LatticeMico SDR SDRAM Controller

The LatticeMico SDR SDRAM controller has a WISHBONE slave port to enable the WISHBONE master in the platform to gain access to the SDRAM memory.

Version

This document describes the 3.9 version of the LatticeMico SDR SDRAM controller.

Features

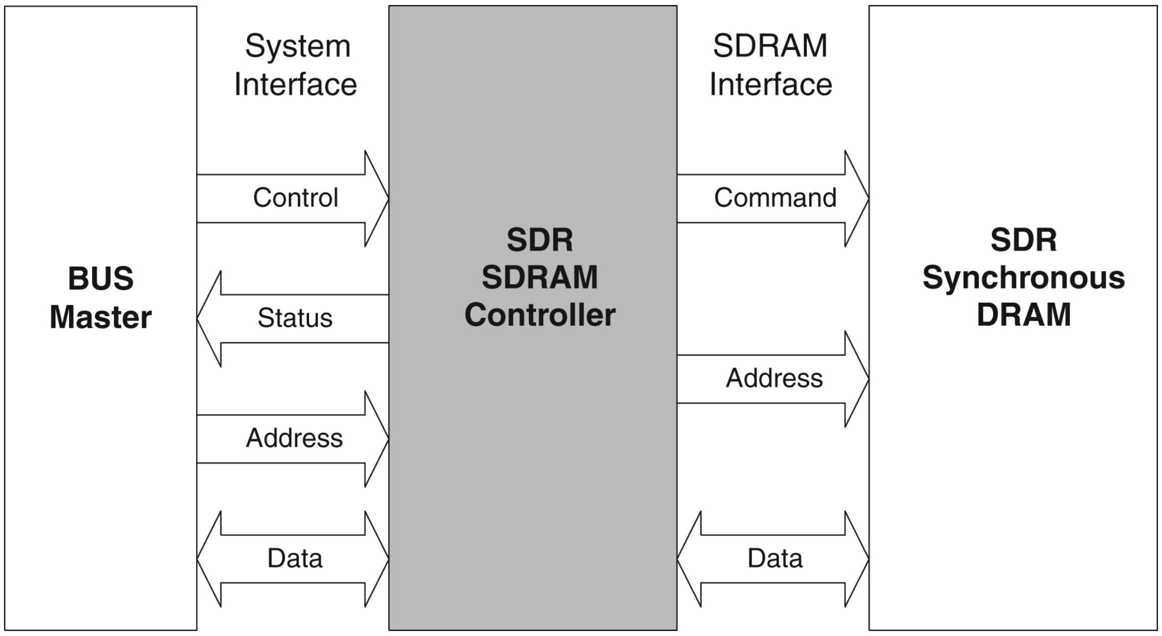
The LatticeMico SDR SDRAM controller includes the following features:

- WISHBONE B.3 interface
- Configurable SDRAM data bus width of up to 32 bits
- Configurable SDRAM row address width of up to 14 bits
- Configurable SDRAM column address width of up to 14 bits
- Configurable bank address width of up to 4 bits
- Configurable timing delays for the PRECHARGE, AUTO REFRESH, and LOAD MODE REGISTER commands
- Configurable timing delays for ACTIVE and 100-µs delay
- Support for linear increment burst-mode transfer
- Internal state machine built for SDRAM power-on initialization
- Support for MachXO2™, LatticeXP™, LatticeECP™, LatticeEC™, LatticeECP2M™, LatticeXP2™, and LatticeSC™ devices

- Slower system clock when you take advantage of the sysCLOCK™ PLL feature. The system interface clock does not need to be the same as the SDRAM clock.
- System interface that can be in any I/O standards supported by this feature, with the support of the sysIO™ feature on MachXO2, LatticeXP, LatticeECP/EC, LatticeECP2, Lattice ECP2M, LatticeXP2, and LatticeSC devices

The SDRAM controller, located between the SDRAM and the bus master, reduces the effort required to use the SDRAM command interface by providing a system interface to the WISHBONE bus master. Figure 1 shows the place of the controller between the WISHBONE bus master and the SDRAM controller. The bus master can be either a microprocessor or your proprietary module interface.

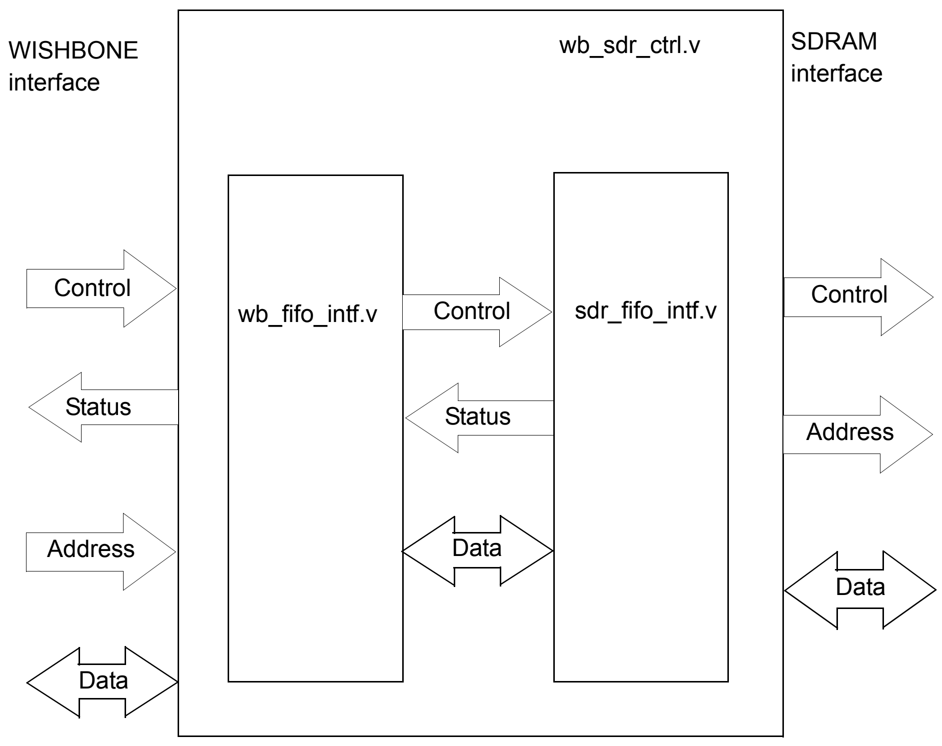
Figure 1: SDR SDRAM Controller System



Functional Description

The functional block diagram of the SDRAM controller is shown in Figure 2.

Figure 2: Block Diagram of the SDR SDRAM Controller

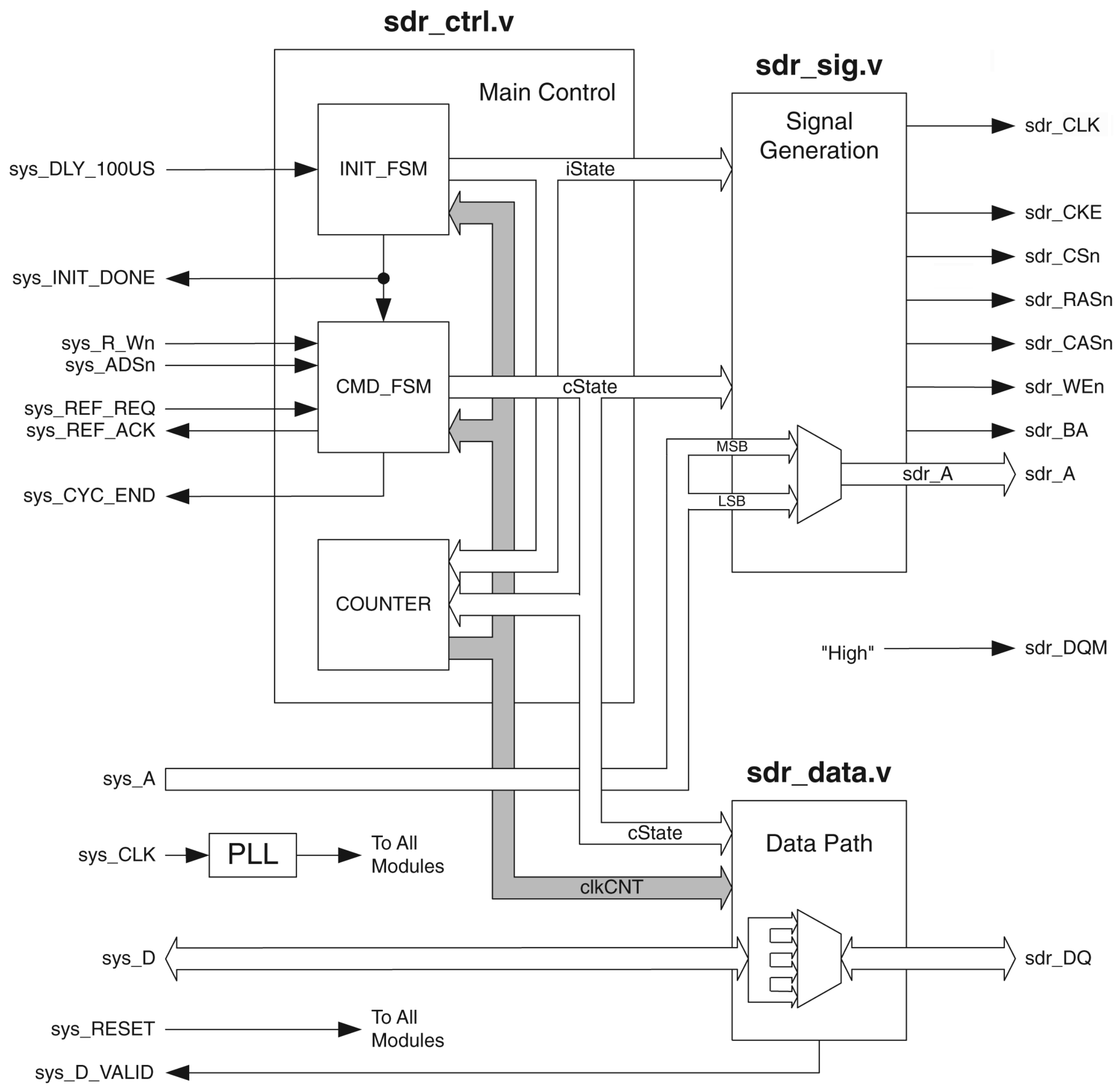


The WISHBONE interface block puts the incoming WISHBONE command in the WISHBONE FIFO. The SDRAM interface block consists of three modules, as shown in Figure 3:

- Main control module – Contains two state machines and a counter. It is the primary module of the design. It generates proper iState and cState outputs according to the system interface control signals.
- Signal generation module – Generates the address and command signals required for the SDRAM according to iState and cState.

- Data path module – Performs the data latching and dispatching of the data between the bus master and the SDRAM.

Figure 3: Block Diagram of the SDR SDRAM Interface Block



PLL

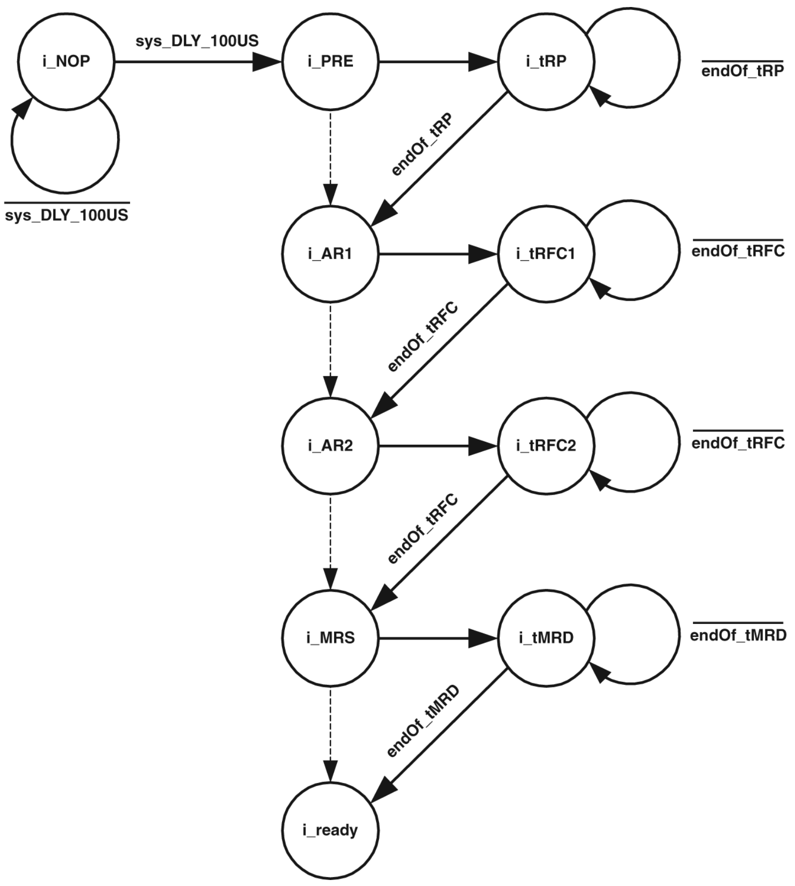
The SDRAM clock is generated by the internal PLL of the device. For example, if the system is running at 40 MHz, you can obtain a 100-MHz SDRAM clock through the dedicated PLL output pin by setting the proper PLL attributes (multiply = 5 and divide = 2).

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

SDRAM Initialization

Before it can perform normal memory accesses, the SDRAM must be initialized by a sequence of commands. The INIT_FSM state machine handles this initialization. Figure 4 shows the state diagram of the INIT_FSM state machine. During reset, INIT_FSM is forced to the i_NOP state. After reset, the sys_100µs signal is sampled at the rising edge of every PLL clock cycle to determine if the 100-µs power and clock stabilization delay is completed. After the power and clock stabilization is complete, the SDRAM initialization sequence begins, and INIT_FSM switches from the i_NOP to the i_PRE state. The initialization starts with the PRECHARGE command, followed by two AUTO REFRESH commands, 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 must be satisfied before you can issue any command other than NOP. These timing delays are t_{RP}, t_{RFC}, and t_{MRD} for the PRECHARGE, AUTO REFRESH, and LOAD MODE REGISTER commands, respectively. After the LOAD MODE REGISTER command is issued and the t_{MRD} timing delay is satisfied, INIT_FSM goes to the i_ready state and remains there for the normal memory access cycles unless sys_RESET is asserted. Also, the sys_INIT_DONE signal is set to high to indicate that SDRAM initialization is complete.

Figure 4: INIT_SFM State Diagram



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, and so forth. 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 the idle state by 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 is not changed after the initialization. This design assumes that the mode register stays the same after initialization and that a fixed-mode register content is implemented in the HDL code.

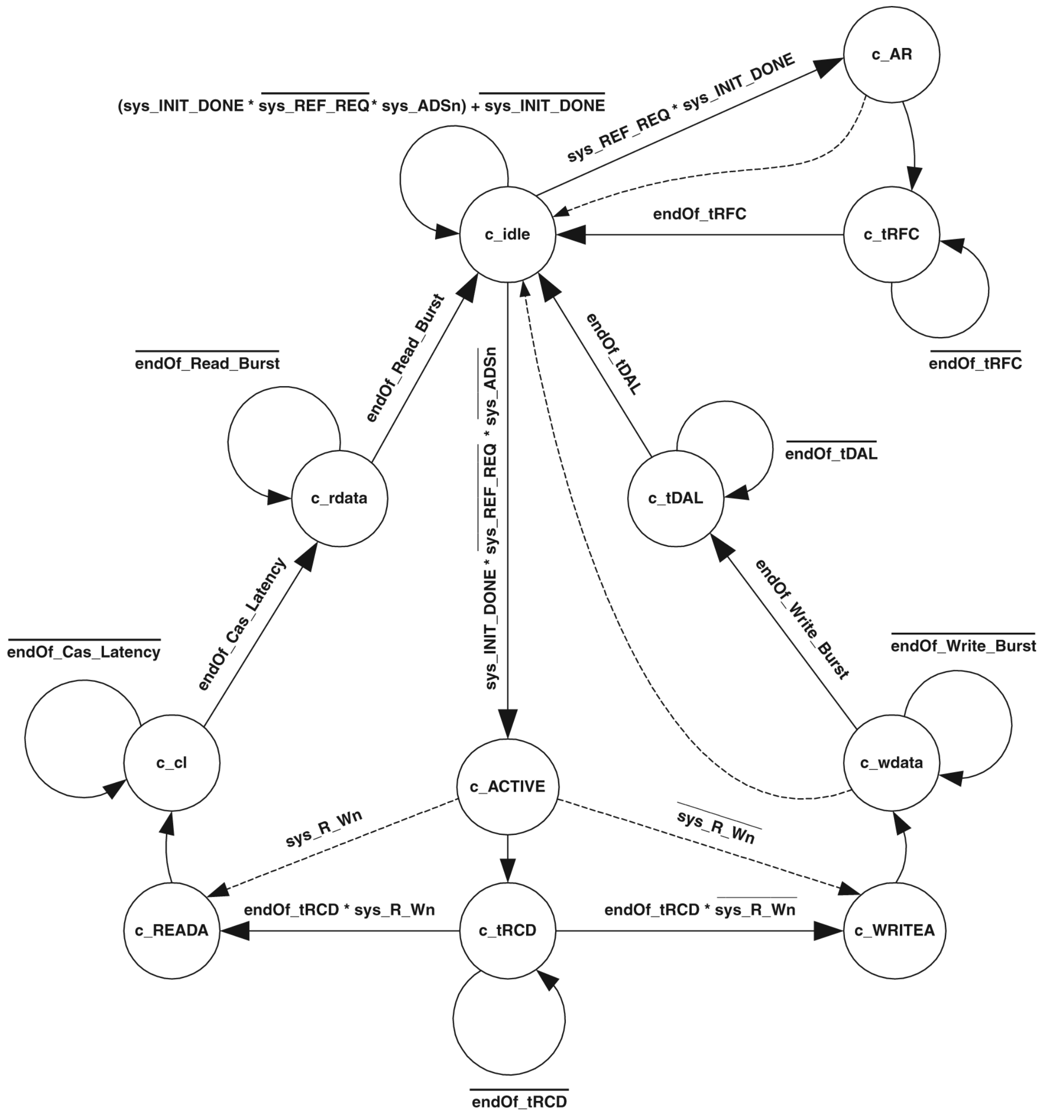
As noted earlier, certain timing delays (t_{RP}, t_{RFC}, and t_{MRD}) must be satisfied before you can issue another non-NOP command. These SDRAM delays vary from speed grade to speed grade and sometimes from vendor to vendor.

According to these timing values, the number of clocks in which the state machine stays in the i_t_{RP}, i_t_{RFC1}, i_t_{RFC2}, and i_t_{MRD} states is determined. In cases where t_{CK} is larger than the timing delay, the state machine does not need to switch to the timing delay states and can go directly to the command states. The dashed lines in Figure 4 show the possible state switching paths.

Read/Write Cycle

Figure 5 shows the CMD_FSM state diagram, 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 that the SDRAM initialization sequence is not yet complete. Once the initialization is done, sys_ADS*n* and sys_REF_REQ are sampled at the rising edge of every clock cycle. A logic high sampled on sys_REF_REQ starts a SDRAM refresh cycle, which is described in "Refresh Cycle" on page 9. If a logic low is sampled on both sys_REF_REQ and sys_ADS*n*, a system read cycle or a system write cycle will begin. These system cycles are composed of a sequence of SDRAM commands.

Figure 5: CMD_FSM State Diagram


(sys_INIT_DONE * $\overline{\text{sys\_REF\_REQ}}$ * sys_ADSn) + $\overline{\text{sys\_INIT\_DONE}}$
sys_REF_REQ * sys_INIT_DONE
c_AR
c_idle
endOf_tRFC
c_tRFC
$\overline{\text{endOf\_tRFC}}$
$\overline{\text{endOf\_Read\_Burst}}$
endOf_Read_Burst
sys_INIT_DONE * $\overline{\text{sys\_REF\_REQ}}$ * $\overline{\text{sys\_ADSn}}$
endOf_tDAL
$\overline{\text{endOf\_tDAL}}$
c_rdata
c_tDAL
endOf_Cas_Latency
endOf_Write_Burst
$\overline{\text{endOf\_Cas\_Latency}}$
$\overline{\text{endOf\_Write\_Burst}}$
c_cl
c_ACTIVE
c_wdata
sys_R_Wn
$\overline{\text{sys\_R\_Wn}}$
c_READA
endOf_tRCD * sys_R_Wn
c_tRCD
endOf_tRCD * $\overline{\text{sys\_R\_Wn}}$
c_WRITEA
$\overline{\text{endOf\_tRCD}}$

As with the FP and EDO DRAM, the row address and the column address are required to pinpoint the memory cell location of the SDRAM access. Since the SDRAM is composed of four banks, you must provide the bank address 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 must 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, at most four rows can be 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. The ACTIVE command opens the rows and the PRECHARGE command (or the AUTO PRECHARGE command hidden in the WRITE and READ commands, as used in this design) closes the rows. When you issue the commands for opening or closing the rows, you must provide both the row address and the bank address.

In this design, the ACTIVE command is issued for each read or write access to open the row. After the t_{RCD} delay is satisfied, a READ or WRITE command is issued with a high sdr_A[10] to enable AUTO REFRESH to close the row after access. Therefore, the clocks required for the read and the 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 the t_{RCD} 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 the SDRAM to bus master. The number of clocks in which 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 the bus master to the SDRAM, then switches to c_tDAL. As with the read cycle, the number of clocks in which the state machine stays in c_wdata state is determined by the burst length. The t_{DAL} time delay is the sum of WRITE recovery time, t_{WR}, and the AUTO PRECHARGE timing delay, t_{RP}. After the rising clock edge of the last data in the burst sequence, no commands except NOP can be issued to the SDRAM before t_{DAL} is satisfied.

As noted in the “SDRAM Initialization” on page 5, the dashed lines indicate possible state switching paths when the t_{CK} period is longer than timing delay specification.

Refresh Cycle

As with the other DRAMs, the SDRAM requires a memory refresh. An SDRAM refresh request is generated by activating the controller’s sdr_REF_REQ signal. The sdr_REF_ACK signal acknowledges the recognition of sdr_REF_REQ and is active throughout the whole refresh cycle. The sdr_REF_REQ signal must be maintained until the sdr_REF_ACK signal goes active in order to be recognized as a refresh cycle. No system read and write access cycles are allowed when sdr_REF_ACK is active. All system interface cycles are ignored during this period. The sdr_REF_REQ signal assertion must be removed when the sdr_REF_ACK acknowledgement is received; otherwise, another refresh cycle is again performed.

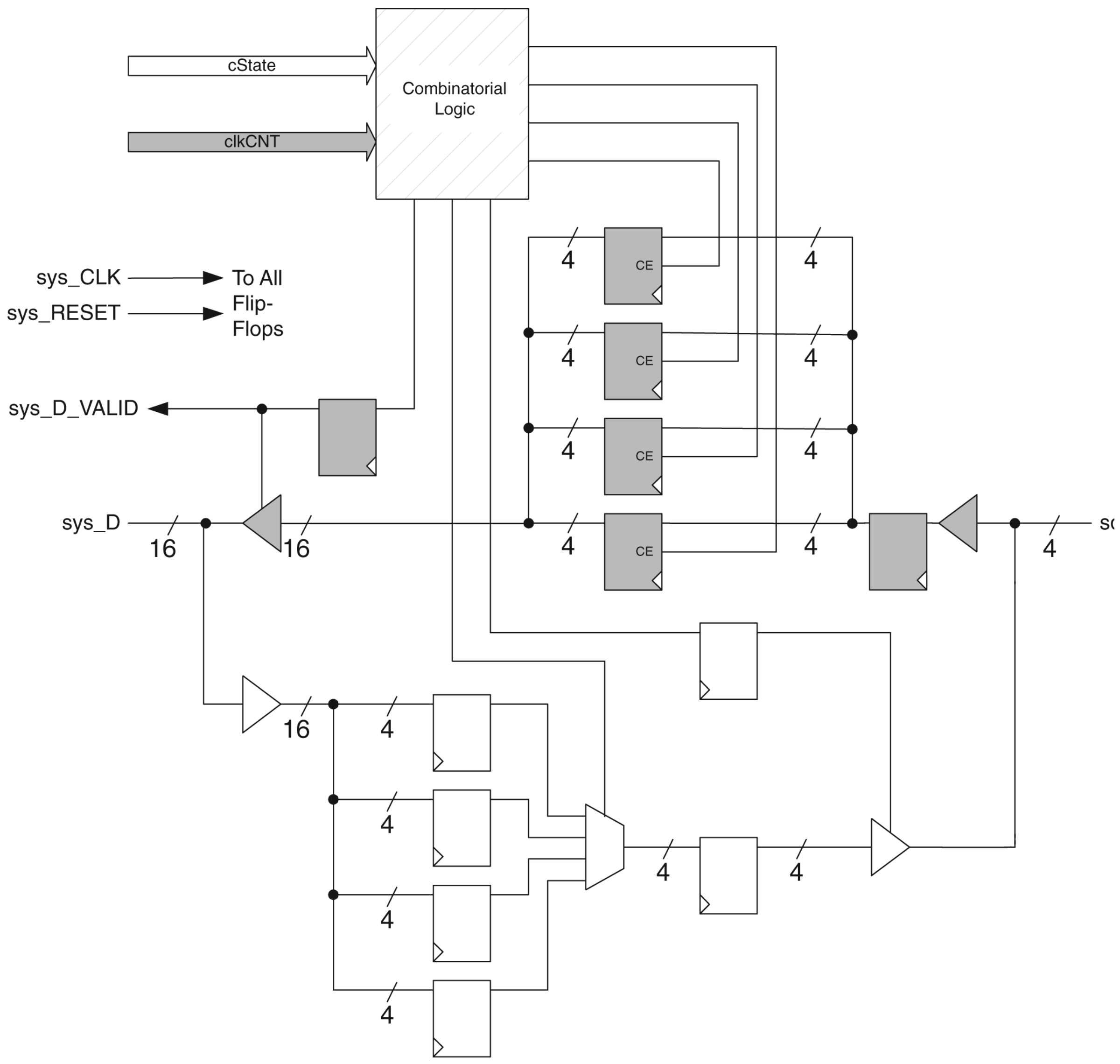
When the sdr_REF_REQ assertion is received, the CMD_FSM state machine enters the c_AR state to issue an AUTO REFRESH command to the SDRAM. After the t_{RFC} time delay is satisfied, CMD_FSM returns to c_idle.

Data Path

Figure 6 shows the data flow design between the SDRAM and the system interface. The module in this design interfaces between the SDRAM with a 4-bit data bus and the bus master with a 16-bit data bus.

The size of each bus in Figure 6 is shown by the number under the slash across the bus. The components shown in gray are for read cycles, and the components shown in white are for write cycles.

Figure 6: Data Path Module



Configuration

The following sections describe the graphical user interface (UI) parameters and the I/O ports that you can use to configure and operate the LatticeMico SDR SDRAM controller.

UI Parameters

Table 1 shows the UI parameters available for configuring the LatticeMico SDR SDRAM controller through the Mico System Builder (MSB) interface.

Table 1: SDR SDRAM Controller UI Parameters

Dialog Box Option	Description	Allowable Values	Default Value
Instance Name	Specifies the name of the SDRAM controller instance.	Alphanumeric and underscores	sdram
Base Address	Specifies the base address.	0X00000000–0XFFFFFFFF	0X00000000
SDRAM Data Size	Specifies the width of the memory's data bus.	4, 8, 16, 32	16
SDRAM Frequency (MHz)	Specifies the frequency of the SDRAM clock, in megahertz.		100
SDR Row Width	Specifies the width of the memory's row address.	12–13	12
SDR Col Width	Specifies the width of the memory's column address.	8–12	8
SDR Bnk Width	Specifies the width of the memory's bank address.	1–2	2
Size	Specifies the size of the internal SDRAM memory, in bytes.	16384–134217728	1048576
AC Timing			
SDR_TMRD	Specifies the number of SDRAM clock cycles for the LOAD MODE REGISTER command.		2
SDR_TRP	Specifies the number of SDRAM clock cycles for the PRECHARGE command.		3
SDR_TRFG	Specifies the number of SDRAM clock cycles for the AUTO REFRESH command during the SDRAM initialization phase.		7
SDR_TRCD	Specifies the number of SDRAM clock cycles for the ACTIVE command.		3
SDR_TDAL	Specifies the number of SDRAM clock cycles for the WRITE recovery time plus the AUTO PRECHARGE time.		5

Table 1: SDR SDRAM Controller UI Parameters

Dialog Box Option	Description	Allowable Values	Default Value
SDR_TREF1	Specifies the number of SDRAM clock cycles loaded into the AUTO REFRESH counter (after this number of clock cycles, the SDRAM memory goes into AUTO REFRESH mode).		1564
T 100 µs	Defines the 100-µs power/clock stabilization delay.		10000

I/O Ports

Table 2 describes the input and output ports of the LatticeMico SDR SDRAM controller.

Table 2: SDR SDRAM Controller I/O Ports

I/O Port	Active	Direction	Initial State	Description
WISHBONE Slave Port				
S_ADR_I	High	I	X	Slave address bus
S_DAT_I	High	I	X	Slave data input bus
S_WE_I	High	I	X	Slave write enable signal
S_STB_I	High	I	X	Slave strobe signal
S_CYC_I	High	I	X	Slave cycle signal
S_SEL_I	High	I	X	Slave select signal
s_LOCK_I	High	I	X	Slave lock signal
S_CTI_I	High	I	X	Slave CTI signal
S_BTE_I	High	I	X	Slave BTE signal
S_DAT_O	High	O	0	Output data bus
S_ACK_O	High	O	0	Acknowledge to master device
S_ERR_O	High	O	0	Slave error signal
S_RTY_O	High	O	0	Slave retry signal
S_LOCK_O	High	I	X	Slave lock signal
Clock and Reset Signal				
CLK_I	High	I	X	Input clock signal
RST_I	High	I	X	Reset signal (active high)
SDRAM Port				
sdr_DQ	High	I/O	0	SDRAM data bus

Table 2: SDR SDRAM Controller I/O Ports

I/O Port	Active	Direction	Initial State	Description
sdr_A	High	O	0	SDRAM address bus
sdr_BA	High	O	0	SDRAM bank address
sdr_CLK	High	O	0	SDRAM clock
sdr_CKE	High	O	0	SDRAM clock enable
sdr_CSn	Low	O	0	SDRAM command inputs CS#
sdr_RASn	Low	O	0	SDRAM command inputs RAS#
sdr_CASn	Low	O	0	SDRAM command inputs CAS#
sdr_Wen	Low	O	0	SDRAM command inputs WE#
sdr_DQM	High	O	0	SDRAM data bus mask

Timing Diagrams

Figure 7 and Figure 8 are the read cycle and write cycle timing diagrams of the design with three CAS latency cycles. The timing diagrams may be different because of the values of the t_{MRD}, t_{RP}, t_{RFC}, t_{RCD}, or t_{WR} timing delays; the t_{CK} clock period; and the CAS latency. The total number of clocks for read and write cycles are decided by these factors.

Figure 7: SDR SDRAM Read

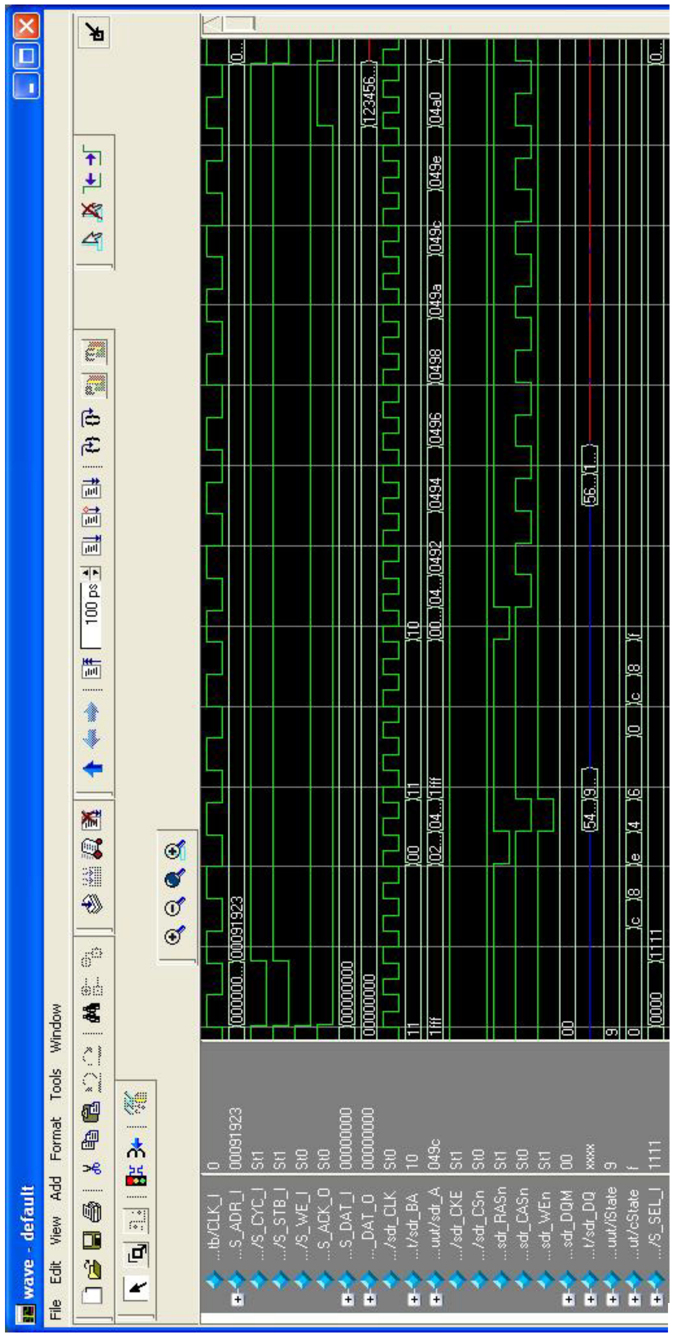
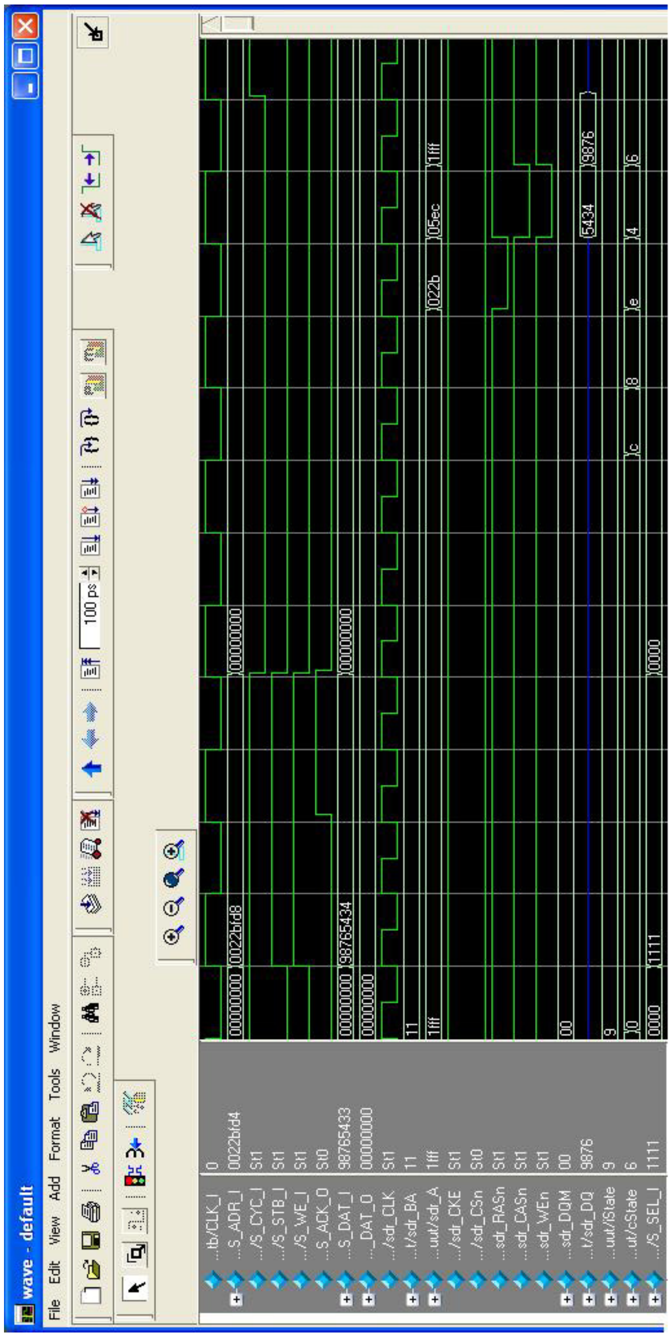


Figure 8: SDR SDRAM Write



EBR Resource Utilization

The LatticeMico SDR SDRAM has no EBRs.

Implementation

Table 3 shows the implementation of the LatticeMico SDR SDRAM controller.

Table 3: Implementation of the LatticeMico SDR SDRAM Controller

Device	Resources Used	Maximum Clock Frequency[1]
LFECP10EFPBGA484-5	634 LUTs	101.916
LFE2-35EFPBGA484-7	622 LUTs	114.811
LFXP2-17EFPBGA484-7	621 LUTs	129.938
LFXP20CFPBGA484-5	634 LUTs	84.753
LFSC3GA25EFPBGA900-7	635 LUTs	212.45

1. The maximum clock frequency is obtained by performing a timing analysis with the Lattice Semiconductor design software.

SDR SDRAM Clock Network Distribution

This section describes the clock network distribution guidelines for the SDR SDRAM controller.

Root Cause of the Problem

In the SDR SDRAM controller design, the SDRAM clock is generated from other main clocks, such as the CPU clock, so the SDRAM clock is used both internally by the FPGA logic and externally by the SDRAM chip on the board. In the SDR SDRAM, a PIO register is used for the SDRAM's output signals. The clock-to-Q value (T_{CO}) from the internal clock net to the pad output is very small (around 2 ns for a LatticeECP2 device with a -6 speed grade). If the SDRAM clock goes off-chip, the clock signal must be routed from the primary clock network to the PIO. It will have a long routing delay, which in most cases will be larger than the T_{CO} just given. The result, if you check the T_{CO} externally by subtracting T_{CLK_ROUTE} from the internal T_{CO}, is a minus value and violates the SDRAM chip's setup-(T_{SU})-and-hold-(T_H) time requirement, so the design will fail to run.

PLL Phase Shift Solution

Use the PLL's output port, CLKOS, to generate the clock phase-shifted by 90 degrees to drive the ODDR primitive. The ODDR primitive's output drives the SDRAM clock, as shown in the example in Figure 9.

Figure 9: Using the PLL Output Port to Generate SDRAM Clock Output

```
pmi_pll #(.pmi_freq_clki(25),
          .pmi_freq_clkop(100),
          .pmi_freq_clkos(100),
          .pmi_family(`LATTICE_FAMILY),
          .pmi_phase_adj(90),
          .pmi_duty_cycle(50),
          .pmi_clkfb_source("CLKOP"),
          .pmi_fdel("OFF"),
          .pmi_fdel_val(0),
          .module_type("pmi_pll"))
 U1_pmi_pll (.CLKI(CLK_I),
  .CLKFB(sdr_clk_c),
  .RESET(reset_i),
  .CLKOP(sdr_clk_c),
  .CLKOS(sdr_clk_io));
 ODDRXB sdr_clk_inst (
  .CLK (sdr_clk_io),
  .DA(1'b0),
  .DB(1'b1),
  .LSR(1'b0),
  .Q(sdr_CLK));
```

In Figure 9, the PLL generates two clock outputs: the sdr_clk_c drives the internal FPGA logic, and the other CLKOS output with a 90-degree phase shift (if the clock frequency is 100 MHz, it is equivalent to a 2.5-ns delay) drives the PIO register and feeds directly into the ODDR primitive to generate the SDRAM clock.

Revision History

Component Version	Description
1.0	Initial release.
3.0 (7.0 SP2)	Added burst read support.
3.1	Added burst write support.
3.2	Added PLL to generate SDRAM clock. Used I/O register for all SDRAM interface signals.
3.3	PLL generation fixed.
3.4	ECP3 support and read burst fixes.
3.5	ECP3-E silicon clock generation fix.
3.6	Fixed issues with synthesis.

Revision History (Continued)

Component Version	Description
3.7	Support added for LatticeECP3 -E and -EA silicon. The -EA silicon is default. Updated document with new corporate logo.
3.8	▶ Added support for LatticeMico8-based designs in addition to LatticeMico32-based designs. ▶ Component can be used in designs that do not include a processor.
3.9	▶ Added support for MachXO2 device family.

.