SDRAM Controller User Guide

Application Note

General Description

The SDRAM Controller provides a high performance interface to SDR (Single Data Rate) SDRAM devices. The controller is optimized to archive high frequency operation with minimal constraints. A simple wishbone slave interface offers an easy integration with other chip logic such as microcontroller buses and FIFOs. The controller is provided with configuration settings for timing parameters (tCL, tRAS, tRFC, tRCD, tRP, tREF, tWR) as well as memory configuration settings. This ensures compatibility with any SDRAM configuration.

Feature

1. Support industry standard SDRAM chips and DIMMs in configurable width, support 4 kinds types of width configuration，8，16，32，64.
2. Configurable timing parameters - tCL, tRAS, tRCD, tRP, tREF,tRFC, tWR.
3. Configurable memory settings, support maximum 8 chip selects, and maximum 4 memory banks
4. Automatic generation of initialization and refresh sequences
5. Support burst read/write mode (1,2,4,8 and full page)
6. Support Auto-Precharge option to enable low random access latencies
7. Compatible with Wishbone RevB.3 Classic interface as slave

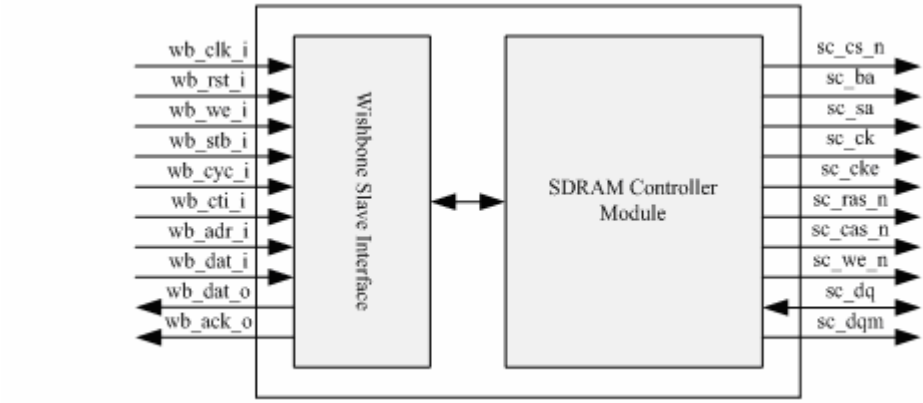


Figure 1 SDRAM Pin Diagram

Core I/O Signals

The core signal I/O has not been fixed to specific device pins to provide flexibility for interfacing with user logic. The descriptions of all signal I/Os are provided in Table 1.

Table 1 Core I/O Signals.

Signal	Signal Direction	Description
Wishbone Slave Interface Signals		
wb_clk_i	Input	System clock
wb_rst_i	Input	System reset, active high
wb_adr_i(m-1:0)	Input	Requested memory address m-bit, m = c_sdram_addr_width
wb_dat_i(n-1:0)	Input	Data input from WB Master ,n = c_sdram_data_width
wb_dat_o(n-1:0)	Output	Data output to WB Master ,n = c_sdram_data_width
wb_we_i	Input	Write request signal
wb_cyc_i	Input	Valid Bus Cycle
wb_ack_o	Output	Normal Bus Cycle termination (Indicate the write operation ends or data is ready for a read operation)
wb_stb_i	Input	Strobe/Core select
wb_cti_i	Input	Indicate the end of BURST(TAG_TYPE)
SDRAM Interface Signals		
sc_sa	Output	SDRAM address bus
sc_ba	Output	SDRAM bank address
sc_cs_n	Output	SDRAM chip selects
sc_cke	Output	SDRAM clock enable, hold low by controller during reset to ensure DQ and DQS are hi-z
sc_ras_n	Output	Row address strobe command output
sc_cas_n	Output	Column address strobe command output
sc_we_n	Output	Write enable command output
sc_dq	Bidirectional	Data bus
sc_dqm	Output	Data mask to SDRAM device(s)

Note: c_sdram_addr_width means the total address range of external SDRAM chips. Normally it can be calculated by: c_sdram_addr_width = log2(c_sdram_CS) + c_sdram_bank_width + c_sdram_row_width + c_sdram_col_width. The following Table 2 shows you the description of parameters mentioned above.

External SDRAM Connection Figure

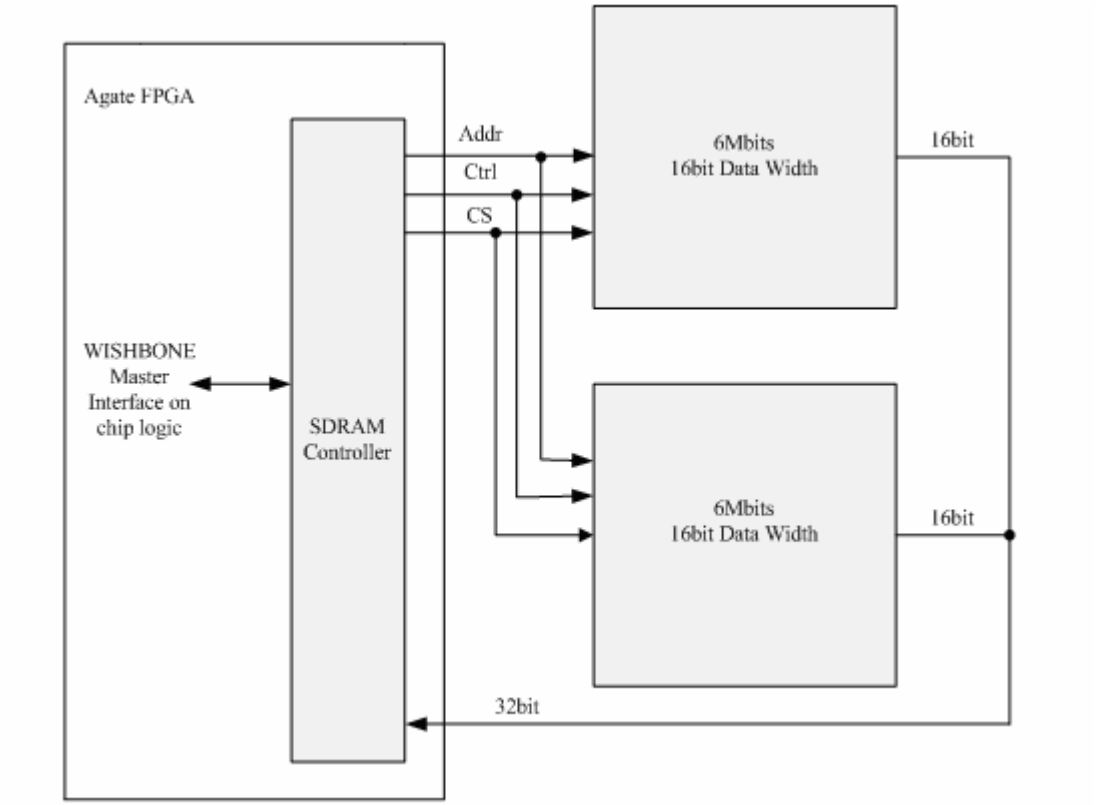


Figure 2 Extended SDRAM data width connection

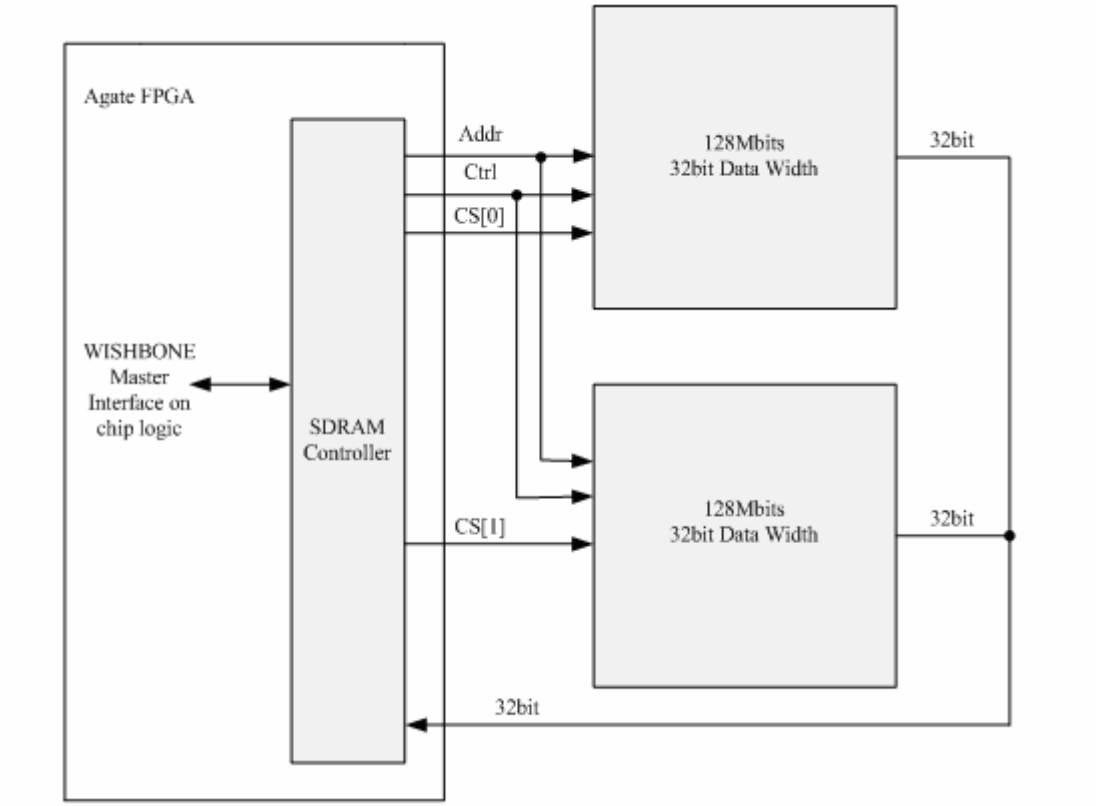


Figure 3 Extended SDRAM address width connection

Table 2 Agate Logic SDRAM Controller Design Parameters

Feature descriptions	Parameter Name	Allowable Values	Default Value
Write Recovery Time (ns)	c_sdram_tWR	<1000ns	15
Delay after ACTIVE command before PRECHARGE command (ns)	c_sdram_tRAS	<1000ns	45
Delay after AUTOREFRESH before another command(ns)	c_sdram_tRFC	<1000ns	70
Delay after ACTIVE command before READ/WRITE command (ns)	c_sdram_tRCD	<1000ns	20
Delay after a PRECHARGE command (ns)	c_sdram_tRP	<1000ns	20
Refresh command interval (ms)	c_sdram_tREF	>2ms	64
CAS Latency(tCK)	c_sdram_tCL	2,3	2
SDRAM data width	c_sdram_data_width	8,16,32,64	32
SDRAM row address width	c_sdram_row_width	11,12,13,14	13
SDRAM column address width	c_sdram_col_width	8,9,10,11,12,13,14	10
SDRAM bank address width	c_sdram_bank_width	2,4	4
SDRAM Chip select	c_sdram_CS	1,2,4,8	1
SDRAM initialization time (us)	c_sdram_init_time	<500us	100
SDRAM initialization auto refresh number	c_sdram_init_afrnum	1 – 15	2
Burst Length	c_sdram_BL	0,1,2,3,full-page	1
Burst Type	c_sdram_BT	0,1	0
Sdram Write Burst	c_sdram_wbmode	0,1	0
System Clk	sys_clk	>10Mhz and <166Mhz	100Mhz
Optimized address distruibution*	mem_dist	0,1	1
Auto precharge select*	mem_opway	0,1	1

***** This SDRAM controller has been optimized to achieve the highest bandwidth. Each time a different row is selected huge access penalties are imposed. To sustain high bandwidth over longer periods of time, it is therefore desirable to have very large row sizes. This can be achieved by selecting SDRAM configurations that naturally provide larger rows sizes (e.g. 8-bit wide SDRAM devices) or by extending the row accesses across multiple banks. This memory controller utilizes this feature. The address bus is divided into column address, row address and bank address. The goal is to minimize the changes in the row address. Therefore this memory controller allows the address to be distributed in two ways. This selector is controlled by mem_dist:

from MSB to LSB: 1 – {Row address, Bank address, Column address}

from MSB to LSB: 0 – {Bank address, Row address, Column address}

Also this SDRAM controller supports Auto-precharge option to enable low random access latencies. This selector is controlled by mem_opway:

1: Keep the current row open when one Read/Write action has completed.

0: Close the current row when one Read/Write action has completed.

Functional Description

The internal components of the SDR SDRAM Controller are shown in Figure 1. The Control and Timing Module generates SDRAM commands in response to requests from the wishbone slave module. The Refresh Control module automatically generates periodic requests to the Control and Timing Module to generate Auto-Refresh commands required by the SDRAM devices. The Initialization Control module performs the initialization sequence required by SDRAM devices, including loading the mode register registered into all the SDRAM devices.

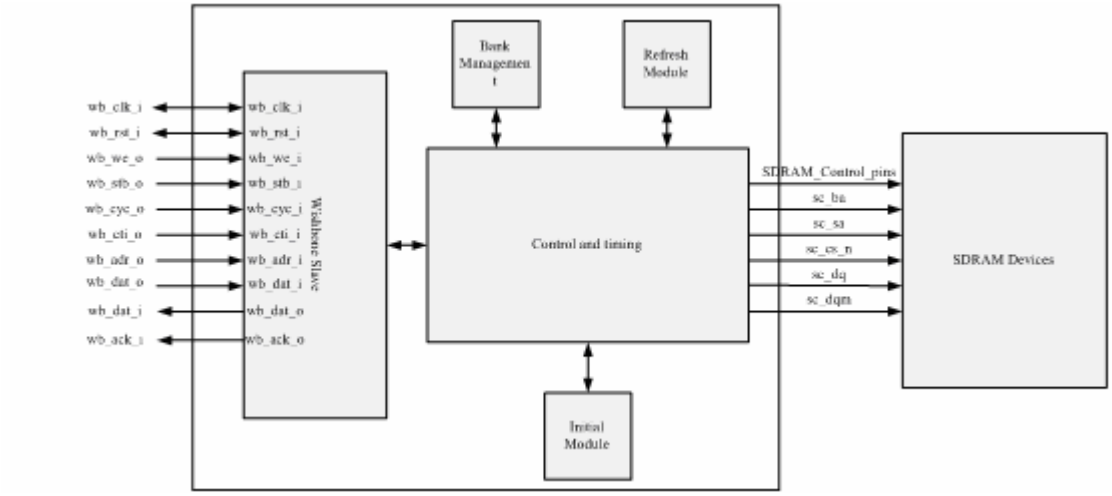


Figure 4 SDR SDRAM Controller Block Diagram

Initial Module

SDRAMs must be powered up and initialized in a predefined manner specified in the SDRAM device data sheet. Once power has been applied and the clock is stable, the SDRAM requires a 100us delay (depends on specified SDRAM chip) prior to issuing any command other than a COMMAND INHIBIT or a NOP.

The Init State Machine provides the 100uS delay and the sequencing of the required SDRAM start-up commands. It instructs the Command State Machine to send the proper commands in the proper sequence to the SDRAM. This state machine starts execution after Reset and returns to the PowerOn state when Reset is applied.

For a typical SDRAM ~300 ns is required after the 100 us reset / power-up time to complete the initialization sequence. During the initialization sequence, the SDRAM Controller will not respond to WISHBONE module. When the initialization sequence has been completed, the INIT_DONE signal asserts.

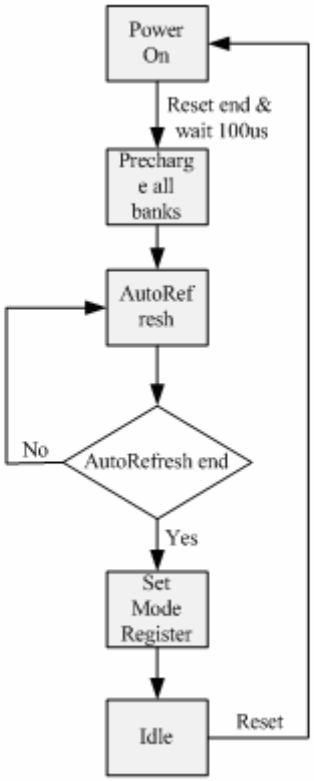


Figure 5

Bank Management module

This module will remember the open or close action to each SDRAM bank, and save the open row address. Then if a read/write action to the open row, it will save the ACTIVE command.

Refresh Control Module

This module will generate the auto_refresh request signal to tell the timing control module do the auto_refresh command. It will calculate the 0.976us as the basic count unit for the auto_refresh timing interval. And in the burst fixed length mode (like burst 4 word), the auto_refresh request will interrupt the user's READ/WRITE command, it will be asserted until the user's command has ended.

SDRAM timing control module

This module is the core of whole IP. It will accept requests from other modules, handle the SDRAM timing sequence, and generate the SDRAM control signals. The state machine also covers the initial module function. A simplified version of the SDRAM data control state machine is shown in the Figure2.
Figure 2: SDRAM data control state machine

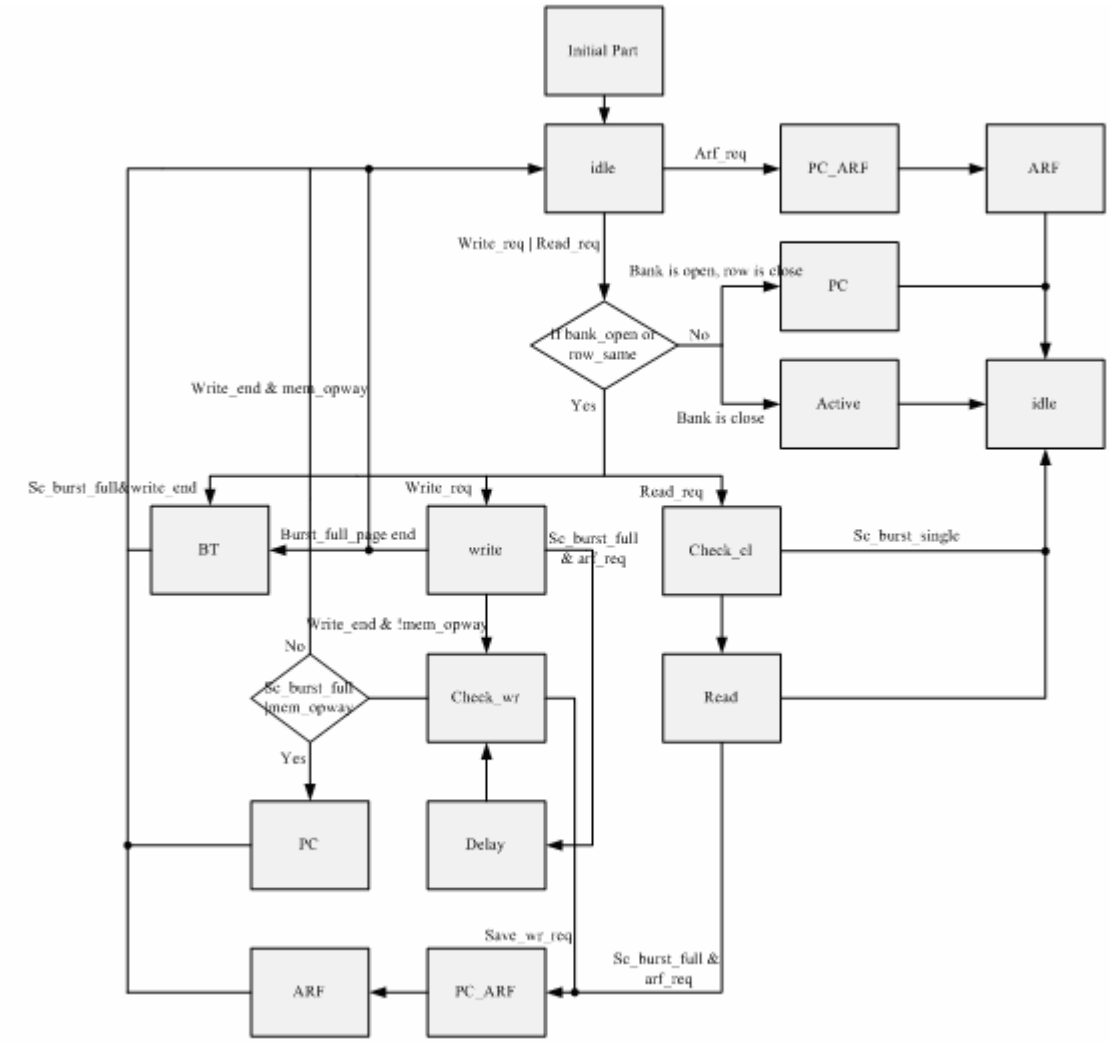


Figure 6

WISHBONE slave module

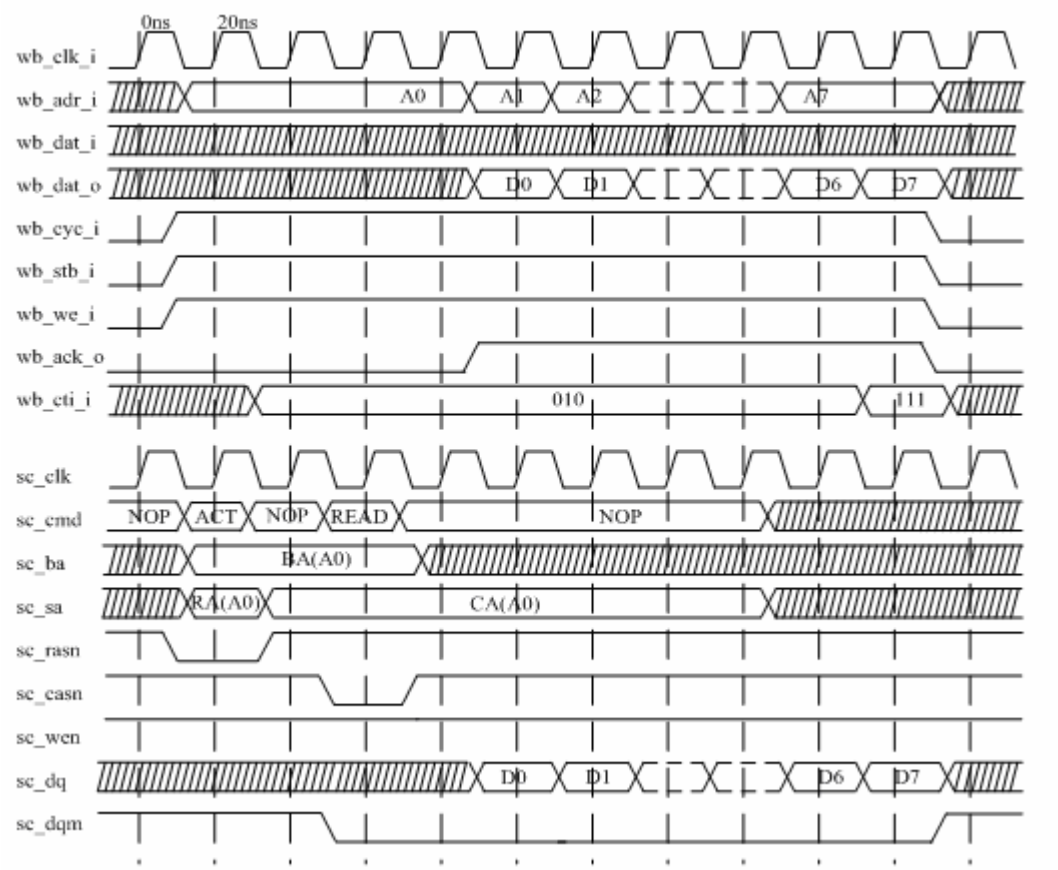
This module works as a simple slave interface compatible with the WISHBONE Rev.B3 protocol. It converts the WISHBONE Master address to the SDRAM address like row address, column address, bank address and chip selects. And a little control logic for feeding the timing control module right address.

This module has different definitions in the burst mode with that in the WISHBONE Rev.B3. In the WISHBONE protocol, this module only supports busrt 2,4,8,16 mode, but in the real SDRAM product, there would be burst 1 and full-page mode. For the burst 1, we can use the classical bus cycle instead, but for the burst full-page mode, there must be another definition.

SDRAM Timing Diagrams

The following figures will show the timing relationships between the Wishbone slave interface and the SDRAM device during various read and write accesses.
If no special notes, the following diagrams use the same parameters:
tCL = 2tCK, tRAS = 44ns, tRCD = 20ns, tRP = 15ns, tWR = 2tCK.

Read_SDRAM_Timing (read 8 words, use the burst bus cycle)



Write_SDRAM_Timing (write 8 words, use the burst bus cycle)

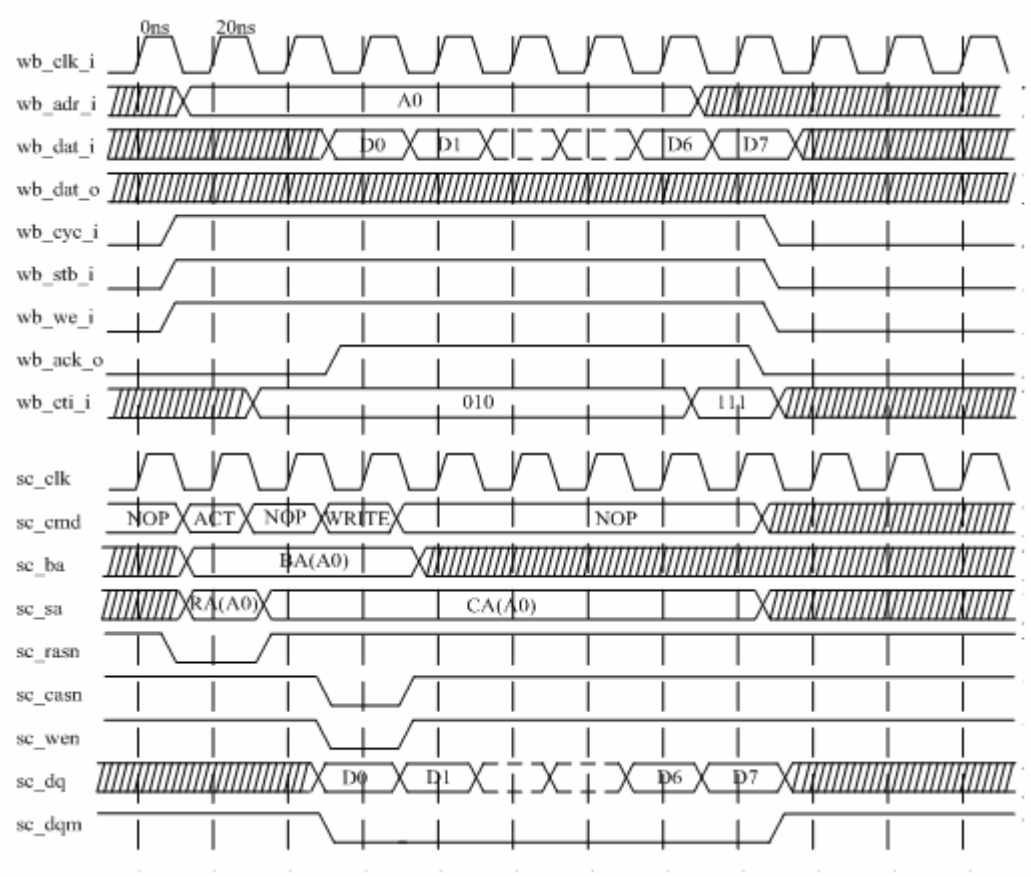


Table 3 SDRAM FPGA Performance and Resource Utilization Benchmarks (Angelo)

Parameters Value		Device Resource	Fmax (MHz)
c_sdram_CS	c_sdram_BL	LE	FmaxMhz
1	0	407	17.476
2	1	517	13.674
4	2	681	12.339
8	3	1066	11.932

Example Wrapper Verilog

```
Module user-defined-module-name (
wb_clk_i,          // System clock
wb_rst_i,          // System reset, active high
wb_adr_i,          // Memory address
wb_dat_i,          // Data input
wb_dat_o,          // Data output
wb_we_i,           // Write request signal, active high
wb_cyc_i,          // Valid Bus Cycle probe, active high
wb_ack_o,          // Normal Bus Cycle termination flag, active high
wb_stb_i,          // Strobe/Core select
wb_cti_i,          // Indicate the end of BURST(TAG_TYPE)
sc_sa,             // SDRAM address
sc_ba,             // SDRAM bank address
sc_cs_n,           // SDRAM chip select
sc_cke,            // SDRAM clock enable
sc_ck,             // SDRAM clock
sc_ras_n,          // SDRAM row address strobe command input
sc_cas_n,          // SDRAM column address strobe command input
sc_we_n,           // SDRAM write enable command input
sc_dq,             // SDRAM data bus
sc_dqm             // SDRAM data mask
);
defparam    sys_clk             = 50;
defparam    c_sdram_CS          = 1;
defparam    mem_opway           = 1;
defparam    mem_dist            = 1;
defparam    c_sdram_BL          = 3'b 000;
defparam    c_sdram_BT          = 1'b 0;
defparam    c_sdram_tCL         = 3'b 011;
defparam    c_sdram_wbmode      = 1'b 0;
defparam    c_sdram_tRFC        = 66;
defparam    c_sdram_tRAS        = 44;
defparam    c_sdram_tRCD        = 20;
defparam    c_sdram_tRP         = 20;
defparam    c_sdram_tWR         = 15;
defparam    c_sdram_tREF        = 64;
defparam    c_sdram_init_time   = 100;
defparam    c_sdram_init_afrnum = 8;
defparam    c_sdram_data_width  = 16;
defparam    c_sdram_row_addr    = 12;
```

```
defparam     c_sdram_col_addr     = 9 ;
defparam     c_sdram_bank_addr  = 2 ;
input                                        wb_clk_i;
input                                        wb_rst_i;
input     [c_sdram_data_width-1 : 0]    wb_dat_i;
input     [c_sdram_addr_width-1 : 0]    wb_adr_i;
output    [c_sdram_data_width-1 : 0]    wb_dat_o;
input                                        wb_we_i;
input                                        wb_cyc_i;
input                                        wb_stb_i;
input     [2:0]                              wb_cti_i;
output                                       wb_ack_o;
output    [c_sdram_row_addr-1 : 0]      sc_sa;
output    [c_sdram_bank_addr-1 :0]      sc_ba;
output    [c_sdram_CS -1 : 0]               sc_cs_n;
output    sc_cke;
output    sc_ck;
output                                       sc_ras_n;
input                                        sc_cas_n;
input                                        sc_we_n;
inout     [c_sdram_data_width-1 : 0]    sc_dq;
output    [c_sdram_data_width/8-1 : 0] sc_dqm;
sdram_ctrl system-define-name(
.wb_clk_i          (wb_clk_i    ),
.wb_rst_i          (wb_rst_i    ),
.wb_adr_i          (wb_adr_i    ),
.wb_dat_i          (wb_dat_i    ),
.wb_dat_o          (wb_dat_o    ),
.wb_we_i           (wb_we_i     ),
.wb_cyc_i          (wb_cyc_i    ),
.wb_ack_o          (wb_ack_o    ),
.wb_stb_i          (wb_stb_i    ),
.wb_cti_i          (wb_cti_i    ),
.sc_sa             (sc_sa       ),
.sc_ba             (sc_ba       ),
.sc_cs_n           (sc_cs_n     ),
.sc_cke            (sc_cke      ),
.sc_ck             (sc_ck       ),
.sc_ras_n          (sc_ras_n    ),
.sc_cas_n          (sc_cas_n    ),
.sc_we_n           (sc_we_n     ),
.sc_dq             (sc_dq       ),
.sc_dqm            (sc_dqm      )
);
```

```
defparam    system-defined-name.sys_clk             = sys_clk           ;
defparam    system-defined-name.c_sdram_CS          = c_sdram_CS        ;
defparam    system-defined-name.mem_opway           = mem_opway         ;
defparam    system-defined-name.mem_dist            = mem_dist          ;
defparam    system-defined-name.c_sdram_BL          = c_sdram_BL        ;
defparam    system-defined-name.c_sdram_BT          = c_sdram_BT        ;
defparam    system-defined-name.c_sdram_tCL         = c_sdram_tCL       ;
defparam    system-defined-name.c_sdram_wbmode      = c_sdram_wbmode    ;
defparam    system-defined-name.c_sdram_tRFC        = c_sdram_tRFC      ;
defparam    system-defined-name.c_sdram_tRAS        = c_sdram_tRAS      ;
defparam    system-defined-name.c_sdram_tRCD        = c_sdram_tRCD      ;
defparam    system-defined-name.c_sdram_tRP         = c_sdram_tRP       ;
defparam    system-defined-name.c_sdram_tWR         = c_sdram_tWR       ;
defparam    system-defined-name.c_sdram_tREF        = c_sdram_tREF      ;
defparam    system-defined-name.c_sdram_init_time   = c_sdram_init_time ;
defparam    system-defined-name.c_sdram_init_afrnum= c_sdram_init_afrnum;
defparam    system-defined-name.c_sdram_data_width = c_sdram_data_width;
defparam    system-defined-name.c_sdram_row_addr    = c_sdram_row_addr ;
defparam    system-defined-name.c_sdram_col_addr    = c_sdram_col_addr  ;
defparam    system-defined-name.c_sdram_bank_addr   = c_sdram_bank_addr;

endmodule
```

About Agate Logic

Agate Logic is the global pioneer and leader of the innovative Adaptable Programmable Gate Array (APGA) technologies. The company offers a full spectrum of programmable logic devices, software design tools, intellectual property (IP) and design services. Focusing on multiple applications such as telecommunication equipments, industrial control systems and consumer products, we use the Chinese leading foundry partner, SMIC, to manufacture our chips to offer solutions tailored for the market in China.

Technical Support Assistance

Tel: +86 10 82150100

E-mail: support@agatelogic.com

Website: www.agatelogic.com.cn