

# Design and Characterization of SDRAM Controller IP Core

<sup>1</sup>Arathy S, <sup>2</sup>Nandakumar R, <sup>3</sup>Ashwin P V

<sup>1,3</sup>Dept. of ECE, Saintgits College of Engineering, Kottayam, Kerala, India

<sup>2</sup>VLSI Design Group, NIELIT Calicut, Kerala, India

## Abstract

In modern digital systems large capacity and data transfer rate is required. Synchronous DRAM (SDRAM) become the memory of choice due to its speed, burst access and pipeline features. A Controller is required to provide proper commands for SDRAM initialization, read/write accesses and memory refresh. This paper describes the design of a synthesizable SDRAM Controller IP core which is vendor neutral. The design is described using Verilog HDL, simulated using ModelSim and prototyped in Altera® platform FPGA. Resource utilization and power analysis was done using Altera® Quartus II. Hardware test results are obtained from Signal Tap Logic Analyzer.

## Keywords

Synchronous DRAM, IP Core, Power Analysis, Resource Utilization

## I. Introduction

Synchronous DRAMs (SDRAMs) become the memory of choice in many digital systems because it provides a significant improvement in bandwidth performance over traditional asynchronous DRAMs such as "FPM" (Fast Page Mode) and "EDO" (Extended Data Out). In Synchronous DRAMs input address, data, and control signals are typically latched on the positive edge of the clock signal. Synchronous DRAMs offer several features such as multiple internal banks, burst mode access, and pipelining of operation executions, that helps to improve bandwidth performance.

There are two popular types of SDRAM in market. The most common single data rate (SDR) SDRAM transfers data typically on the rising edge of the clock. The other is the double data rate (DDR) SDRAM [5] 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 paper describes a design that is targeted for SDR SDRAM. However, due to the similarity of SDR and DDR SDRAM, this design can also be adapted for a DDR SDRAM controller. For benchmarking purpose, the Micron® SDR SDRAM MT48LC8M16A2 is chosen as a target for this design. Also, this design has been verified by using memory simulation model.

The section II of this paper is a tutorial review of Synchronous DRAMs. The section III describes the proposed SDRAM Controller design. The section IV describes the simulation results. The section V describes the implementation results of the proposed controller.

## II. Synchronous DRAM Review

SDRAM, or Synchronous Dynamic Random Access Memory is a form of semiconductor memory that can run at faster speeds than conventional DRAM and therefore the use of SDRAM is becoming more widespread. Since SDRAM has a synchronous interface, it has an internal finite state machine that pipelines incoming instructions. Thus the speed of operation is much higher.

As a result of the multiple banks present and pipelining feature, SDRAM is capable of keeping two sets of memory addresses

open simultaneously. By transferring data alternately from one set of addresses, and then the other, SDRAM cuts down on the delays associated with asynchronous RAM, which must close one address bank before opening the next. The term pipelining is used to describe the process whereby the SDRAM can accept a new instruction before it has finished processing the previous one. In other words, it can effectively process two or more instructions at once. cxjvkxhc

Another important feature of SDRAM is its facility for burst read and burst write. Accesses continue for a programmed number of locations in a programmed sequence. When ACTIVE command is issued, address bits are used to select the bank and row. After ACTIVE command READ or WRITES command is issued. During this time address bits are used to select the starting column location.

The Micron® 128Mb [4] SDRAM referenced in this paper is a high-speed CMOS, dynamic random access memory containing 134,217,728 bits. It is internally configured as a quad-bank DRAM with a synchronous interface. Each of the 33,554,432-bit banks is organized as 4,096 rows by 512 columns by 16 bits. The 128Mb SDRAM is designed to operate in 3.3V memory systems. All inputs and outputs are LVTTTL-compatible.

## III. Proposed Controller Design

### A. Introduction

The proposed SDRAM Controller is designed to work with a standard memory from Micron Technology® with series MT48LC8M16A2™. It has a user interface end on one side and 128Mb SDRAM on the other end. The design is coded in Verilog HDL. The controller offers facility for programmable burst lengths of 1, 2, 4, and 8, programmable CAS latency of 2 and 3.

Initialization should be done before applying any normal operation. Read and Write should be performed only after the initialization. The proposed controller design automatically performs all the initialization procedures [1-2,4].

### B. Block Diagram

The block diagram of proposed SDRAM Controller is shown in fig. 1. The memory inputs are a set of control signals whose value indicates a particular operation. The main function of the controller is to convert user commands to commands that can be understandable by the memory [6]. The main blocks inside the controller are datapath to control the data movement, a finite state machine to convert user commands to SDRAM commands and an initialization control block to perform automatic initialization.

The inputs like command (cmd) and reset are collectively shown as control input. The memory address is given through the address input. The data is given through the data input.

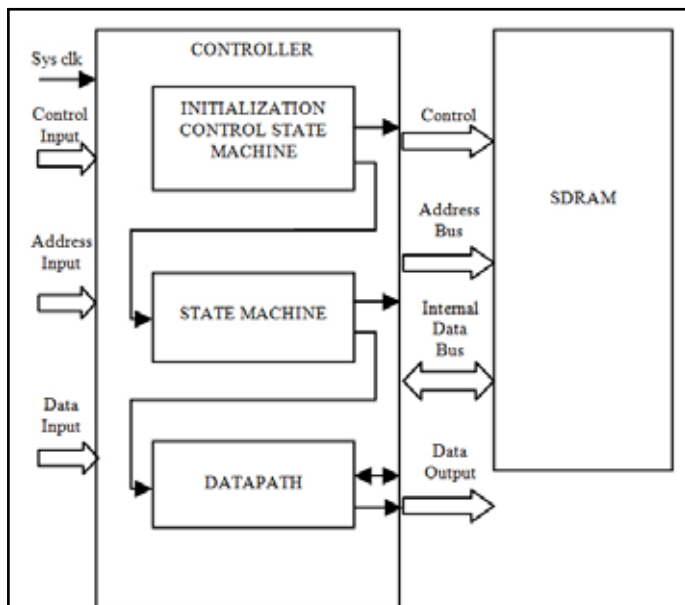


Fig. 1: Controller Block Diagram

### C. Pin Description

The input output diagram of controller is shown in fig. 2. The outputs from the controller are given as inputs to the memory.

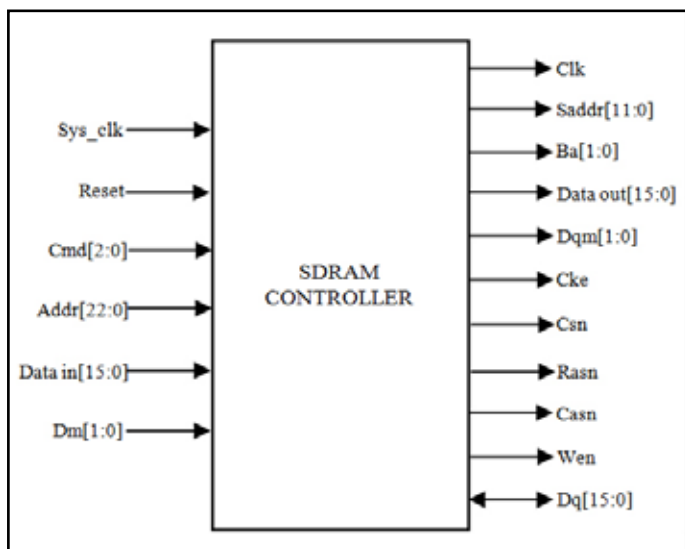


Fig. 2: Pin Out Diagram of Controller

All the pins in the pin diagram are described in Table 1. Type and the function of all the pins are also described.

Table 1: Pin Description

| PIN             | TYPE   | DESCRIPTION                                                                                                                           |
|-----------------|--------|---------------------------------------------------------------------------------------------------------------------------------------|
| Sys Clk         | Input  | Input clock to the controller. Used to generate the clock (clk) to the memory.                                                        |
| Reset           | Input  | System reset - Used to reset the total system.                                                                                        |
| Cmd [2:0];      | Input  | Command input which distinguishes between different operations like Precharge, refresh, load mode register, activate, read and write. |
| Data in [15:0]  | Input  | 16-bit data input to the controller which is used as data during write.                                                               |
| Data out [15:0] | Output | 16-bit data output from the controller during read operation.                                                                         |

|              |        |                                                                                                                                                                                                                                                                                                                                  |
|--------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Addr [22:0]  | Input  | 23-bit Address bus. From this 23-bit address row address, column address and bank address is decoded.                                                                                                                                                                                                                            |
| Dm [1:0]     | Input  | Dm is used to mask input data during write cycle and output data during read cycle.                                                                                                                                                                                                                                              |
| Clk          | Output | CLK is driven by the system clock. Typical values are 100MHz and 133MHz.                                                                                                                                                                                                                                                         |
| Saddr [11:0] |        | Address input A0–A11 are sampled during the ACTIVE command (row address A0–A11) and READ/WRITE command (column-address A0–A8).                                                                                                                                                                                                   |
| Ba [1:0]     | Output | Bank address is used to select the bank                                                                                                                                                                                                                                                                                          |
| Cke          | Output | CKE activates (HIGH) and deactivates (LOW) the CLK signal.                                                                                                                                                                                                                                                                       |
| Csn          | Output | CS enables (LOW) and disables (HIGH) the command decoder. All commands are masked when CS is registered HIGH.                                                                                                                                                                                                                    |
| Rasn         | Output | Command inputs WE, CAS, and RAS (along with CS) define the command being entered.                                                                                                                                                                                                                                                |
| Casn         | Output |                                                                                                                                                                                                                                                                                                                                  |
| Wen          | Output |                                                                                                                                                                                                                                                                                                                                  |
| Dqm [1:0]    | Output | DQM is an input mask signal for write accesses and an output enable signal for read accesses. Input data to the memory is masked when DQM is sampled HIGH during a WRITE cycle. The output buffers are placed in a High-Z state (2-clock latency) when DQM is sampled HIGH during a READ cycle. DQM is generated from Dm signal. |
| Dq [15:0]    | Inout  | Bidirectional data bus which is connected to the bidirectional data bus of memory. During write operation Dq is directed from controller to the memory. During read operation Dq is directed from memory to the controller                                                                                                       |

### D. Finite State Machine

According to architecture and working principle of SDRAM [4], the controller design uses two Finite State Machines to implement timing-logic control [1-2]. The SDRAM must be powered up and initialized in a predefined manner before any normal operation. One is to automatically initialize the SDRAM. Another state machine is the main FSM or the core of the controller. On one hand it can switch into any state according to its condition, on another hand it can adjust the scheduling of those modules and make a decision for the signals that come from the others modules. Former one is shown below.

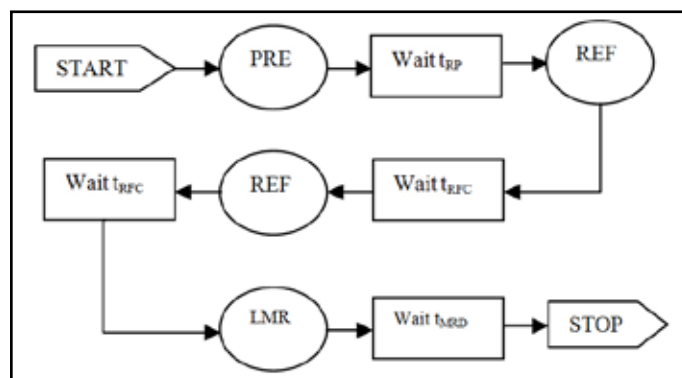


Fig. 3: Initialization State Machine

Initialization process consists of sequence of operations. First Precharge all banks should be done that means deactivate all

banks and put them in idle state. Then wait for tRP (150 ns) period and execute Auto Refresh. Then wait for tRFC (495 ns) period and again apply Auto Refresh. Again wait for tRFC period and at last Load Mode Register command should be issued. With this command a special attributes are set (i.e. burst packet length access type CAS latency and other). Then wait for tMRD (15 ns) period. After the initialization operations, the SDRAM goes into the idle state, which is the one state of the main FSM which is shown below.

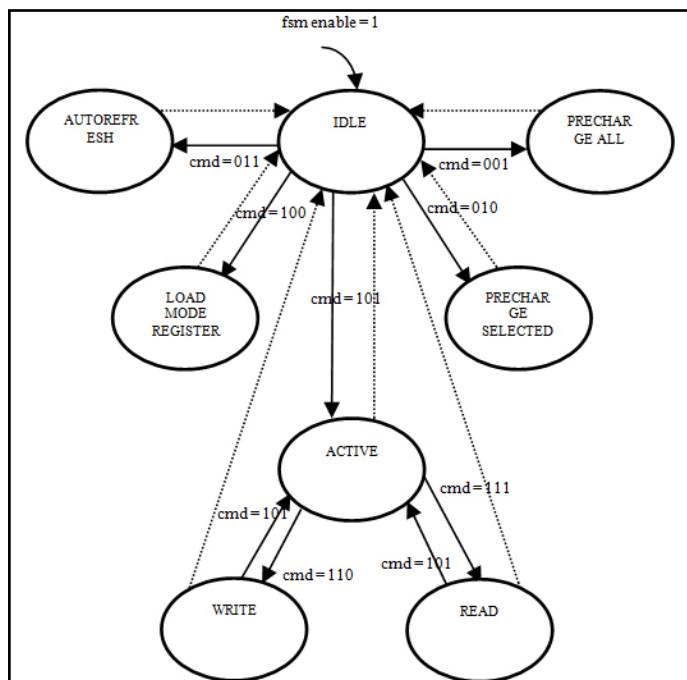


Fig. 4: Main FSM

### E. Datapath Circuit

The data flow design between the SDRAM and the system interface is shown in figure. The datapath circuit consists of four number of 16-bit D flip flops and a tristate buffer. Flip flop is used to shift the data. Tristate buffer is used to determine the direction of bidirectional data bus. Tri state buffer has a control input oe which determines whether it is a read or write operation.

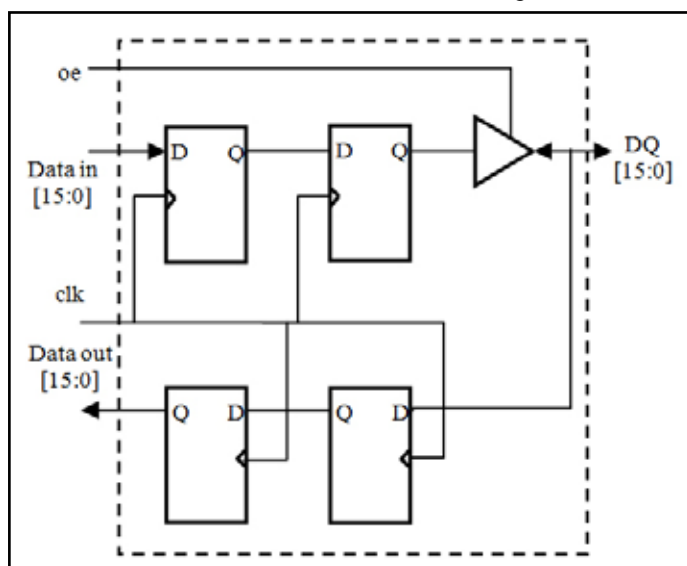


Fig. 5: Datapath Circuit of the Controller

### IV. Simulation Results

The Controller along with memory can be simulated using

Modelsim software. The functional simulation result of the wrapper module is shown in fig. 6. The designed IP core can be further prototyped using Altera DE2 FPGA board [3].



Fig. 6: Simulation Result

Note on simulation result :

Reset:(clk=1, reset=1, cmd=xxx, dm=xx, data\_in=xxxxxxxxxxxxxxxxxx, addr=xxxxxxxxxxxxxxxxxxxxxxxxxx, data\_out=0000000000000000)

Initialization:(clk=1, reset=0, cmd=xxx, dm=xx, data\_in=xxxxxxxxxxxxxxxxxx, addr=0000000000000000010010, data\_out=0000000000000000)

Activate:(clk=1, reset=0, cmd=101, dm=00, data\_in=1100110011001100, addr=0000000000000000000000, data\_out=0000000000000000)

Write:(clk=1, reset=0, cmd=110, dm=00, data\_in=1100110011001100, addr=0000000000000000000000, data\_out=0000000000000000)

Write:(clk=1, reset=0, cmd=110, dm=00, data\_in=0110011001100110, addr=00000000000000000000001, data\_out=0000000000000000)

Write:(clk=1, reset=0, cmd=110, dm=00, data\_in=0011001100110011, addr=00000000000000000000010, data\_out=0000000000000000)

Write:(clk=1, reset=0, cmd=110, dm=00, data\_in=1001100110011001, addr=00000000000000000000011, data\_out=0000000000000000)

Idle:(clk=1, reset=0, cmd=000, dm=00, data\_in=xxxxxxxxxxxxxxxxxx, addr=xxxxxxxxxxxxxxxxxxxxxxxxxx, data\_out=0000000000000000)

Activate:(clk=1, reset=0, cmd=101, dm=00, data\_in=xxxxxxxxxxxxxxxxxx, addr=0000000000000000000000, data\_out=0000000000000000)

Read:(clk=1, reset=0, cmd=111, dm=00, data\_in=xxxxxxxxxxxxxxxxxx, addr=0000000000000000000000, data\_out=1100110011001100)

Read:(clk=1, reset=0, cmd=111, dm=00, data\_in=xxxxxxxxxxxxxxxxxx, addr=00000000000000000000001, data\_out=0110011001100110)

Read:(clk=1, reset=0, cmd=111, dm=00, data\_in=xxxxxxxxxxxxxxxxxx, addr=00000000000000000000010, data\_out=0011001100110011)

Read:(clk=1, reset=0, cmd=111, dm=00, data\_in=xxxxxxxxxxxxxxxxxx, addr=00000000000000000000011, data\_out=1001100110011001)

### V. Implementation Results

The controller module is tested by using Altera® DE2 FPGA board. Sys\_clk is taken from the board itself. Reset is given as trigger input. Altera Quartus® II is used to synthesis the design. Signal Tap® II Logic Analyzer is used to take hardware test result from FPGA. Power analysis and Resource utilization result is listed in Table 2 and Table 3 respectively.



| Operation | Cs_n | Ras_n | Cas_n | We_n |
|-----------|------|-------|-------|------|
| Idle      | 1    | 0     | 0     | 0    |
| Active    | 0    | 0     | 1     | 1    |
| Read      | 0    | 1     | 0     | 1    |
| Write     | 0    | 1     | 0     | 0    |

Fig. 7: Hardware Test Result

The hardware test results tabulated above are subjected to vis-à-vis comparison for the function table, for verifying compliance. Power Analysis table in Table 2, gives details about the core dynamic thermal power dissipation, core static thermal power dissipation, I/O thermal power dissipation and total thermal power dissipation.

Table 2: Power Analysis Table

| PARAMETER                              | RESULT    |
|----------------------------------------|-----------|
| Total Thermal Power Dissipation        | 118.16 mW |
| Core Dynamic Thermal Power Dissipation | 0.95 mW   |
| Core Static Thermal Power Dissipation  | 79.95 mW  |
| I/O Thermal Power Dissipation          | 37.26 mW  |

Resource Utilization table summarizes usage statistics for resources including logic elements, registers, I/O pins, memory blocks, interconnect usage, and fan-out.

Table 3: Resource utilization table

| RESOURCE                                    | USAGE   |
|---------------------------------------------|---------|
| Estimated Total logic elements              | 1108    |
| Total combinational functions               | 679     |
| Logic element usage by number of LUT inputs |         |
| -- 4 input functions                        | 337     |
| -- 3 input functions                        | 152     |
| -- 2 input functions                        | 190     |
| Logic elements by mode                      |         |
| -- normal mode                              | 625     |
| -- arithmetic mode                          | 54      |
| Total registers                             | 838     |
| -- Dedicated logic registers                | 838     |
| -- I/O registers                            | 0       |
| I/O pins                                    | 55      |
| Total memory bits                           | 4096    |
| Maximum fan-out node                        | sys_clk |
| Maximum fan-out                             | 558     |
| Total fan-out                               | 5161    |
| Average fan-out                             | 3.21    |

## VI. Conclusion

This paper describes the design, simulation and characterization of synthesizable SDRAM Controller IP core. The design was

developed using Verilog HDL. It can be easily modified for different system design requirements. The proposed design has been tested by implementing the design on Altera DE2 board which uses Cyclone-II device.

## References

- [1] Tomasz Szymanski, Rafak Kiekbik, Andrzej Napieralski, "SDRAM controller for real time digital image processing systems", CAD Systems in Microelectronics, 2001. Proceedings of the 6th International Conference.
- [2] Qiu Daqiang, Hu Bing, Li Dandan, "Design of SDRAM Controller in High-Speed Data Acquisition Based on PCI Bus", The Eighth International Conference on Electronic Measurement and Instruments 2007.
- [3] "Altera DE2 Development Board User Manual", [Online] Available: [http://www.altera.com/up/pub/Webdocs/DE2\\_UserManual.pdf](http://www.altera.com/up/pub/Webdocs/DE2_UserManual.pdf)
- [4] MICRON Technology Inc., "Synchronous DRAM: MT48LC8M16A2 Data Sheet".
- [5] MICRON Technology Inc., "DOUBLE DATA RATE (DDR) SDRAM: MT46V32M16 Data Sheet".
- [6] Xilinx Inc., XAPP134, "Synthesizable High Performance SDRAM Controller", 2000.



Arathy S was born in Kerala, India in March 1989. She has B.Tech Degree in electronics and communication engineering from Cochin University of Science and Technology in 2010. She is currently pursuing masters degree program in VLSI and Embedded System.