank
- Programmable CAS latency
- Data mask signals for partial write operations
- Bank management architecture, which minimizes latency
- Automatic controlled refresh
- Static synchronous design
- Fully synthesizable

2. IO ports

2.1 Core Parameters

Parameter	Type	Default	Description
SDR_DW	Bit	16	SDRAM DATA Width Selection: 16 – 16 Bit SDRAM Mode 32 – 32 Bit SDRAM Mode
SDR_BW	Bit	2	SDRAM BYTE Width Selection 2 – 16 Bit SDRAM Mode 4 – 32 Bit SDRAM Mode

2.2 WISHBONE interface signals

Port	Width	Direction	Description
wb_clk_i	1	Input	Master clock
wb_rst_i	1	Input	Synchronous reset, active high
wb_adr_i	32	Input	Lower address bits
wb_dat_i	32	Input	Data towards the core
wb_dat_o	32	Output	Data from the core
wb_sel_i	4	Input	Write Data Valid
wb_we_i	1	Input	Write enable input
wb_stb_i	1	Input	Strobe signal/Core select input
wb_cyc_i	1	Input	Valid bus cycle input
wb_ack_o	1	Output	Bus cycle acknowledge output

2.3 Application interface signals

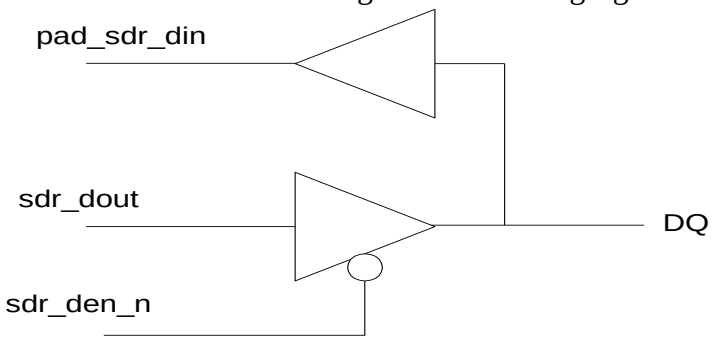
Port	Width	Direction	Description
app_req	1	Input	Application Request
app_req_addr	30	Input	Address
app_req_addr_mask	29	Input	Address Mask
app_req_wr_n	1	Input	0 - Write, 1 – Read
app_req_wrap	1	Input	Address Wrap
app_req_ack	1	Output	Application Request Ack
sdr_core_busy_n	1	Output	SDRAM Controller Busy Indication, 0 - busy, 1 - free
app_wr_data	32	Input	Write Data
app_wr_next	1	Input	Next Write Data Request

_req			
app_wr_en_n	4	Output	Byte wise write Enable, Active low
app_rd_data	32	Input	Read Data
app_rd_valid	1	Output	Read Valid

2.4 SDRAM External connections

Port	Width	Direction	Description
sdr_cke	1	Output	SDRAM clock enable
sdr_cs_n	1	Output	SDRAM command inputs CS#
sdr_ras_n	1	Output	SDRAM command inputs RAS#
sdr_cas_n	1	Output	SDRAM command inputs CAS#
sdr_we_n	1	Output	SDRAM command inputs WE#
sdr_dqm	2/4	Output	SDRAM data bus mask
sdr_ba	2	Output	SDRAM bank address
sdr_addr	12	Output	SDRAM address bus
pad_sdr_din	16/32	Input	SDRAM Data Inut
sdr_dout	16/32	Output	SDRAM Data Output
sdr_den_n	16/32	Output	SDRAM Data Output enable

The tri-state buffers for the DQ lines must be added at a higher hierarchical level. Connections should be made according to the following figure:



Verilog code for 32 BIT SDRAM:

```
assign Dq[7:0]    = (sdr_den_n[0] == 1'b0) ? sdr_dout[7:0]   : 8'hZZ;
assign Dq[15:8]   = (sdr_den_n[1] == 1'b0) ? sdr_dout[15:8]  : 8'hZZ;
assign Dq[23:16]  = (sdr_den_n[2] == 1'b0) ? sdr_dout[23:16] : 8'hZZ;
assign Dq[31:24]  = (sdr_den_n[3] == 1'b0) ? sdr_dout[31:24] : 8'hZZ;
```

Verilog code for 16 BIT SDRAM:

```
assign Dq[7:0]    = (sdr_den_n[0] == 1'b0) ? sdr_dout[7:0]   : 8'hZZ;
assign Dq[15:8]   = (sdr_den_n[1] == 1'b0) ? sdr_dout[15:8]  : 8'hZZ;
```

3 Configuration

Port	Width	Description
cfg_sdr_en	1	SDRAM Controller Enable
cfg_colbits	2	SDRAM Column Bit 00 — 8 Bit 01 — 9 Bits 10 — 10 Bits 11 — 11 Bits
cfg_sdr_mode_reg	12	SDRAM Mode Register
cfg_sdr_tras_d	4	SDRAM active to precharge, specified in clocks
cfg_sdr_trp_d	4	SDRAM precharge command period (tRP), specified in clocks.
cfg_sdr_trcd_d	4	SDRAM active to read or write delay (tRCD), specified in clocks.
cfg_sdr_cas	3	SDRAM CAS latency, specified in clocks
cfg_sdr_trcar_d	4	SDRAM active to active / auto-refresh command period (tRC), specified in clocks.
cfg_sdr_twr_d	4	SDRAM write recovery time (tWR), specified in clocks
cfg_sdr_rfsh	12	Period between auto-refresh commands issued by the controller, specified in clocks.
cfg_sdr_rfmax	3	Maximum number of rows to be refreshed at a time (tRFSH)

4 SDRAM Overview

SDRAM is high-speed Dynamic Random Access Memory (DRAM) with a synchronous interface. The synchronous interface and fully pipelined internal architecture of SDRAM allows extremely fast data rates if used efficiently. SDRAM is organized in banks of memory addressed by row and column. The number of row and column address bits depends on the size and configuration of the memory.

SDRAM is controlled by bus commands that are formed using combinations of the ras_n, cas_n, and we_n signals. For instance, on a clock cycle where all three signals are high, the associated command is a No Operation (NOP). A NOP is also indicated when the chip select is not asserted. The standard SDRAM bus commands are shown in

Command	ras_n	cas_n	we_n
No Operation (NOP)	H	H	H
Active	L	H	H
Read	H	L	H
Write	H	L	L
Burst Terminate	H	H	L
Recharge	L	H	L
Autorefresh	L	L	H
Load mode Register	L	L	L

SDRAM devices are typically divided into four banks. These banks must be opened before a range of addresses can be written to or read from. The row and bank to be opened are registered coincident with the Active command. When a new row on a bank is accessed for a read or a write it may be necessary to first close the bank and then re-open the bank to the new row. Closing a bank is performed using the Precharge command. Opening and closing banks costs memory bandwidth, so the SDRAM Controller Core has been designed to monitor and manage the status of the four banks simultaneously. This enables the controller to intelligently open and close banks only when necessary.
When the Read or Write command is issued, the initial column address is presented to the SDRAM devices. The initial data is presented concurrent with the Write command. For the read command, the initial data appears on the data bus 1-4 clock cycles later. This is known as CAS latency and is due to the time required to physically read the internal DRAM and register the data on the bus. The CAS latency depends on the speed grade of the SDRAM and the frequency of the memory clock. In general, the faster the clock, the more cycles of CAS latency is required. After the initial Read or Write command, sequential read and writes will continue until the burst length is reached or a Burst Terminate command is issued. SDRAM devices support a burst length of up to 8 data cycles. The SDRAM Controller Core is capable of cascading bursts to maximize SDRAM bandwidth.

SDRAM devices require periodic refresh operations to maintain the integrity of the stored data. The SDRAM Controller Core automatically issues the Auto Refresh command periodically. No user intervention is required.

The Load Mode Register command is used to configure the SDRAM operation. This register stores the CAS latency, burst length, burst type, and write burst mode. The SDR controller only writes to the base mode register.

To reduce pin count, SDRAM row and column addresses are multiplexed on the same pins. Table lists the number of rows, columns, banks, and chip selects required for various standard discrete SDR SDRAM devices. The SDR SDRAM Controller Core will support any of these devices.

Chip Size	Config	Row	Columns	Banks
64MB	4M * 16	12	8	4
64MB	2M * 32	11	8	4

Address Mapping from Application layer to SDRAM.

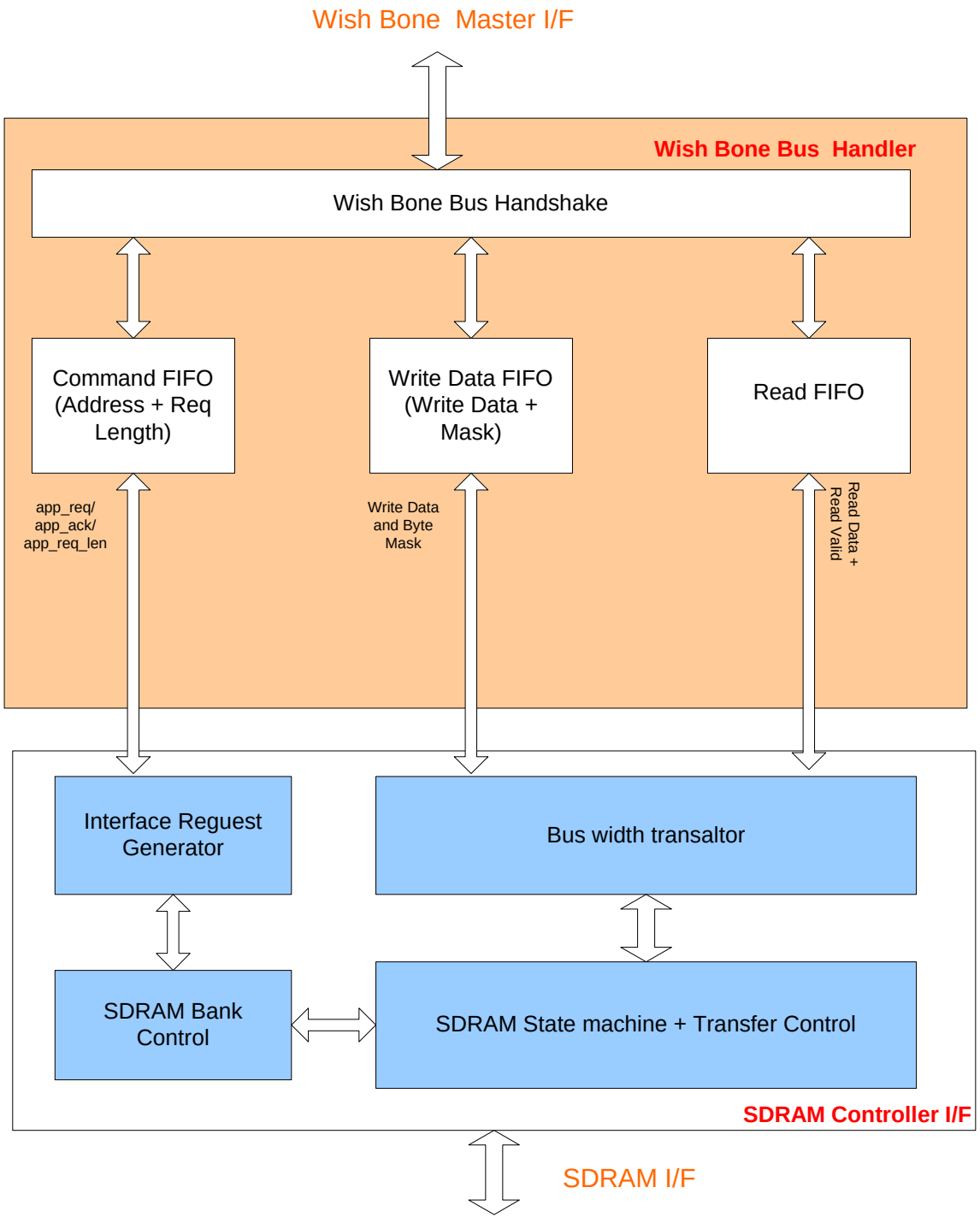
1. For 8 Bit SDRAM mode:
   SDRAM Mapping Address[25:0] = Application Address [25:0];
2. For 16 Bit SDRAM Mode
   SDRAM Mapping Address[25:0] = Application Address [26:1];
3. For 32 Bit SDRAM Mode
   SDRAM Mapping Address[25:0] = Application Address [26:2];

SDRAM Address Mapping Format

Column, Bank and Row Address decoding base on configuration:

cfg_col_bits	
2'b00	Column Address[11:0] = {4'h0,SDRAM Mapping Address[7:0]}; Bank Address[1:0] = {SDRAM Mapping Address[9:8]}; Row Address[11:0] = {SDRAM Mapping Address[21:10];
2'b01	Column Address[11:0] = {3'h0,SDRAM Mapping Address[8:0]}; Bank Address[1:0] = {SDRAM Mapping Address[10:9]}; Row Address[11:0] = {SDRAM Mapping Address[22:11];
2'b10	Column Address[11:0] = {2'h0,SDRAM Mapping Address[9:0]}; Bank Address[1:0] = {SDRAM Mapping Address[11:10]}; Row Address[11:0] = {SDRAM Mapping Address[23:12];
2'b11	Column Address[11:0] = {1'h0,SDRAM Mapping Address[10:0]}; Bank Address[1:0] = {SDRAM Mapping Address[12:11]}; Row Address[11:0] = {SDRAM Mapping Address[24:13];

5 Functional Description



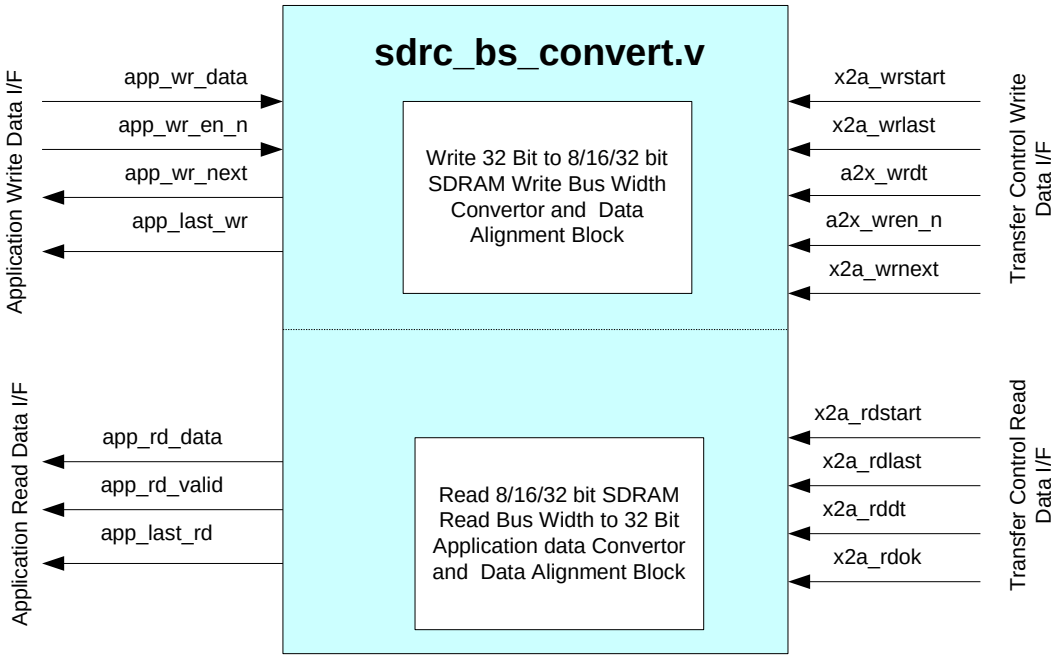
5. 1 Wish Bone Bus Handler:

This block handles the Protocol handshake between wish bone master and custom SDRAM controller. This block also takes care of necessary clock domain change over. This block include ; Command Async FIFO, Write Data Async FIFO, Read Data Async FIFO.

5.2 SDRAM Controller

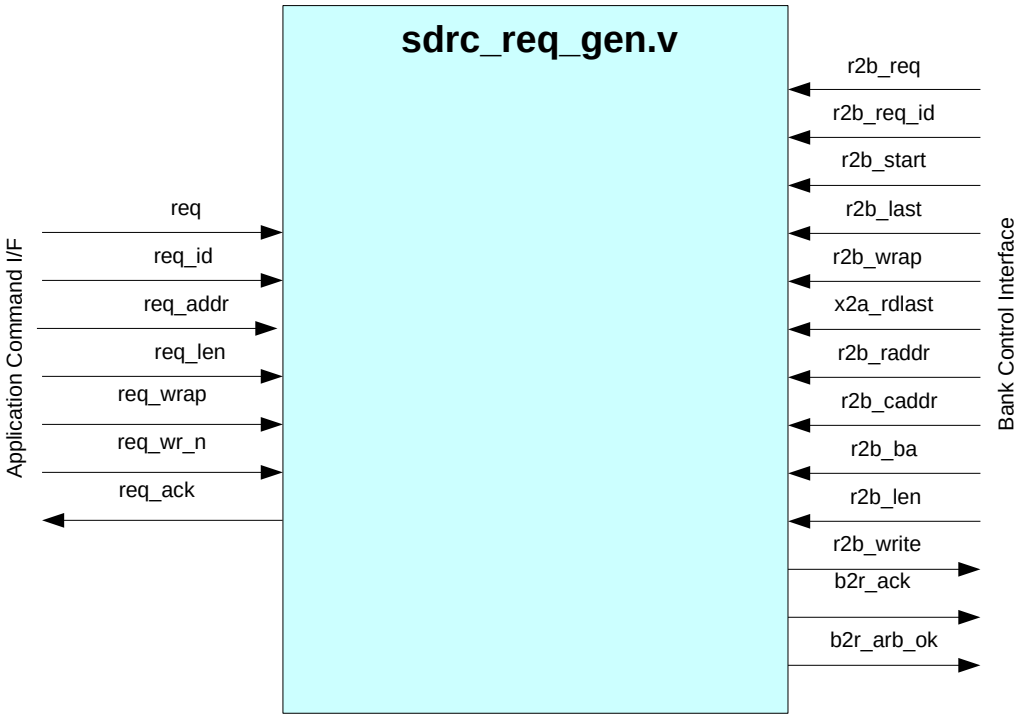
This block include four sub block:

5.2.1 SDRAM Bus convertor: This block convert and re-align the the system side 32 bit into equivalent 8/16/32 SDR format.



During write transaction, it split the 32 Bit Application data into 8/16/32 SDRAM Bus width format.
During Read transaction, it re-packs the 8/16/32bit SDRAM data into 32 bit Application data.

5.2.2 SDRAM Request Generator:



This block does following function:

- Based on the SDRAM bus width, internal address and burst length will be re modified as follows

Sdr_width = 2'b00 32 Bit SDRAM	Internal req address	App_req_addr
	Internal Req Length	App_req_Len
Sdr_width = 2'b01 16 Bit SDRAM	Internal req address	{App_req_addr,1'b0}
	Internal Req Length	{App_req_Len,1'b0
Sdr_width = 2'b10 8 Bit SDRAM	Internal req address	{App_req_addr,2'b0}
	Internal Req Length	{App_req_Len,2'b0

- If the wrap = 0 and current application burst length is crossing the page boundary, then request will be split into two with corresponding change in request address and request length.
- If the wrap = 0 and current burst length is not crossing the page boundary, then request from application layer will be transparently passed on the bank control block.
- If the wrap = 1, then this block will not modify the request address and length. The wrapping functionality will be handle by the bank control module and column address will rewind back as follows XX -> FF → 00 → 1….
- Based on column configuration bit, this block generate the column address, row address and bank address.

Note: With Wrap = 0, each request from Application layer will be splits into two request, if the current burst cross the page boundary.

cfg_colbits= 2'b00	Column Address	Address[7:0]
	Bank Address	Address[9:8]
	Row Address	Address[21:10]
cfg_colbits= 2'b01	Column Address	Address[8:0]
	Bank Address	Address[10:9]
	Row Address	Address[22:11]
cfg_colbits= 2'b10	Column Address	Address[9:0]
	Bank Address	Address[11:10]
	Row Address	Address[23:12]
cfg_colbits= 2'b11	Column Address	Address[10:0]
	Bank Address	Address[12:11]
	Row Address	Address[24:13]

5.2.3 SDRAM BANK CONTROLLER: This module takes requests from SDRAM request generator, checks for page hit/miss and issues precharge/activate commands and then passes the request to SDRAM Transfer Controller.

5.2.4 SDRAM Transfer Controller: This module takes requests from SDRAM Bank controller, runs the transfer and controls data flow to/from the app. At the end of the transfer it issues a burst terminate if not at the end of a burst and another command to this bank is not available.

5.3 SDRAM Interface

Prior to normal operation, SDRAM must be initialized. The following sections provide detailed information covering device initialization, register definition, command descriptions and device operation.

INITIALIZATION

SDRAM must be powered up and initialized in a predefined manner. Operational procedures other than those specified may result in undefined operation. Once power is applied to VDD and the clock is stable, the SDRAM requires a 100 µs delay prior to issuing any command other than a COMMAND INHIBIT or NOP. Starting at some point during this 100 µs period and continuing at least through the end of this period, COMMAND INHIBIT or NOP commands should be applied.

Once the 100 µs delay has been satisfied with at least one COMMAND INHIBIT or NOP command having been applied, a PRECHARGE command should be applied. All device banks must then be precharged, thereby placing the device in the all banks idle state. Once in the idle state, two AUTO REFRESH cycles must be performed. After two refresh cycles are complete, SDRAM ready for mode register programming. Because the mode registers will power up in unknown state, it should be loaded prior to applying any operational command.

MODE REGISTER

The mode register is used to define the specific mode of operation of SDRAM. This definition includes the selection of burst length, a burst type, a CAS latency, an operating mode and a write burst mode as shown in the Mode Register Definition Diagram below. The mode register is programmed via the LOAD MODE REGISTER command and will retain the stored information until it is programmed again or the device loses power.

Mode register bits M0-M2 specify the burst length, M3 specifies the type of burst (sequential or interleaved), M4-M6 specify the CAS latency, M7-M8 specify the Operating mode, M9 specifies the write burst mode, M10 and M11 are reserved for future use. M12 is undefined but should be driven LOW during loading of the mode register. The mode register must be loaded when all device banks are idle, and the Controller must wait the specified time before initiating the subsequent operation. Violating either of these requirements will result in unspecified operation.

Operating Mode

The normal operating mode is selected by setting M7 and M8 to zero; the others combinations of values for M7 and M8 are reserved for future use and/or test modes. The programmable burst length applies to both READ and WRITE bursts.

Writing Burst Mode

When M9 = 0, the burst length programmed via M0-M2 applies to both READ and WRITE bursts; when M9 = 1, the programmed burst length applies to READ bursts, but write accesses are single-location (non-burst) accesses.

Burst Length

The burst length determines the maximum number of column locations that can be accessed for a given READ or WRITE command. Burst lengths of 1, 2, 4 or 8 locations are available for both the sequential and the interleaved burst types, and fullpage burst is available for sequential type only. The full-page burst is used in conjunction with the BURST TERMINATE command to generate arbitrary burst lengths. When a READ or WRITE command is issuing, a block of columns equal to Burst length is effectively selected. All accesses for that burst take place within this block, meaning that the burst will wrap within the block if a boundary is reached, as shown in the Burst Definition Table below. The block is uniquely selected by A1-A9 when burst length is set to two; by A2-A9 when burst length is set to four; and by A3-A9 when burst length is set to eight. The remaining address bits are used to select the starting location within block.

Burst Type

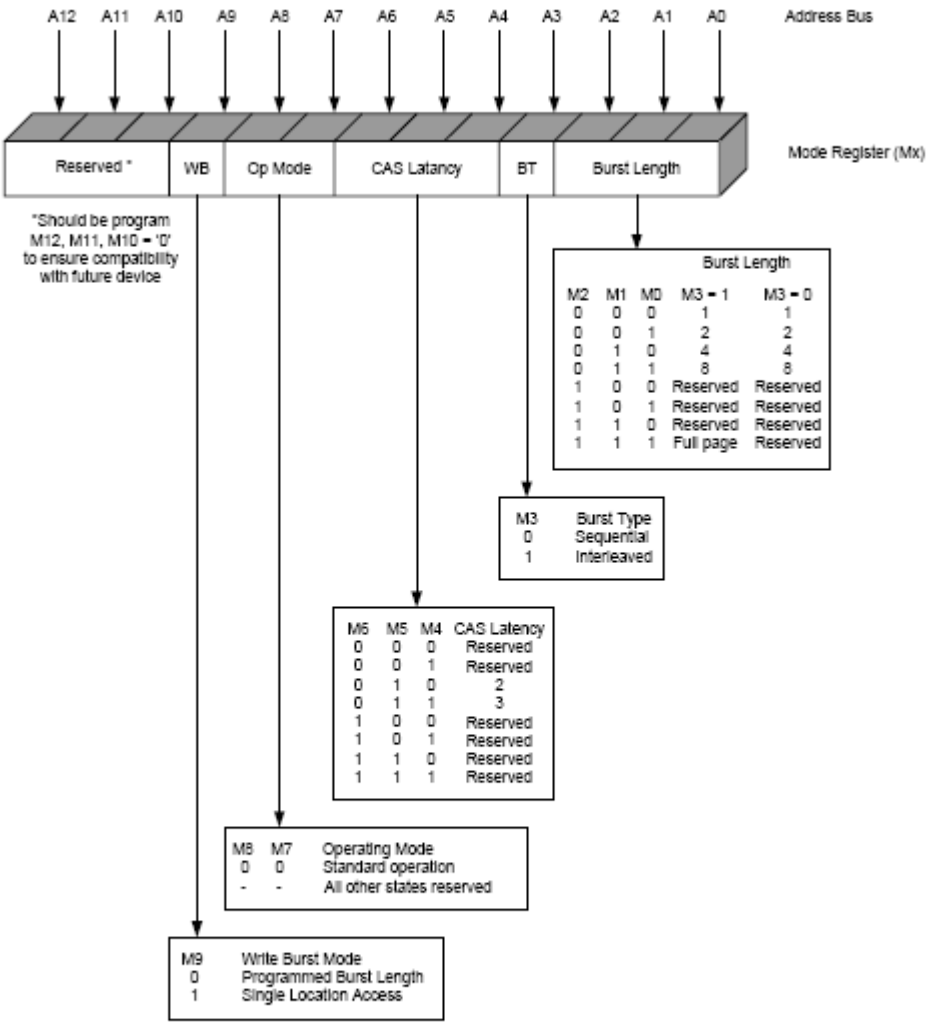
The sequential or interleaved burst is selected via bit M3**.** The ordering of accesses within a burst is determined by the burst length, the burst type and starting column address, as shown in the Burst Definition Table.

Burst Definition Table

Burst Length	Starting Column Address:			Order of Accesses Within a Burst	
				Type = Sequential	Type = interleaved
2			A0		
			0	0-1	0-1
			1	1-0	1-0
4		A1	A0		
		0	0	0-1-2-3	0-1-2-3
		0	1	1-2-3-0	1-0-3-2
		1	0	2-3-0-1	2-3-0-1
		1	1	3-0-1-2	3-2-1-0
8	A2	A1	A0		
	0	0	0	0-1-2-3-4-5-6-7	0-1-2-3-4-5-6-7
	0	0	1	1-2-3-4-5-6-7-0	1-0-3-2-5-4-7-6
	0	1	0	2-3-4-5-6-7-0-1	2-3-0-1-6-7-4-5
	0	1	1	3-4-5-6-7-0-1-2	3-2-1-0-7-6-5-4
	1	0	0	4-5-6-7-0-1-2-3	4-5-6-7-0-1-2-3
	1	0	1	5-6-7-0-1-2-3-4	5-4-7-6-1-0-3-2
	1	1	0	6-7-0-1-2-3-4-5	6-7-4-5-2-3-0-1
	1	1	1	7-0-1-2-3-4-5-6	7-6-5-4-3-2-1-0
Full Page (y)	n = A0-A11/9/8 (location 0-y)			C_n, $C_n + 1$, $C_n + 2$, $C_n + 3$, $C_n + 4$… … $C_n - 1$, C_n	Not supported

NOTE:

1. For full-page access: y = 1,024.
2. For a burst length of two, A1-A9 selects the block-of-two burst; A0 selects the starting column within the block.
3. For a burst length of four, A2-A9 selects the block-of-four burst; A0-A1 selects the starting column within the block.
4. For a burst length of eight, A3-A9 selects the block-of-eight burst; A0-A2 selects the starting column within the block.
5. For a full-page burst, full row is selected and A0-A9 selects the starting column.

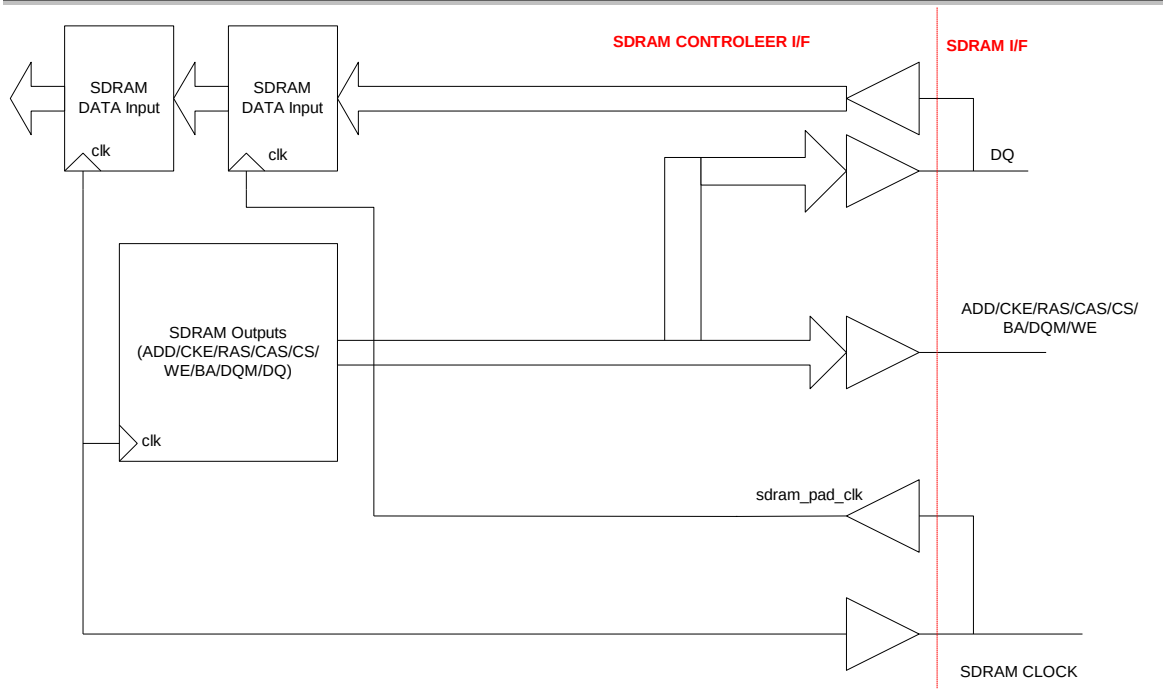


Mode Register Definition Diagram

CAS Latency

The CAS latency is the delay, in clock cycles, between the registration of a READ command and the availability of the first piece of output data. The latency can be set two or three clocks. The CAS Latency Table indicates the operating frequencies at which each CAS latency setting can be used.

6. SDRAM Data Path



To simply the interface timing issue at SDRAM interface, all the output signals are driven with SDRAM_CLOCK. DQ Input is captured with Delayed sdram_pad_clk.

7 Simulation

Run Directorty: **sdr_ctrl/trunk/verif/run**

All the compile and simulation are compatible with model simulator. Automatic simulation is available at top-level (SDRAM Controller Core + Wish Bone Converter) and also at Stand-alone SDRAM Controller Core level

- Filelist_core.f → Includes RTL and TB files list corresponds to stand-alone SDRAM Controller core level
- Filelist_top.f → Includes RTL and TB files list corresponds to integrated SDRAM Controller core + WishBone converter.
- Filelist_rtl.f → Includes RTL files list corresponds to integrated SDRAM Controller core + WishBone converter.
- compile.modelsim → Model Simulator compile script
- run_modelsim → Include compile and simulation script.
- run_all → Automated run scripts include 8/16/32 Bit SDR mode. This script includes both stand-alone SDRAM Controller + Integrated SDRAM Controller with wishbone interface

Simulation complete test-case at **top-level – Integrated SDRAM Controller with wishbone interface**

1. Compiling and Simulating in 8 BIT SDR Mode
   ./run_modelsim top SDR_8BIT
2. Compiling and Simulating in 16 BIT SDR Mode
   ./run_modelsim top SDR_16BIT
3. Compiling and Simulating in 32 BIT SDR Mode
   ./run_modelsim top SDR_32BIT

Simulation complete test-case at **core level – Standalone SDRAM Controller**

1. Compiling and Simulating in 8 BIT SDR Mode
   ./run_modelsim core SDR_8BIT
2. Compiling and Simulating in 16 BIT SDR Mode
   ./run_modelsim core SDR_16BIT
3. Compiling and Simulating in 32 BIT SDR Mode
   ./run_modelsim core SDR_32BIT

Running individual test-case at SDRAM Top Level (Refer file: run_modelsim)

1. Running in 8 Bit SDRA
   ./compile.modelsim top SDR_8BIT
   vsim -do run.do -c tb_top
2. Running in 16 Bit SDRAM
   ./compile.modelsim top SDR_16BIT

vsim -do run.do -c tb_top

3. Running in 32 Bit SDRA
   ./compile.modelsim top SDR_32BIT
   vsim -do run.do -c tb_top

Running individual test-case at SDRAM Core Level (Refer file: run_modelsim)

4. Running in 8 Bit SDRA
   ./compile.modelsim core SDR_8BIT
   vsim -do run.do -c tb_top
5. Running in 16 Bit SDRAM
   ./compile.modelsim core SDR_16BIT
   vsim -do run.do -c tb_top
6. Running in 32 Bit SDRA
   ./compile.modelsim core SDR_32BIT
   vsim -do run.do -c tb_top

Golden Log file are available under: **sdr_ctrl/trunk/verif/log**

Log files corresponds the Integrated SDRAM Top-level:

- **Top_SDR_16BIT_complie.log** - SDR-16BIT Compile log
- **Top_sdr16_sim.log** – SDR-16BIT Simulation log
- **Top_SDR_16BIT_basic_test1.log** -- SDR-16BIT Basic Test simulation log
- **Top_SDR_32BIT_complie.log** - SDR-32BIT Compile log
- **Top_sdr32_sim.log** -- SDR-32BIT Simulation log
- **Top_SDR_32BIT_basic_test1.log** – SDR-32BIT Basic Test Simulation log

Log files corresponds the Stand-alone SDRAM core-level:

- **core_SDR_16BIT_complie.log** - SDR-16BIT Compile log
- **core_sdr16_sim.log** – SDR-16BIT Simulation log
- **core_SDR_16BIT_basic_test1.log** -- SDR-16BIT Basic Test simulation log
- **core_SDR_32BIT_complie.log** - SDR-32BIT Compile log
- **core_sdr32_sim.log** -- SDR-32BIT Simulation log
- **core_SDR_32BIT_basic_test1.log** – SDR-32BIT Basic Test Simulation log

Typical Regression Status Report:

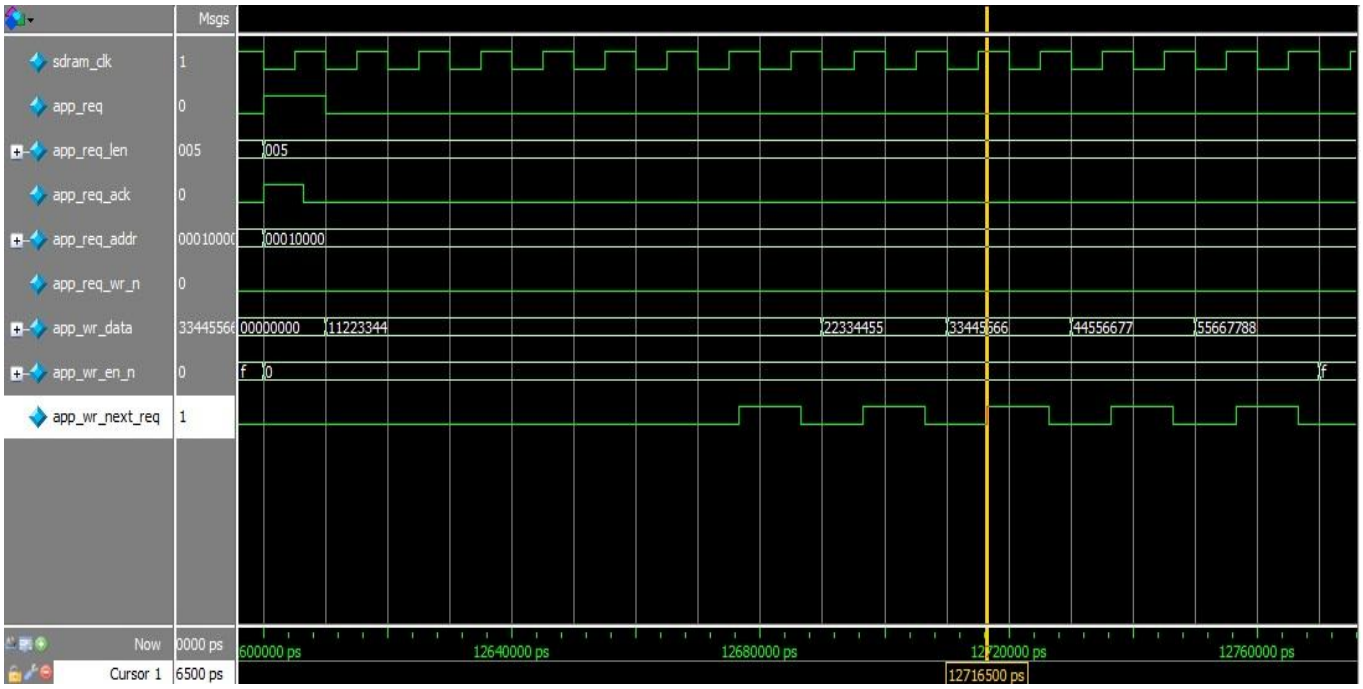
```
##########################################
  Analysis the Regression Status
##########################################
##########################################
### test 1: top_SDR_8BIT_basic_test1 --> PASSED
### test 2: top_SDR_16BIT_basic_test1 --> PASSED
### test 3: top_SDR_32BIT_basic_test1 --> PASSED
### test 4: core_SDR_8BIT_basic_test1 --> PASSED
### test 5: core_SDR_16BIT_basic_test1 --> PASSED
### test 6: core_SDR_32BIT_basic_test1 --> PASSED
```

```
##############################################

##############################################
###  Test Logs
   test 1: ../log/top_SDR_8BIT_basic_test1.log
   test 2: ../log/top_SDR_16BIT_basic_test1.log
   test 3: ../log/top_SDR_32BIT_basic_test1.log
   test 4: ../log/core_SDR_8BIT_basic_test1.log
   test 5: ../log/core_SDR_16BIT_basic_test1.log
   test 6: ../log/core_SDR_32BIT_basic_test1.log
##############################################

##############################################
###  Test Summary
###
### Failed 0 of 6 tests
##############################################
```

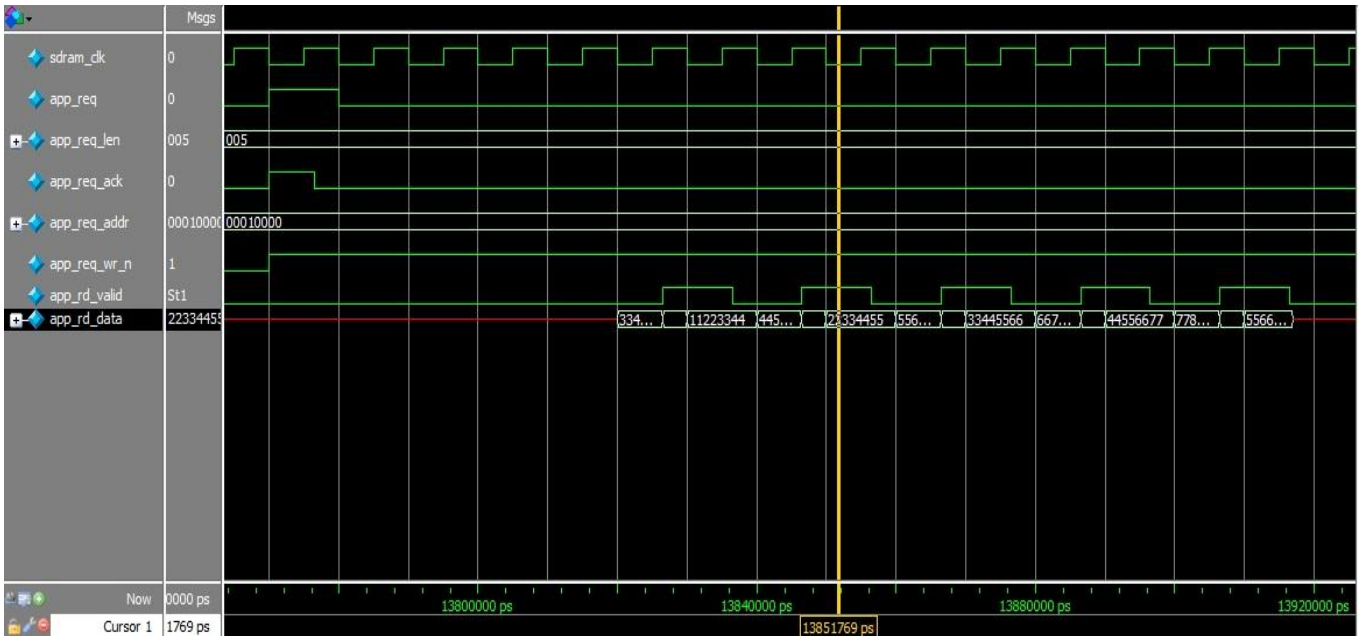
8 WAVEFORM

Image files are available under: **sdr_ctrl\trunk\verif\dump**

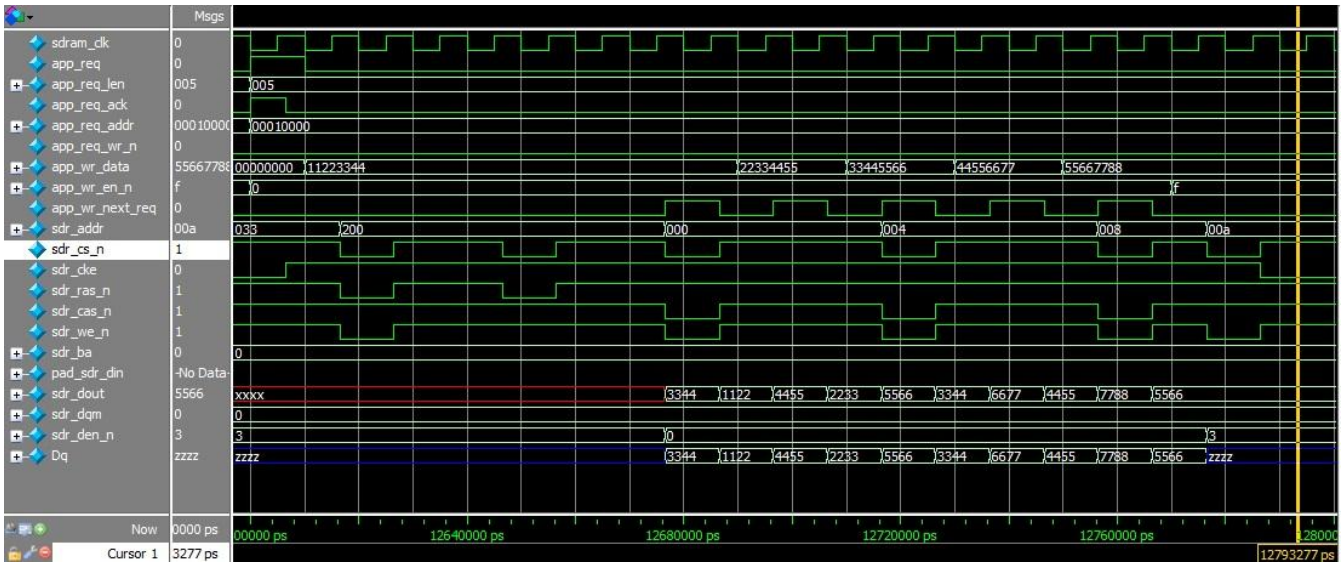
1. Application Write Request:



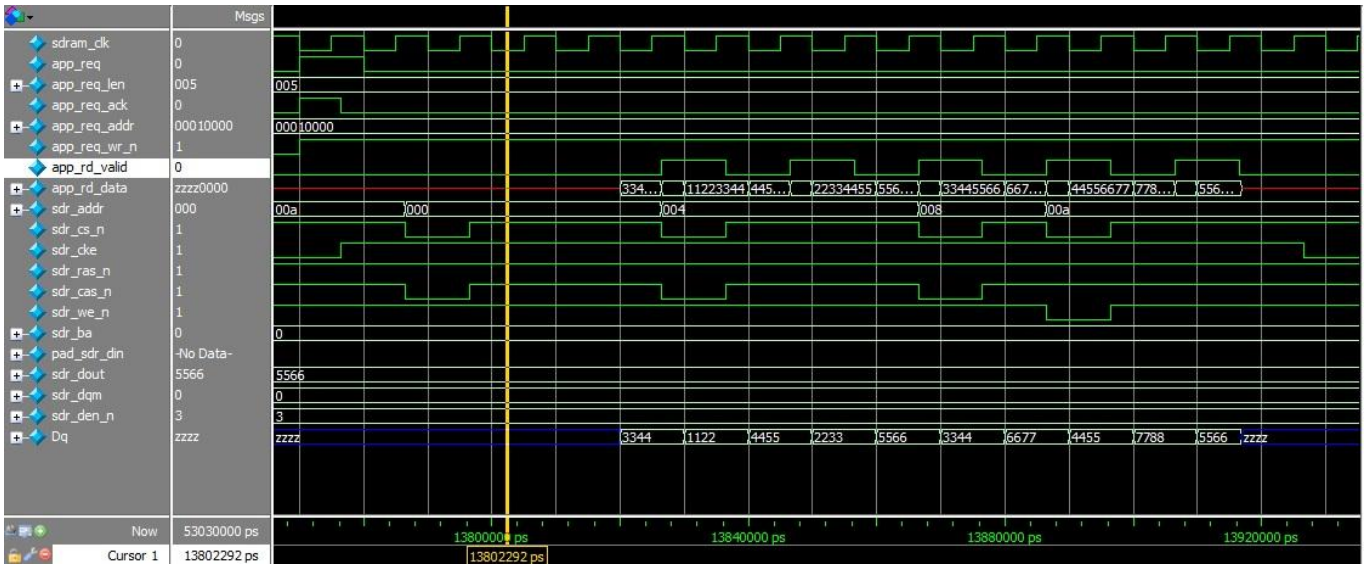
2. Application Read Request



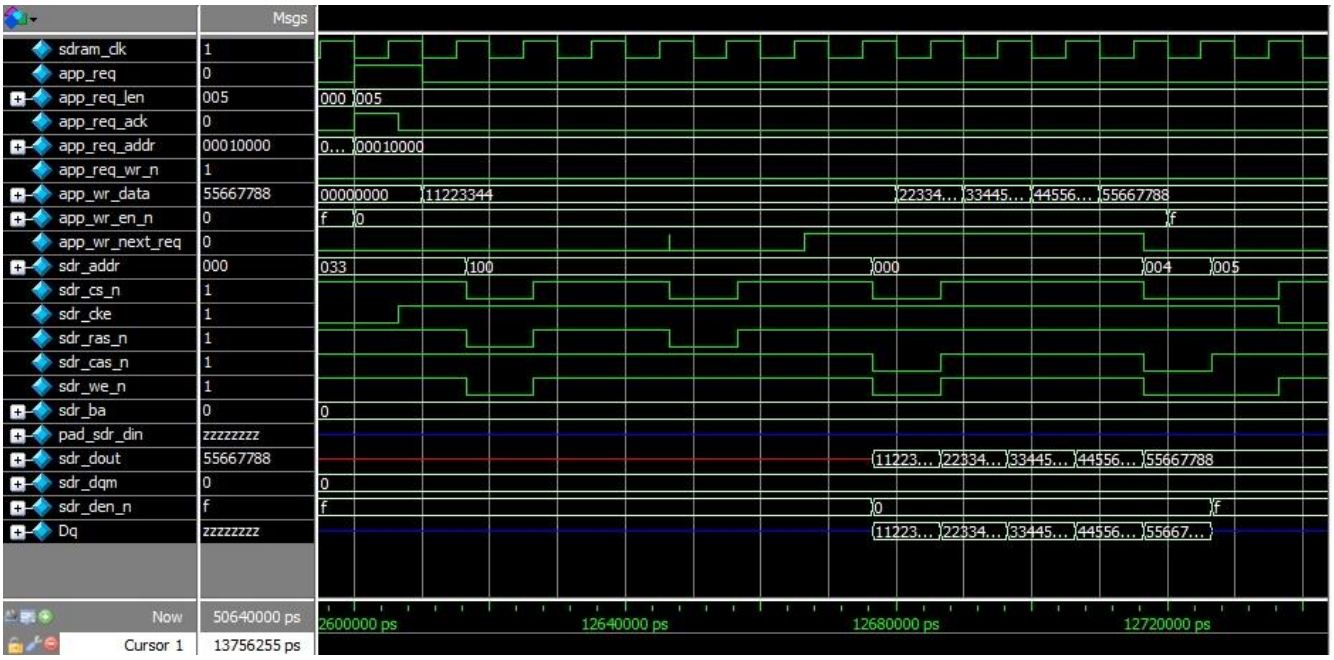
3. SDRAM 16 Bit Write Transaction



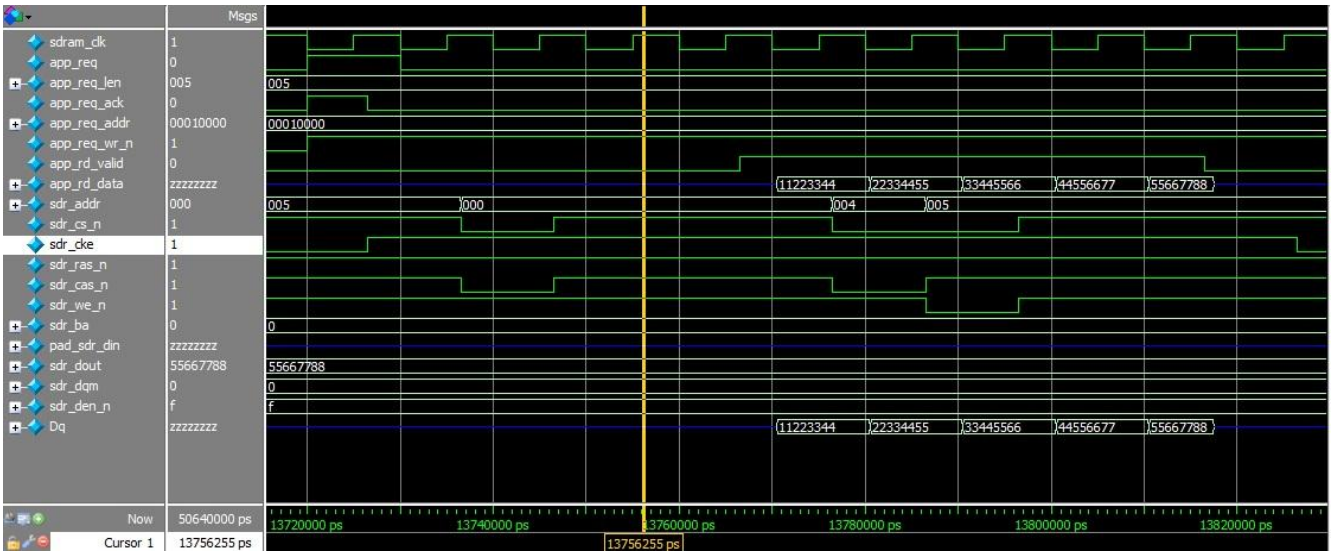
4. SDRAM 16 Bit Read Transaction



5. SDRAM 32 Bit Write Transaction

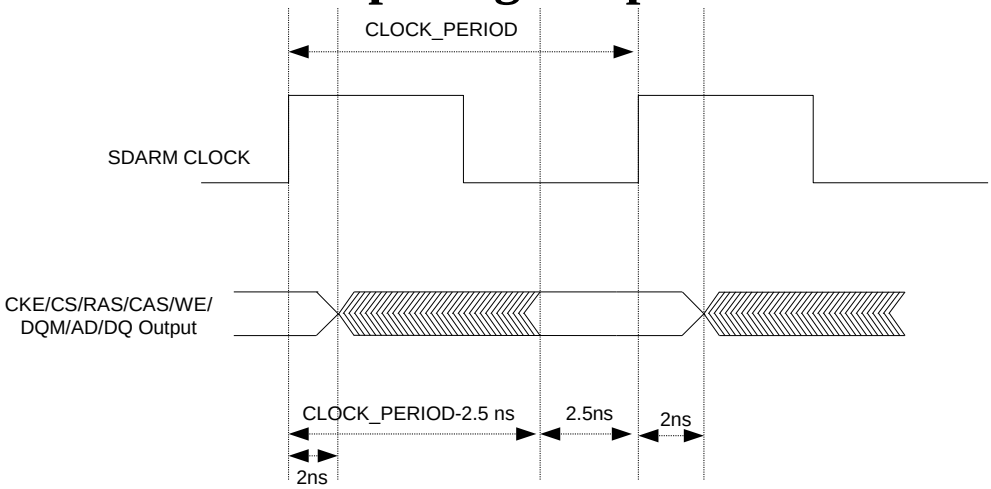


6. SDRAM 32 bit Read Transaction



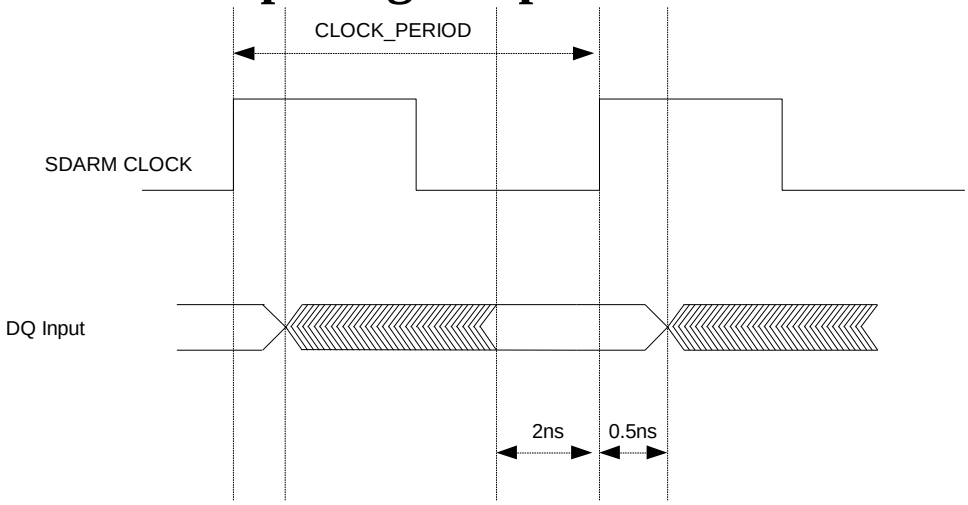
Appendix A : Timing

A.1 SDRAM Output signals parameter definition



Signal Name	output Delay (ns)	
	Minimum	Maximum
sdr_cs_n, sdr_cke, sdr_ras_n, sdr_cas_n, sdr_we_n, sdr_dqm, sdr_ba, sdr_addr, sdr_dq	2	CLK_PERIOD-2.5ns

A2. SDRAM Input signals parameter definition



Signal Name	Input Delay (ns)	
	Setup	Hold
sdr_dq	2	0.5

Appendix A : Frequently Asked Q

1. Design and implementation language used in the IP
   <Answer> Design implementation is done Verilog and System verilog language
2. What are the SDRAM Bus width are supported by the IP?
   <Ans> This IP Supports 8/16/32 Bit interface
3. What are the Application Bus width are supported by the IP?
   <Ans> This IP Supports only 32 bit Application Bus width
4. Can Application clock and SDRAM clock be Asynchronous to each other?
   <Ans> Yes, IP support both Synchronous and Asynchronous Application clock and SDRAM clock
5. Is the application layer is compatible to wish-bone standard?.
   <Ans> Yes, Application Layer is wishbone compatible.
6. Is SDRAM cores is also available with custom interface?
   <Ans> Yes. SDRAM core is separately available with automated test-bench.
7. Test bench scripts are compatible to which tool?
   <Answer> Verification scripts are compatible to model simulator