Scientific Research Publishing

Journal of Computer and Communications, 2020, 8, 14-22
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Design and Verification of SDRAM Controller Based on FPGA

Song Won Kil, Dong Ho Kim, Hyo Il Ri

Department of Computer Science, University of Science, Pyongyang, D. P. R. Korea
Email: kilsw122@163.com

How to cite this paper: Kil, S.W., Kim, D.H. and Ri, H.I. (2020) Design and Verification of SDRAM Controller Based on FPGA. *Journal of Computer and Communications*, **8**, 14-22.
https://doi.org/10.4236/jcc.2020.87002

Received: May 12, 2020
Accepted: July 3, 2020
Published: July 6, 2020

Abstract

SDRAM (Synchronous DRAM) has become the memory standard in many digital system designs, because of low price and high read/write speed. In this paper, Based on the analysis of the working principle and characteristics of SDRAM, an SDRAM controller design method is proposed based on field programmable logic gate array FPGA. In order to reduce resource consumption and increase the read and write speed of SDRAM, the performance control of SDRAM is further optimized. We designed SDRAM controller by using Verilog HDL and Altera Quartus II 14.1 software, and simulated about this design with Model Sim-Altera 10.3c software. Then we verified this design by using Cyclone V 5CSEMA5F31C6 FPGA in DE1-SoC development board. The verification results show that the SDRAM is initialized successfully, the input and output data are completely consistent, and it has stable refresh and read and write functions. The SDRAM controller design meets the requirements.

Keywords

SDRAM Controller, Verilog HDL, DE1-SoC

1. Introduction

SDRAM is a memory chip to use widely for embedded real-time systems and high-speed data transmission [1] [2].

In the development of computer systems, in order to give full play to the processor's ability to process data at high speed and improve the efficiency of the entire computer system, it is necessary to use cache technology to store input and output data, intermediate calculation results and temporary data exchanged with external memory Final Results. Most CPUs integrate a cache internally as a

matching buffer between the processor and main memory. Cache generally uses SRAM, but although SRAM is high speed, but its capacity is small, so to complete a large amount of data cache generally uses DRAM as the main memory. The core of modern computer technology is an integrated circuit with transistors as the basic unit. For a long time, in the process manufacturing of computers, processors and main memories were manufactured using different production technology lines, so that they can meet the design to the maximum Demand [3].

The processor process line is manufactured on the basis of logic technology, using high-speed transistors and multilayer metals to achieve high system performance and operating frequency; and the main memory production line is manufactured on the basis of DRAM technology, using units as much as possible Small area capacitors, low leakage current transistors and multilayer polysilicon interconnects to achieve large capacity, low cost and low refresh intervals. With the gradual progress of integrated circuit technology, the average annual growth rate of processor performance from 1986 to 2004 is about 52%, and the average annual growth rate after 2004 is about 20% [4], but the average of main memory. The growth rate is only about 7% per year [5] [6], the performance difference between the processor and the main memory is increasing, and the performance of the main memory has become an important factor restricting the performance of the processor [7] [8], which is the famous "Memory Wall" problem [9].

In designing the SDRAM controller, Seiji Miura and Satoru Akiyamma focused on solving the latency problem in order to improve data transmission efficiency. Finally, two solutions, address queue and virtual cache, were proposed to reduce latency [10]. Bonatto, Soates, *et al.* in order to generate a smaller delay in the encoding of the management macro definition module based on the H.264/AVC video decoder, use a four-layer module structure to implement a multi-channel SDRAM controller [11].

2. Design of SDRAM Controller

This SDRAM Controller is designed to interface to standard microprocessors. The controller is independent of processor type. This design supports two 16 MB memory regions configured as 4 M × 32 bits. Changing byte enable inputs and address inputs will change the width and size of this design. For example, if a 64 bit wide data bus is desired, increase the byte enable signals from 4 to 8. If a larger memory space is required, add address inputs and reconfigure the row and column address appropriately or add more chip selects. An overview of the state machine of the SDRAM is shown in **Figure 1**.

The output signals of SDRAM controller depend on the kind of SDRAM command to perform. **Table 1** lists all SDRAM commands and output signals ($\overline{CS}$: Chip Select, $\overline{RAS}$: Row Address Strobe, $\overline{CAS}$: Column Address Strobe, $\overline{WE}$: Write Enable, DQM: Data Qualifier Mask, ADD: Address).

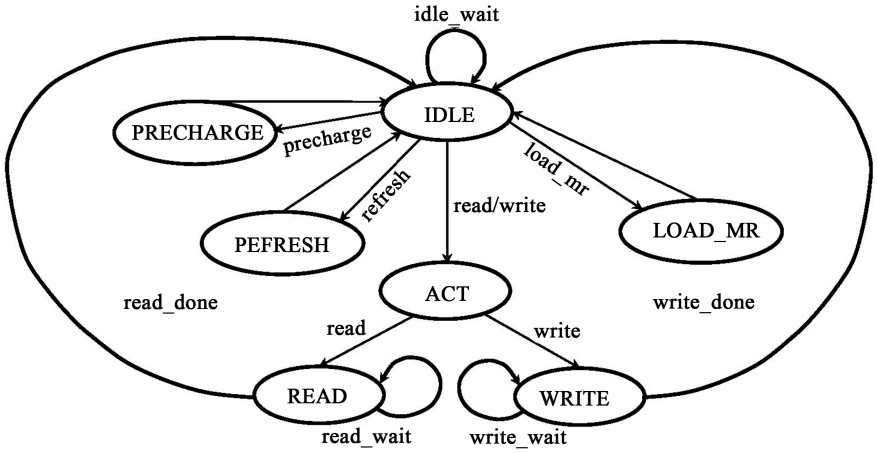

Figure 1. State machine diagram.

Table 1. Table type styles.

Command	$\overline{CS}$	$\overline{RAS}$	$\overline{CAS}$	$\overline{WE}$	DQM	ADD
Command Inhibit	H	X	X	X	X	X
No Operation	L	H	H	H	X	X
Activate	L	L	H	H	X	Bank/Row
Read	L	H	L	H	X	Bank/Col
Write	L	H	L	L	X	Bank/Col
Burst Terminate	L	H	H	L	X	X
Precharge	L	L	H	L	X	code
Refresh	L	L	L	H	X	X
Load Mode Register	L	L	L	L	X	Opcode
Write Enable/Output Enable	-	-	-	-	L	-
Write Inhibit/Output High-Z	-	-	-	-	H	-

2.1. Design of Top Level Module

Table 2 shows the input/output signals of the SDRAM Controller. Signals ending with "_L" indicate an active low signal. This convention is used throughout the design. All input signals except SDRAM_EN must be synchronous to the clock. SDRAM_EN is synchronized internally. The name of top level module is SD_TOP.

2.2. Design of Sub-Modules

The SDRAM Controller is comprised of a top-level module called SD_TOP as shown in Figure 2. This top-level module instantiates the four sub-modules: SD_CONFIG, SD_REFRESH, SD_STATE, SD_SIGNAL.

2.2.1. SD_CONFIG Module

The SD_CONFIG module performs the configuration and initialization of the SDRAMs. When the reset signal becomes inactive, and if the SDRAM_EN signal

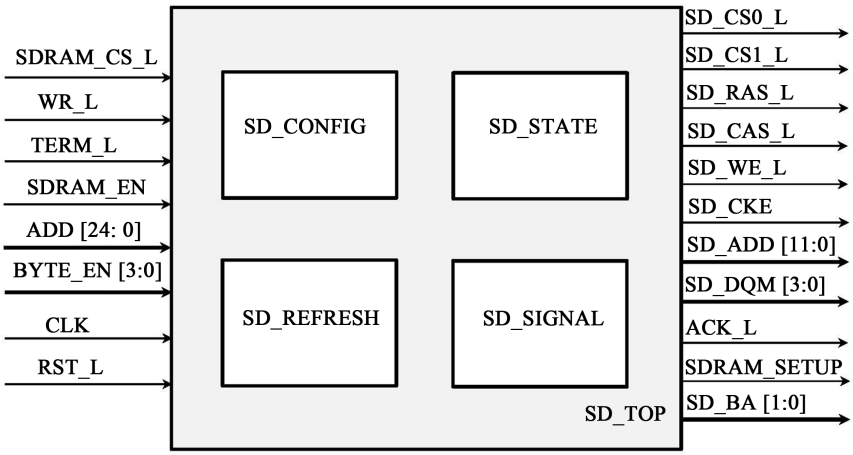

Figure 2. SDRAM block diagram.

Table 2. SDRAM signals.

Signal	Type	Description
SDRAM_CS_L	Input	SDRAM Chip Select from processor.
WR_L	Input	Write pulse from processor.
SDRAM_EN	Input	SDRAM Enable signal—may be tied high.
TERM_L	Input	Terminates burst cycles.
CLK	Input	Input clock signal.
RST_L	Input	Reset signal.
BYTE_EN [3:0]	Input	Byte enable signals.
ADD [24:0]	Input	Processor Address bus. Used to address SDRAM.
SD_CKE	Output	SDRAM Clock Enable signal.
SD_BA [1:0]	Output	SDRAM Bank Address signals.
SD_CS0_L	Output	SDRAM Chip Select signal for lower 16 MB region.
SD_CS1_L	Output	SDRAM Chip Select signal for upper 16 MB region.
SD_RAS_L	Output	SDRAM Row Address Strobe.
SD_CAS_L	Output	SDRAM Column Address Strobe.
SD_WE_L	Output	SDRAM Write Enable Strobe.
SD_ADD [11:0]	Output	SDRAM Address Signals.
SD_DQM [3:0]	Output	SDRAM Data Qualifier Mask.
ACK_L	Output	Acknowledge—Indicates when data cycles are active.
SDRAM_SETUP	Output	SDRAM Setup indicates that the SDRAM has been initialized.

is active, this module sends requests to the state machine module to initialize the SDRAM. The SDRAM_EN signal could be removed from the design or can be tied high if this feature is not needed. If tied high, the controller will initialize the SDRAM sub-system when the reset signal becomes false.

The parameters for the mode register are stored in this module. These can be modified to suit the user's specific needs. Some of the bits are reserved and must be kept as "0". The parameters CAS LATENCY, BURST MODE, BURST LENGTH and BURST TYPE can be changed in this module.

Table 3 describes the input and output signals of this module.

2.2.2. SD_REFRESH Module

The SD_REFRESH module provides a refresh request signal to the state machine module. The refresh module has a 12 bit counter that is clocked by the system clock. The output of the counter is set so that a request occurs every 15.6 μsec. The parameter value "count" can be changed depending on the clock frequency. The counter doesn't start counting until after the SDRAM has been initialized.

Table 4 describes the input and output signals of this module.

2.2.3. SD_STATE Module

The SD_STATE module takes requests from:

- The processor to perform data cycles.
- The SD_CONFIG module to perform command cycles.
- The SD_REFRESH module to perform refresh cycles.

Table 3. SD_CONFIG signals.

Signal	Type	Description
SDRAM_EN	Input	SDRAM Enable signal.
CLK	Input	Clock signal.
RST_L	Input	Reset signal.
SDRAM_CYCLE [3:0]	Input	State machine bits: indicates the type of cycle: 00 = idle, 01 = command, 10 = data, 11 = refresh.
STATE_CNTR [3:0]	Input	State machine bits—indicates state of cycle.
SDRAM_MODE_REG [11:0]	Output	Mode Register Value.
SDRAM_CMND [1:0]	Output	SDRAM command desired: 00 = nop, 01 = precharge, 10 = autorefresh, 11 = load mode register.
CMND_CYCLE_REQ	Output	Command Cycle Request to state machine.
SDRAM_SETUP	Output	Indicates SDRAM setup is complete.

Table 4. SD_REFRESH signals.

Signal	Type	Description
CLK	Input	Clock signal.
RST_L	Input	Reset signal.
SDRAM_SETUP	Input	Indicates SDRAM setup is complete.
SDRAM_CYCLE [3:0]	Input	State machine bits—indicates the type of cycle: 00 = idle, 01 = command, 10 = data, 11 = refresh.
RFRSH_REQ	Output	Refresh cycle request to state machine.

It outputs a state type vector as well as a state bit vector. The type vector indicates what type of cycle is being performed. The bit vector indicates the state cycle. Table 5 describes the input/output signals for this module.

2.2.4. SD_SIGNAL Signals

The SD_SIGNAL module outputs the appropriate SDRAM signals depending on what type of cycle is occurring and where the state machine is at in the cycle. Table 6 describes the input and output signals for this module.

3. Synthesis of SDRAM Controller

This design was synthesized by using Altera Quartus II 14.1 and Altera's Cyclone V 5CSEMA5F31C6 FPGA. The results of systhesis about each module of SDRAM controller shown below in Figure 3.

Table 7 lists all resource consumption used to synthesis of SDRAM controller.

Table 5. SD_STATE signals.

Signal	Type	Description
SDRAM_CS_L	Input	Chip select signal from processor.
CMND_CYCL_REQ	Input	Command cycle request from SD_CONFIG module.
RFRSH_REQ	Input	Refresh request from SD_REFRESH module.
CLK	Input	Clock signal.
RST_L	Input	Reset signal.
SDRAM_CYCLE	Output	State machine bits: indicates the type of cycle: 00 = idle, 01 = command, 10 = data, 11 = refresh.
STATE_CNTR [3:0]	Output	State machine bits: indicates state of cycle.

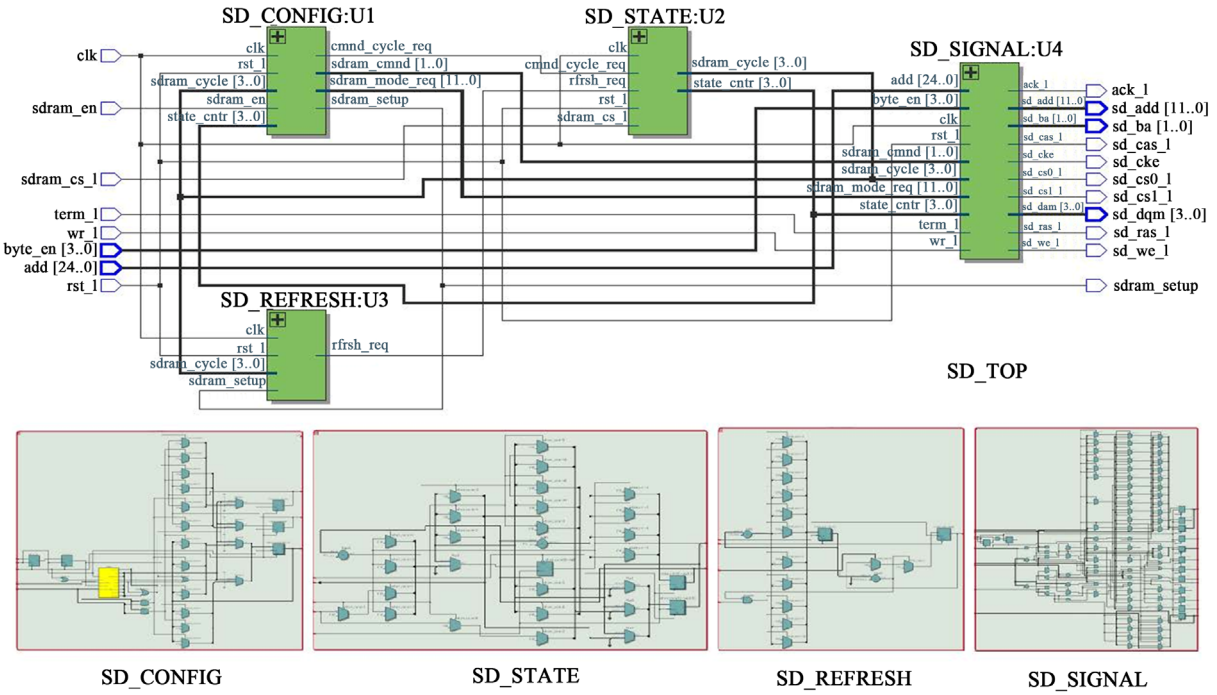

Figure 3. The synthesis result of SDRAM controller.

Table 6. SD_SGINAL signals.

Signal	Type	Description
ADD [24:0]	Input	Address bus from processor.
WR_L	Input	Write strobe from processor.
BYTE_EN [3:0]	Input	Byte enable signals from processor.
TERM_L	Input	Terminate signal from processor.
SDRAM_CYCLE [3:0]	Input	State machine bits—indicates the type of cycle: 00 = idle, 01 = command, 10 = data, 11 = refresh.
STATE_CNTR [3:0]	Input	State machine bits—indicates state of cycle.
SDRAM_MODE_REG [11:0]	Input	Mode Register Value.
SDRAM_CMND [1:0]	Input	SDRAM command desired: 00 = nop, 01 = precharge, 10 = autorefresh, 11 = load mode register.
CLK	Input	Clock signal.
RST_L	Input	Reset signal.
SD_CKE	Output	SDRAM Clock Enable signal.
SD_BA [1:0]	Output	SDRAM Bank Address signals.
SD_CS0_L	Output	SDRAM Chip Select signal for lower 16 MB region.
SD_CS1_L	Output	SDRAM Chip Select signal for upper 16 MB region.
SD_RAS_L	Output	SDRAM Row Address Strobe.
SD_CAS_L	Output	SDRAM Column Address Strobe.
SD_WE_L	Output	SDRAM Write Enable Strobe.
SD_ADD [11:0]	Output	SDRAM Address Signals.
SD_DQM [3:0]	Output	SDRAM Data Qualifier Mask.
ACK_L	Output	Acknowledge—indicates when data cycles are active.

Table 7. The resource consumption of SDRAM controller.

FPGA device	Cyclone V 5CSEMA5F31C6
Logical utilization	46/32,070 (<1%)
Total registers	75
Total pins	61/457 (13%)
Total block memory bits	0/4,065,280 (0%)
Total DSP Blocks	0/87 (0%)

4. Simulation and Verification of SDRAM Controller

This design was simulated by using Model Sim-Altera 10.3c software. For simulation, it used 100 MHz clock frequency to be provided by PLL (Phase Locked Loop). The results of simulation about read and write cycle of SDRAM controller shown below in **Figure 4** and **Figure 5**.

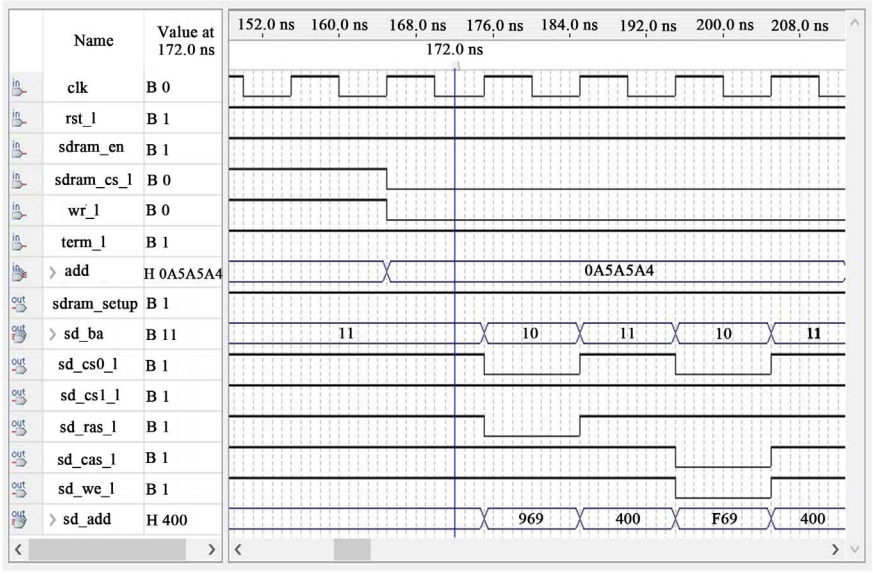

Figure 4. The read cycle of SDRAM controller.

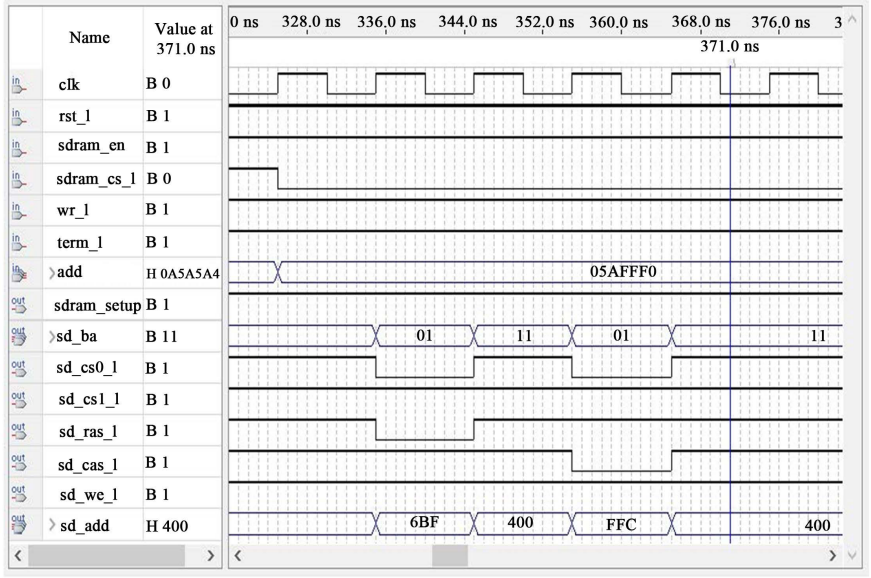

Figure 5. The write cycle of SDRAM controller.

5. Conclusion

We verified accuracy of design through the simulation and downloaded code at Cyclone V 5CSEMA5F31C6 FPGA in DE1-SoC development board. Then we linked with display unit by VGA bus and showed image stored to SDRAM at scene. The result shows that the SDRAM controller designed in this paper satisfy requirement of design.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Anup, P. and Ramana Reddy, R. (2012) A Low Power DDR SDRAM Controller Design. *International Journal of Computer Science and Information Technologies*, **3**, 4270-4274.

[2] Sharma, D. and Bhargava, S. (2011) Design and VLSI Implementation of DDR SDRAM Controller for High Speed Applications. *International Journal of Computer Science and Information Technologies*, **2**, 1625-1632.

[3] Upadhyay, P., Kar, R., *et al*. (2019) A Design of Highly Stable and Low-Power SRAM Cell. Advances in Computer Communication and Computational Sciences, **759**, 281-289. https://doi.org/10.1007/978-981-13-0341-8_26

[4] Wulf, W.A. and McKee, S.A. (1995) Hitting the Memory Wall: Implications of the Obvious. *Computer Architecture News*, **23**, 20-24. https://doi.org/10.1145/216585.216588

[5] Wilkes, M.V. (1995) The Memory Wall and the CMOS End-Point. *Computer Architecture News*, **23**, 4-6. https://doi.org/10.1145/218864.218865

[6] Burger, D.C., Goodman, J.R. and Kagi, A. (1995) The Declining Effectiveness of Dynamic Caching for General-Purpose Microprocessors. Technical Report 1261, Computer Sciences Department, University of Wisconsin, Madison, WI.

[7] Kagi, A., Goodman, J.R. and Burger, D. (1996) Memory Bandwidth Limitations of Future Microprocessors. 23*rd Annual International Symposium on Computer Architecture* (*ISCA* 1996), Philadelphia, PA, 24 May 1996, 78-89.

[8] Huh, J., Burger, D. and Keckler, S.W. (2001) Exploring the Design Space of Future CMPs. *International Conference on Parallel Architectures and Compilation Techniques* (*PACT* 2001), Barcelona, Spain, 8-12 September 2001, 199-210.

[9] Rogers, B., Krishna, A., Bell, G., *et al.* (2009) Scaling the Bandwidth Wall: Challenges in and Avenues for CMP Scaling. 36*th Annual International Symposium on Computer Architecture* (*ISCA* 2009), Austin, TX, 20-24 June 2009, 371-382. https://doi.org/10.1145/1555754.1555801

[10] Miura, S. and Akiyama, S. (2005) A Memory Controller That Reduces Latency of Cached SDRAM. 2005 *IEEE International Symposium on Circuits and Systems*, Kobe, Japan, 23-26 May 2005, 5250-5253. https://doi.org/10.1109/ISCAS.2005.1465819

[11] Bonatto, A.C., Soares, A.B. and Susin, A.A. (2011) Multichannel SDRAM Controller Design for H.264/AVC Video Decoder. 2011 *VII Southern Conference on Programmable Logic* (*SPL*), Cordoba, Argentina, 137-142. https://doi.org/10.1109/SPL.2011.5782638