SDR SDRAM Controller

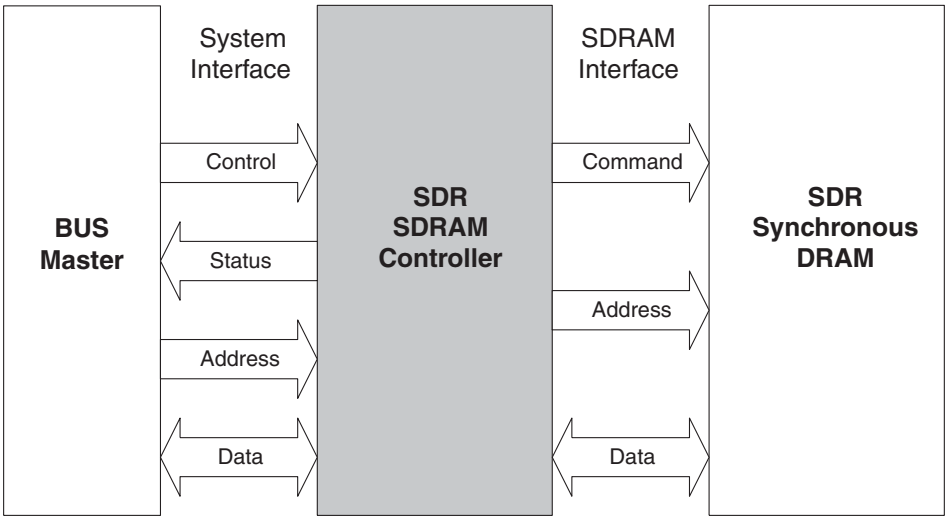
September 2014 Reference Design RD1010

Introduction

Synchronous DRAM (SDRAM) has become a mainstream memory of choice in embedded system memory design due to its speed, burst access and pipeline features. For high-end applications using processors such as Motorola MPC 8260 or Intel StrongArm, 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. In some cases, SDRAM is chosen because the previous generations of DRAM (FP and EDO) are either end-of-life or not recommended for new designs by the memory vendors. From the board design point of view, design using earlier generations of DRAM is much easier and more straightforward than using SDRAM unless the system bus master provides the SDRAM interface module as mentioned above. 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.

In today's SDRAM market, there are two major types of SDRAM distinguished by their data transfer rates. The most common single data rate (SDR) SDRAM transfers data on the rising edge of the clock. The other is the double data rate (DDR) SDRAM which transfers data on both the rising and falling edge to double the data transfer throughput. Other than the data transfer phase, the different power-on initialization and mode register definitions, these two SDRAMs share the same command set and basic design concepts. This reference design is targeted for SDR SDRAM, however, due to the similarity of SDR and DDR SDRAM, this design can also be modified for a DDR SDRAM controller. For illustration purposes, the Micron SDR SDRAM MT48LC32M4A2 (8Meg x 4 x 4 banks) is chosen for this design. Also, this design has been verified by using Micron's simulation model. It is highly recommended to download the simulation model from the SDRAM vendors for timing simulation when any modifications are made to this design.

Figure 1. SDR SDRAM Controller System



Features

- Simplifies SDRAM command interface to standard system read/write interface.
- Internal state machine built for SDRAM power-on initialization.
- Read/write cycle access time optimized automatically according to the SDRAM timing specification and the mode it is configured to.
- Dedicated auto-refresh request input and acknowledge output for SDRAM refresh.
- Easily configurable to support different CAS latency and burst length.
- By taking advantage of the sysCLOCK™ PLL feature, a slower system clock can be used. The system interface clock does not need to be the same as the SDRAM clock.
- With the support of the sysIO™ feature available in all Lattice devices, the system interface can be in any I/O standards supported by the device.

Pin Descriptions

Pin Name	Type	Pin Description
sys_R_Wn	In	System interface read/write signal. High indicates a read cycle and low indicates a write cycle. When this pin is high, it indicates to the controller that the bus master is performing a read cycle. When low, it indicates that it's a write cycle.
sys_ADSn	In	Active low system interface address strobe. This pin indicates the start of a bus master cycle.
sys_DLY_100US	In	This active high signal indicates to the controller that the SDRAM has gone through the 100µs delay for power and clock stabilization.
sys_CK	In	System interface clock.
sys_RESET	In	This active high signal resets the controller to the initial state.
sys_REF_REQ	In	Active high SDRAM refresh request.
sys_REF_ACK	Out	Active high SDRAM refresh acknowledge
sys_A	In	System interface address bus.
sys_D	In/Out	Bi-directional three-state system interface data bus.
sys_D_VALID	Out	Active high data valid signal. This pin activates only for read cycles and indicates the data currently present on the system interface data bus sys_D is valid.
sys_CYC_END	Out	This active high signal indicates to the bus master that the system interface read/write cycle is completed. This pin is active after reset. After that, it is negated at the first clock and asserted at the last clock of the system interface read/write cycle.
sys_INIT_DONE	Out	This active high signal indicates that the SDRAM initialization is completed.
sdr_DQ	In/Out	SDRAM data bus
sdr_A	Out	SDRAM address bus
sdr_BA	Out	SDRAM bank address
sdr_CK	Out	SDRAM clock (If PLL is used, this will be the PLL output pin PLL_OUT0 or PLL_OUT1.)
sdr_CKE	Out	SDRAM clock enable
sdr_CSn	Out	SDRAM command inputs CS#
sdr_RASn	Out	SDRAM command inputs RAS#
sdr_CASn	Out	SDRAM command inputs CAS#
sdr_Wen	Out	SDRAM command inputs WE#
sdr_DQM	Out	SDRAM data bus mask

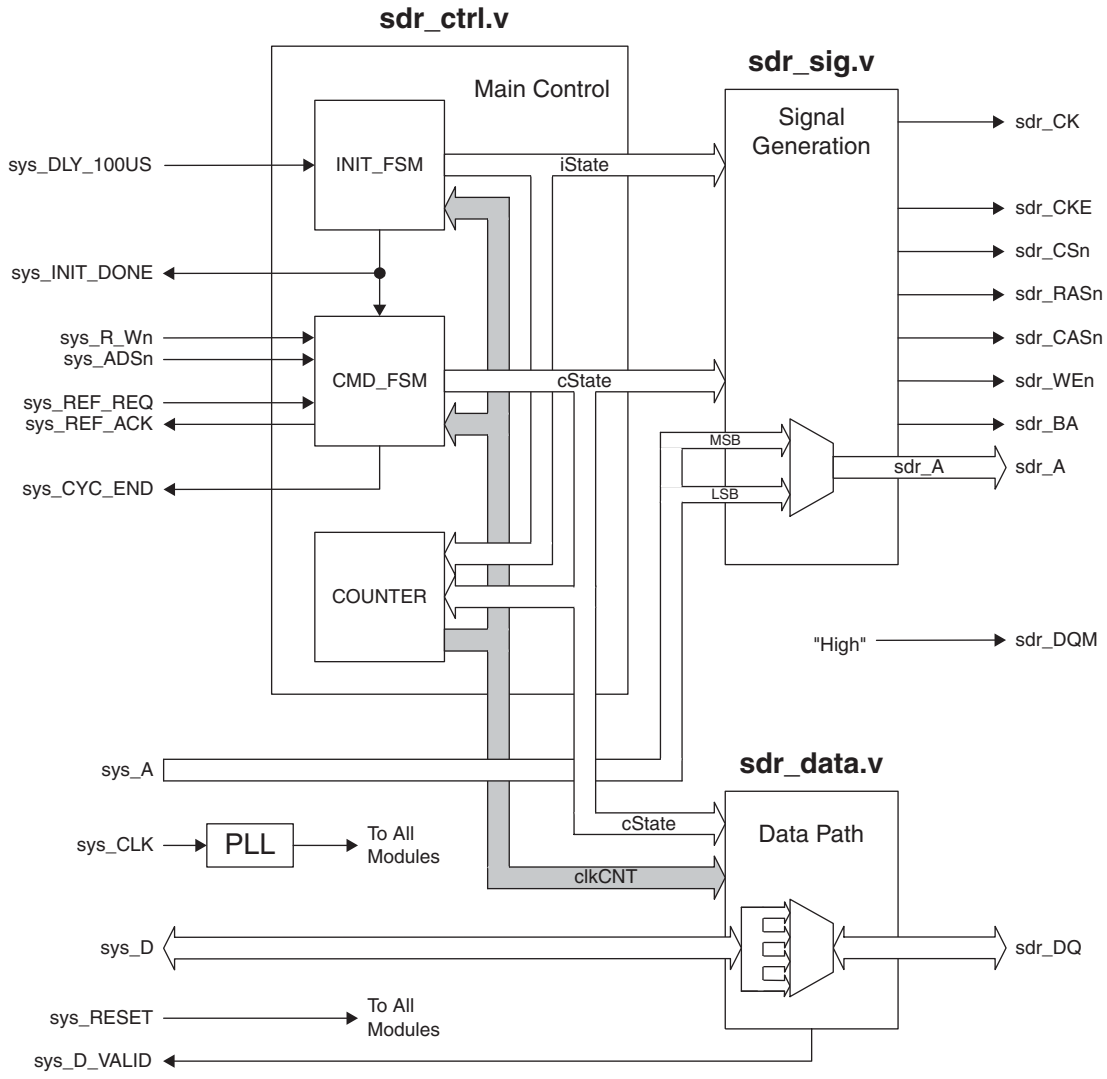
Functional Description

The functional block diagram of the SDRAM controller is shown in Figure 2. It consists of three modules: the main control module, the signal generation module and the data path module. The main control module, containing two state machines and a counter, is the primary module of the design which generates proper iState and cState outputs according to the system interface control signals. The signal generation module generates the address and command signals required for SDRAM based on iState and cState. The data path module performs the data latching and dispatching of the data between the bus master and SDRAM.

If using a PLL, all modules derive internal timing from the PLL clock output. This PLL clock also can be connected to the SDRAM directly. A separate on-board SDRAM clock may not be required. With the sysIO feature, the bus master I/O can be LVCMOS 1.8/2.5/3.3, LVTTL, PCI, PCI-X, GTL+ or any of the other standards supported by the device being used.

When targeting to other Lattice CPLD devices, instead of generating SDRAM clock with the CPLD, the system needs an on-board clock source such as a clock oscillator to generate the clocks for both the CPLD and the SDRAM.

Figure 2. Block Diagram



Benefits of Using PLL

As mentioned in the functional description section above, the SDRAM clock can be generated by the internal PLL of the devices. For example, if the system is running at 40MHz, a 100MHz SDRAM clock can be obtained through the dedicated PLL output pin by setting the proper PLL attributes (multiply = 5 and divide = 2).

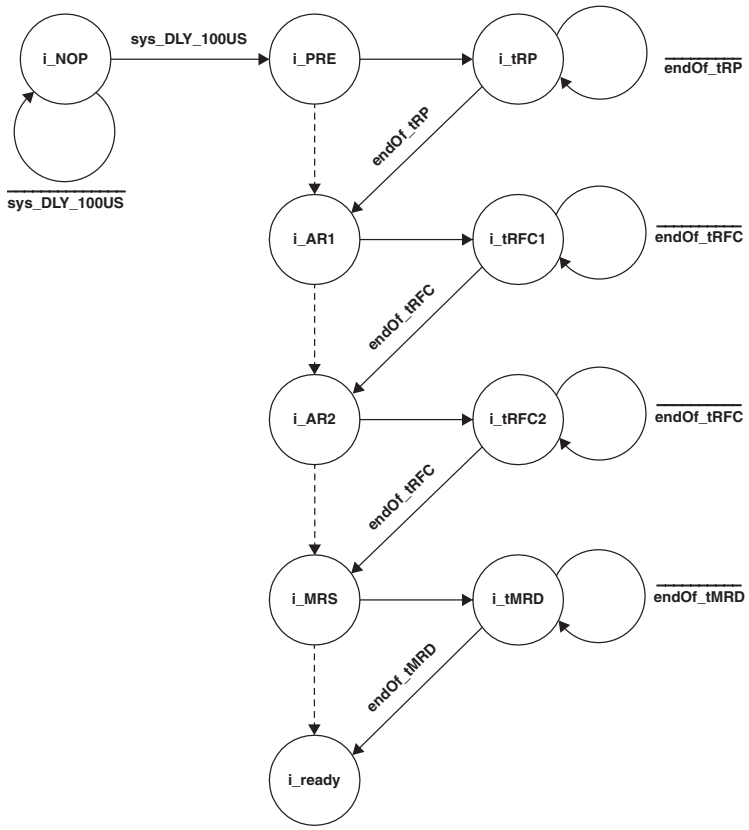
Also, by using the PLL's variable delay line capability, all the output signals to the SDRAM can be tuned to retard or advance the normal output timing for timing optimization and system reliability improving.

SDRAM Initialization

Before normal memory accesses can be performed, the SDRAM needs to be initialized by a sequence of commands. The INIT_FSM state machine handles this initialization. Figure 3 shows the state diagram of the INIT_FSM state machine. During reset, the INIT_FSM is forced to the i_NOP state. After reset, the sys_100μs signal will be sampled at the rising edge of every PLL clock cycle to determine if the 100μs power/clock stabilization delay is completed. After the power/clock stabilization is complete, the SDRAM initialization sequence will begin and the INIT_FSM will switch from i_NOP to i_PRE state. The initialization starts with the PRECHARGE command, followed by two AUTO REFRESH commands, and then the LOAD MODE REGISTER command to configure SDRAM to a specific mode of operation. The i_PRE, i_AR1, i_AR2 and i_MRS states are used for issuing these commands. After each of these commands is issued, a corresponding timing delay needs to be satisfied before any command other than NOP can be issued. These timing delays are t_{RP}, t_{RFC} and t_{MRD} for command PRECHARGE, AUTO REFRESH and LOAD MODE REGISTER respectively. After issuing the LOAD MODE REGISTER command and the tMRD timing delay is satisfied, INIT_FSM goes to i_ready state and remains there for the normal memory access cycles unless sys_RESET is asserted. Also, signal sys_INIT_DONE is set to high to indicate the SDRAM initialization is completed.

The LOAD MODE REGISTER command configures the SDRAM by loading data into the mode register through the address bus. The data present on the address bus during the LOAD MODE REGISTER command is loaded to the mode register. The mode register contents specify the burst length, burst type, CAS latency, etc. Refer to the SDRAM vendor's data sheet for more detailed information about the mode register field definitions. As long as all banks of the SDRAM are put into idle state by the PRECHARGE or AUTO PRECHARGE, the mode register can be reloaded with different values, thereby changing the mode of operation. However, in most applications, the mode register value will not be changed after the initialization. This design assumes the mode register stays the same after initialization and a fixed mode register content is implemented in the HDL code. The mode register content in the HDL code may need to be modified to suit the user's needs.

Figure 3. INIT_FSM State Diagram

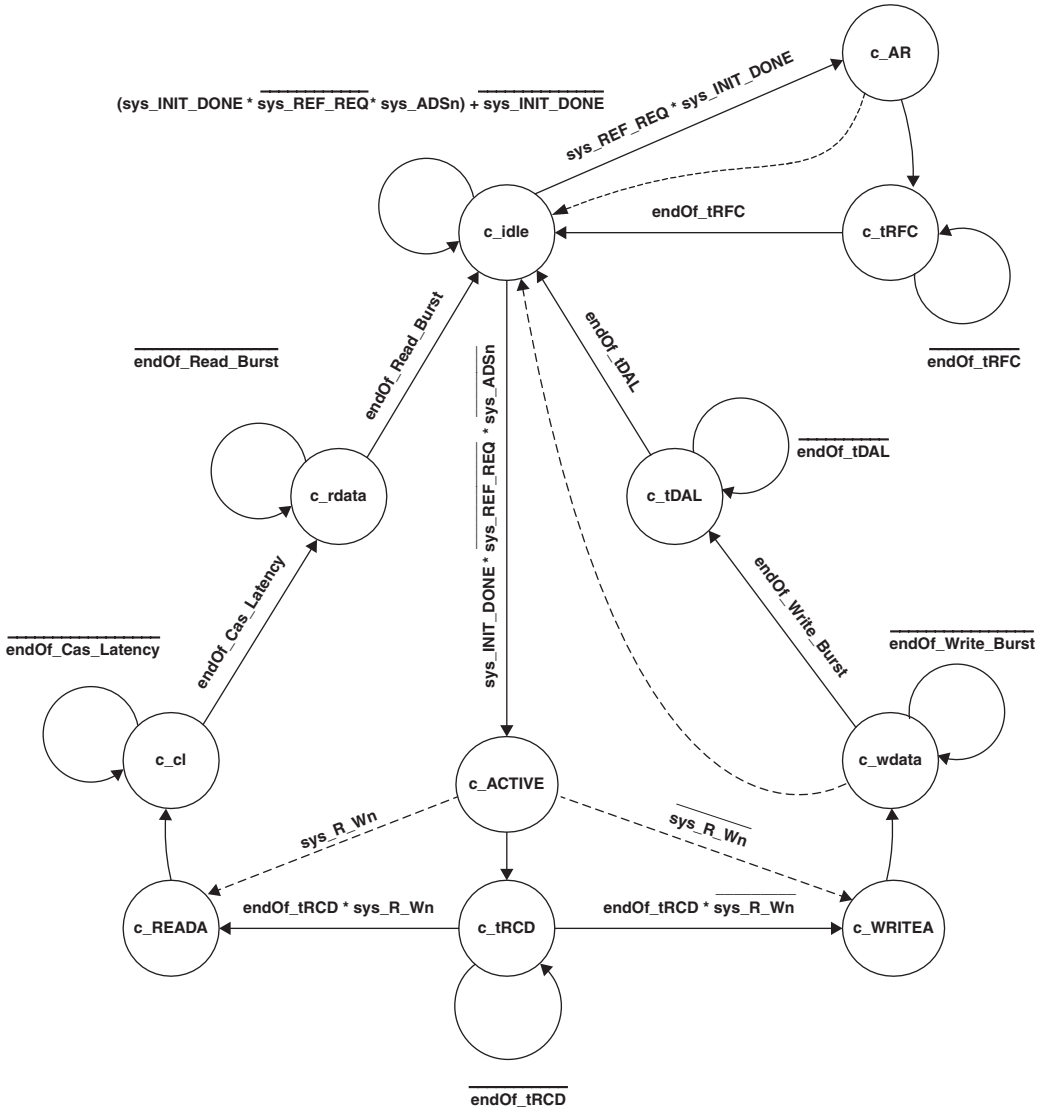


As mentioned above, certain timing delays (t_{RP}, t_{RFC}, t_{MRD}) need to be satisfied before another non-NOP command can be issued. These SDRAM delays vary from speed grade to speed grade and sometimes from vendor to vendor. To accommodate this without sacrificing performance, the designer needs to modify the HDL code for the specific delays and clock period (t_{CK}). According to these timing values, the number of clocks the state machine will stay at i_tRP, i_tRFC1, i_tRFC2, i_tMRD states will be determined after the code is synthesized. In cases where t_{CK} is larger than the timing delay, the state machine doesn't need to switch to the timing delay states and can go directly to the command states. The dashed lines in Figure 3 show the possible state switching paths.

Read/Write Cycle

Figure 4 shows the state diagram of CMD_FSM which handles the read, write and refresh of the SDRAM. The CMD_FSM state machine is initialized to c_idle during reset. After reset, CMD_FSM stays in c_idle as long as sys_INIT_DONE is low which indicates the SDRAM initialization sequence is not yet completed. Once the initialization is done, sys_ADSn and sys_REF_REQ will be sampled at the rising edge of every clock cycle. A logic high sampled on sys_REF_REQ will start a SDRAM refresh cycle. This is described in the following section. If logic low is sampled on both sys_REF_REQ and sys_ADSn, a system read cycle or system write cycle will begin. These system cycles are made up of a sequence of SDRAM commands.

Figure 4. CMD_FSM State Diagram



Similar to the FP and EDO DRAM, row address and column address are required to pinpoint the memory cell location of the SDRAM access. Since SDRAM is composed of four banks, bank address needs to be provided as well.

The SDRAM can be considered as a four by N array of rows. All rows are in the “closed” status after the SDRAM initialization. The rows need to be “opened” before they can be accessed. However, only one row in the same bank can be opened at a time. Since there are four banks, there can be at most four rows opened at the same time. If a row in one bank is currently opened, it must be closed before another row in the same bank can be opened. ACTIVE command is used to open the rows and PRECHARGE (or the AUTO PRECHARGE hidden in the WRITE and READ commands, as used in this design) is used to close the rows. When issuing the commands for opening or closing the rows, both row address and bank address need to be provided.

For sequential access applications and those with page memory management, the proper address assignments and the use of the SDRAM pipeline feature deliver the highest performance SDRAM controller. However, this type of controller design is highly associated with the bus master cycle specification and will not fit the general applications. Therefore, this SDRAM controller design does not implement these custom features to achieve the highest performance through these techniques.

In this design, the ACTIVE command will be issued for each read or write access to open the row. After a t_{RCD} delay is satisfied, READ or WRITE commands will be issued with a high sdr_A[10] to enable the AUTO REFRESH for closing the row after access. So, the clocks required for read/write cycle are fixed and the access can be random over the full address range.

Read or write is determined by the sys_R_Wn status sampled at the rising edge of the clock before tRCD delay is satisfied. If a logic high is sampled, the state machine switches to c_READA. If a logic low is sampled, the state machine switches to c_WRITEA.

For read cycles, the state machine switches from c_READA to c_cl for CAS latency, then switches to c_rdata for transferring data from SDRAM to bus master. The number of clocks the state machine stays in c_rdata state is determined by the burst length. After the data is transferred, it switches back to c_idle.

For write cycles, the state machine switches from c_WRITEA to c_wdata for transferring data from bus master to SDRAM, then switches to c_tDAL. Similar to read, the number of clocks the state machine stays in c_wdata state is determined by the burst length. The time delay tDAL is the sum of WRITE recovery time t_{WR} and the AUTO PRECHARGE timing delay t_{RP}. After the clock rising edge of the last data in the burst sequence, no commands other than NOP can be issued to SDRAM before t_{DAL} is satisfied.

As mentioned in the INIT_FSM section above, the dash lines indicates possible state switching paths when t_{CK} period is larger than timing delay spec.

Refresh Cycle

Similar to the other DRAMs, memory refresh is required. A SDRAM refresh request is generated by activating sdr_REF_REQ signal of the controller. The sdr_REF_ACK signal will acknowledge the recognition of sdr_REF_REQ and will be active throughout the whole refresh cycle. The sdr_REF_REQ signal must be maintained until the sdr_REF_ACK goes active in order to be recognized as a refresh cycle. Note that no system read/write access cycles are allowed when sdr_REF_ACK is active. All system interface cycles will be ignored during this period. The sdr_REF_REQ signal assertion needs to be removed upon receipt of sdr_REF_ACK acknowledge, otherwise another refresh cycle will again be performed.

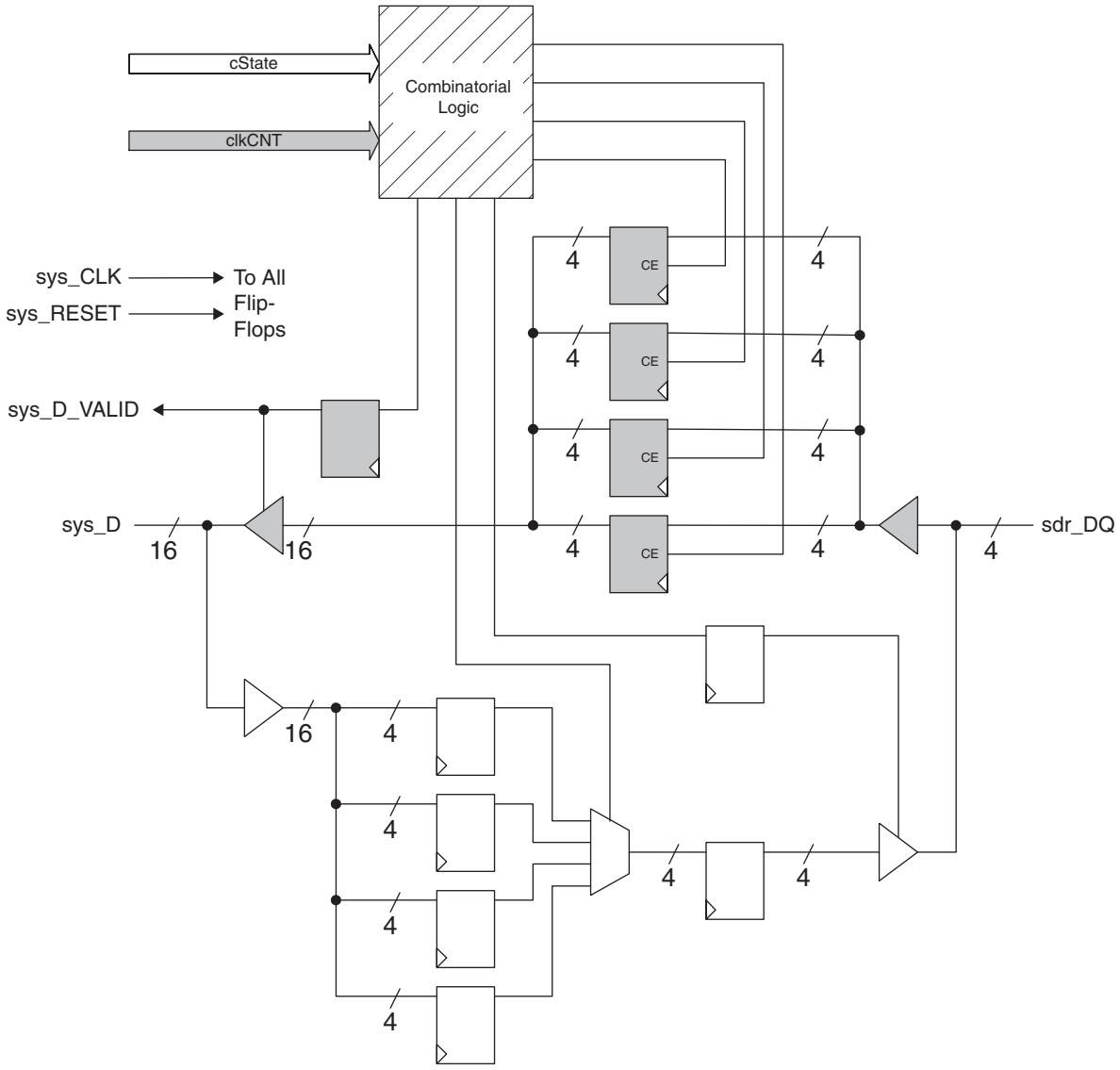
Upon receipt of sdr_REF_REQ assertion, the state machine CMD_FSM enters the c_AR state to issue an AUTO REFRESH command to the SDRAM. After t_{RFC} time delay is satisfied, CMD_FSM returns to c_idle.

Data Path

Figure 5 shows the data flow design between the SDRAM and the system interface. The module in this reference design interfaces between the SDRAM with 4-bit data bus and the bus master with 16-bit data bus. The user should be able to modify this module to customize to fit his/her system bus requirements.

The size of each bus in Figure 5 is shown by the number under the slash across the bus. The grayed components are for read cycles and the white components are for write cycles.

Figure 5. Data Path Module



Timing Diagrams

Figures 6 and 7 are the read cycle and write cycle timing diagrams of the reference design with the two CAS latency cycles and the burst length of four. The timing diagrams may be different due to the values of the timing delays $t_{MRD}/t_{RP}/t_{RFC}/t_{RCD}/t_{RCD}/t_{WR}$, the clock period t_{CK}, the CAS latency and the burst length. The total number of clocks for read and write cycles are decided by these factors. In the example shown in the figures, the read cycle takes 10 clocks and the write cycle takes 9 clocks.

The state variable c_State of CMD_FSM is also shown in these figures. Note that the ACTIVE, READ, WRITE commands are asserted one clock after the c_ACTIVE, c_READA, c_WRITEA states respectively.

The values of the region filed with slashes in the system interface input signals of these figures are “don’t care.” For example, signal sys_R_Wn needs to be valid only at the clock before CMD_FSM switches to the c_READA or c_WRITEA states. Depending on the values of t_{RCD} and t_{CK}, this means the signal sys_R_Wn needs to be valid at state c_ACTIVE or the last clock of state c_tRCD.

Figure 6. Read Cycle Timing Diagram

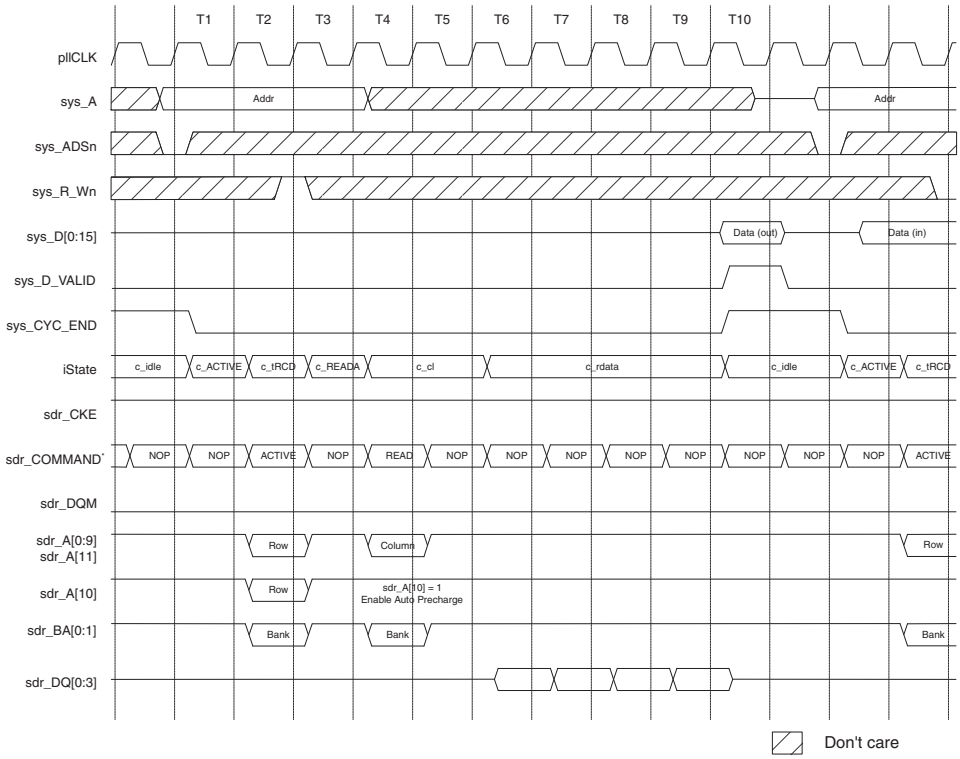
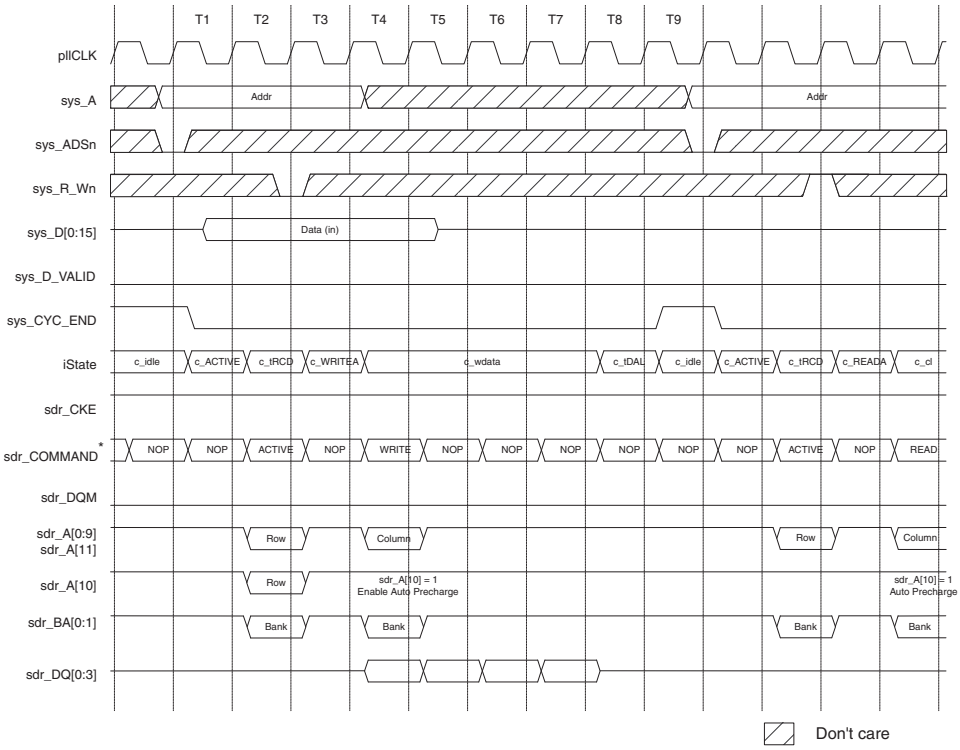


Figure 7. Write Cycle Timing Diagram



Implementation

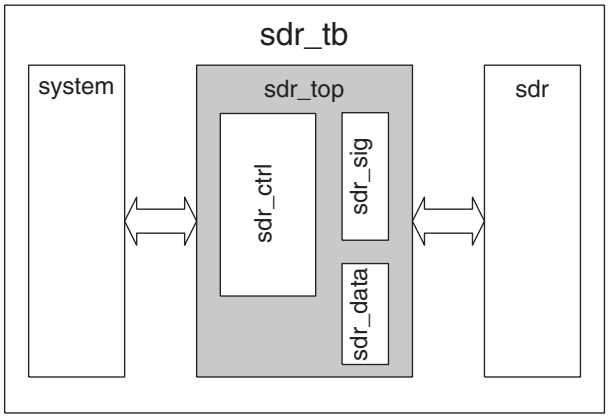
This design is implemented in Verilog and VHDL. When using this design in a different device, density, speed, or grade, performance and utilization may vary. Default settings are used during the fitting of the design.

Table 1. Performance and Resource Utilization

Device Family	Language	Speed Grade	Utilization	f_{MAX} (MHz)	I/Os	Architecture Resources
ECP5™ [9]	Verilog-LSE	-6	92 LUTs	>60	73	N/A
	Verilog-Syn	-6	142 LUTs	>60	73	N/A
	VHDL-LSE	-6	89 LUTs	>60	73	N/A
	VHDL-Syn	-6	125 LUTs	>60	73	N/A
LatticeECP3™ [3]	Verilog-Syn	-7	154 LUTs	>60	73	N/A
	VHDL-Syn	-7	146 LUTs	>60	73	N/A
LatticeECP™ [4]	Verilog-Syn	-5	150 LUTs	>60	73	N/A
	VHDL-Syn	-5	184 LUTs	>60	73	N/A
MachXO2™ [1]	Verilog-LSE	-5	92LUTs	>60	73	N/A
	Verilog-Syn	-5	113 LUTs	>60	73	N/A
	VHDL-LSE	-5	89 LUTs	>60	73	N/A
	VHDL-Syn	-5	104 LUTs	>60	73	N/A
MachXO™ [2]	Verilog-LSE	-3	91 LUTs	>60	73	N/A
	Verilog-Syn	-3	113 LUTs	>60	73	N/A
	VHDL-LSE	-3	87 LUTs	>60	73	N/A
	VHDL-Syn	-3	104 LUTs	>60	73	N/A
Lattice XP2™ [5]	Verilog-Syn	-5	136 LUTs	>60	73	N/A
	VHDL-Syn	-5	189 LUTs	>60	73	N/A
Lattice XP™ [6]	Verilog-Syn	-5	150 LUTs	>60	73	N/A
	VHDL-Syn	-5	184 LUTs	>60	73	N/A
ispMACH® 4000ZE[7]	Verilog	-5 ns	84 Macrocells	>100	73	N/A
	VHDL	-5 ns	84 Macrocells	>100	73	N/A
ispLSI® 5000VE[8]	Verilog	155 MHz	84 Macrocells	>100	73	N/A
	VHDL	155 MHz	84 Macrocells	>100	73	N/A

1. Performance and utilization characteristics are generated using LCMXO2-1200HC-5TG144C, with Lattice Diamond® 3.3 design software.
2. Performance and utilization characteristics are generated using LCMXO256C-3T100C, with Lattice Diamond 3.3 design software.
3. Performance and utilization characteristics are generated using LFE3-95EA-7FN1156C, with Lattice Diamond 3.3 design software.
4. Performance and utilization characteristics are generated using LFECP33E-5F484C with Lattice Diamond 3.3 design software.
5. Performance and utilization characteristics are generated using LFXP2-5E-5FT256C, with Lattice Diamond 3.3 design software.
6. Performance and utilization characteristics are generated using LFXP20C-5F484C with Lattice Diamond 3.3 design software.
7. Performance and utilization characteristics are generated using LC4256ZE-5TN100C with Lattice ispLEVER® Classic 1.4 software.
8. Performance and utilization characteristics are generated using ispLSI5512VE-155LB272 with Lattice ispLEVER Classic 1.4 software.
9. Performance and utilization characteristics are generated using LFE5U-45F-6MG285C, with Lattice Diamond 3.3 design software.

Figure 8. Module Relationships



Technical Support Assistance

e-mail: techsupport@latticesemi.com

Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
—	—	Previous Lattice releases.
July 2009	04.2	Added support for the ispMACH 4000ZE CPLD family.
October 2009	04.3	Added support fo VHDL language.
February 2010	04.4	Added support for LatticeXP2 device family.
November 2010	04.5	Added support for the MachXO2 device family.
		Updated to support Lattice Diamond 1.1 design software.
		Updated to support ispLEVER 8.1 SP1 design software.
April 2011	04.6	Added support for LatticeECP3 device family.
March 2014	04.7	Updated Table 1, Performance and Resource Utilization. - Added support for ECP5 device family. - Added support for Lattice Diamond 3.1 design software.
		Updated corporate logo.
		Updated Technical Support Assistance information.
September 2014	04.8	Updated Table 1, Performance and Resource Utilization. - Added support for Lattice Diamond 3.3 design software. - Added LSE support for LatticeXO and LatticeXO2.
		- Added Synplify support for ECP5.