International Journal of Emerging Engineering Research and Technology
Volume 2, Issue 6, September 2014, PP 56-63
ISSN 2349-4395 (Print) & ISSN 2349-4409 (Online)

Physical Design Implementation of 32 bit DDR SDRAM Controller

Anuraag Marepally

PG Scholar, Department of ECE,
Aurora’s technological and Research Institute,
JNTU Hyderabad, Telangana, India
marepally.anuraag@gmail.com

V.V.N.S. Sudha

Assistant Professor, Department of ECE,
Aurora’s Technological and Research Institute,
JNTU Hyderabad, Telangana, India
vedantam_sudha@yahoo.com

Abstract: *Present day SOC’s and other high-end applications that don’t contain a microprocessor require a dedicated memory controller. Synchronous DRAM is the major off-chip memory module being used presently. A memory controller is required to meet the high speed and high performance requirement of any design. This paper discusses the functional design of a 32-bit DDR SDRAM controller and implementation of the design through the synthesis and physical design flow which form a part of ASIC Design Flow. This block level, semi-custom implementation is done using Synopsys Design Compiler and IC Compiler at 90nm technology node.*

Keywords: *DDR SDRAM, Synopsys Design Compiler, IC Compiler, Physical Design, SOC, semi-custom.*

1. INTRODUCTION

Modern day SOC’s and high end systems all demand for memory devices that provide high speed and high performance. The need for controllers comes in when there are many modules that share the memory resource simultaneously and to maintain high throughput and speed. To attain such high speeds the memory controllers should be designed to work at clock rates which lie in megahertz increasing the complexity of the controller. The speed of the design would also depend on the speed of the off-chip memory module. Synchronous DRAM's provide such facility of high data transfer rate and the Double Data Rate (DDR) SDRAM is one such memory module in the family. In this paper the functional design of DDR SDRAM controller is demonstrated and implementation of the design through the synthesis and physical design phase are done.

The controller would act as an interface between the bus master and the memory module simplifying the command signals like refresh, read and write operation and initialization of the DDR SDRAM. This would minimize the effort of the bus master to communicate with the memory. Figure 1 demonstrates the block diagram of the DDR SDRAM memory controller that is located between the bus master and the memory.

SDRAM’s are classified based on their data transfer rates as Single Data Rate(SDR) SDRAM, in which the data transfer takes place on a single clock edge and Double Data Rate(DDR) SDRAM where the transfer takes place on both the clock edges (positive and negative). This property of DDR SDRAM’s provides for higher data transfer rate without much increase in clock frequency and bus width [1].

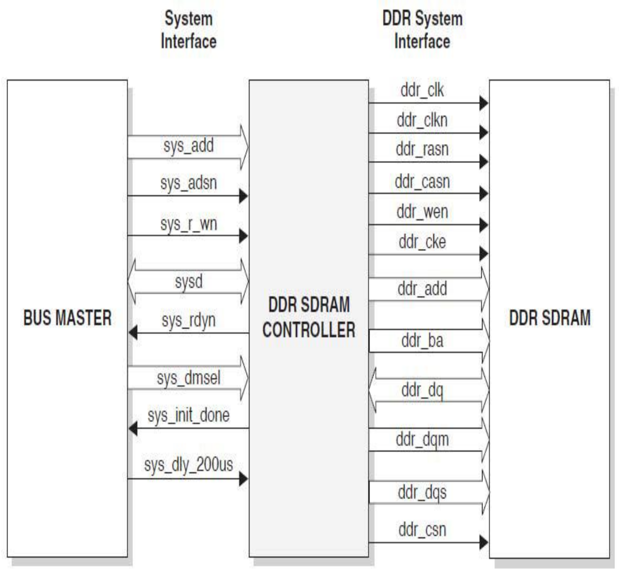


Fig 1. *DDR SDRAM controller system [1]*

2. IMPLEMENTATION METHODOLOGY

The DDR SDRAM Controller architecture is designed using Verilog HDL. The basic steps that an ASIC design must go through are specifications, architectural and behavioural description, simulation, logic synthesis of the RTL code, physical design flow, GDS-II [4]. The implementation of the controller design forms a part of the ASIC Design flow concentrating on Logic Synthesis and Physical Design phases.

3. ARCHITECTURAL DESCRIPTION

DDR SDRAM Controller module receives addresses and control signals from the BUS Master. The Controller generates command signals, based on these signals the data is either read or write to a particular memory location. The DDR SDRAM Controller architecture is shown in Figure 2. It consists of three modules: 1) Main control module 2) signal generation module 3) data path module. The main control module has two state machines and a refresh counter. The two state machines are for initialization of the SDRAM and for generating the commands to the SDRAM. The data paths modules can performs the read and write operations between the bus master and DDR [2]. The main control module consists of three sub modules:

1) Initialization FSM module (INIT_FSM). 2) Command FSM module. 3) Counter module.

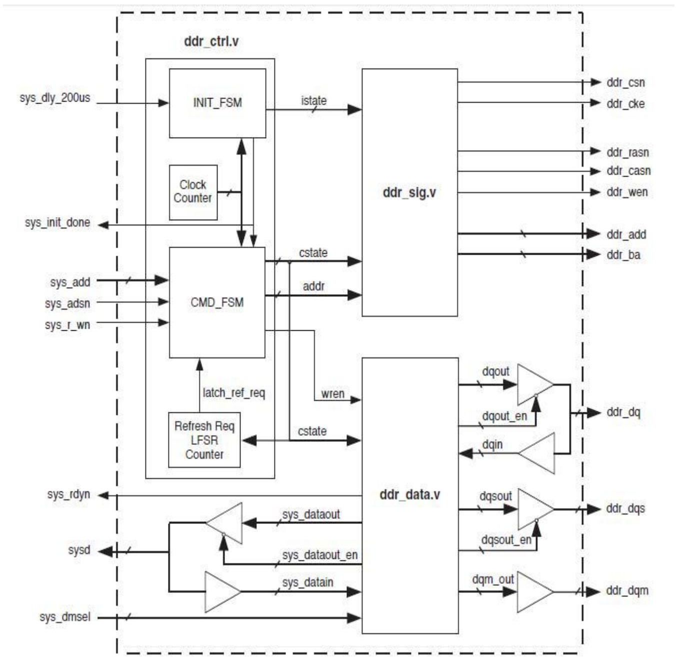


Fig 2. *Architecture of the controller [2]*

The finite state machines that are associated with the design are shown in Figure 3 and Figure 4.

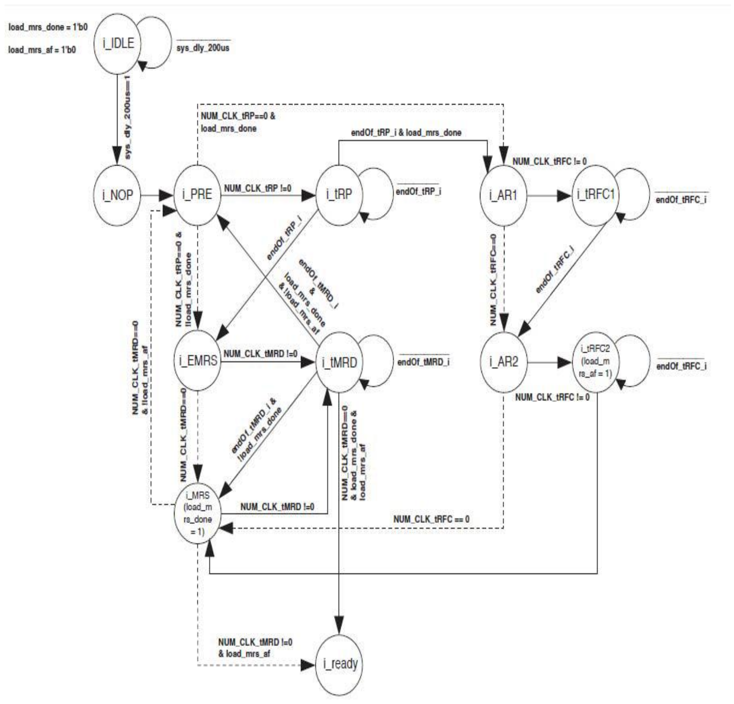


Fig 3. *The DDR SDRAM Initialization fsm (INIT_FSM) [2]*

4. IMPLEMENTATION OF THE DESIGN

The implementation of the design is carried out in two phases: *A)* Logic Synthesis, *B)* Physical Design flow also known as Place and Route.

A) Logic Synthesis:

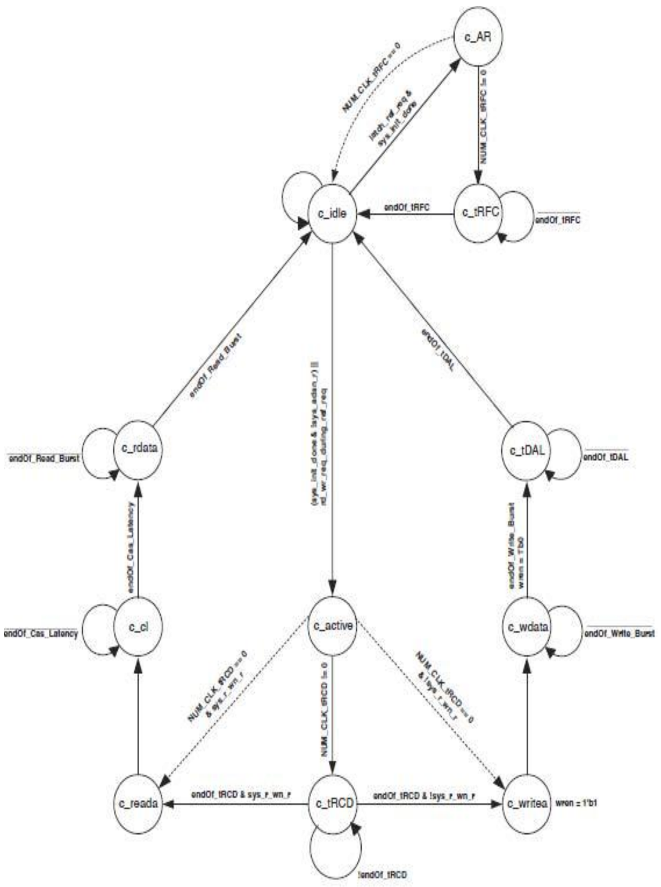


Fig4.*The DDR SDRAM Command FSM (CMD_FSM)[2]*

Logic Synthesis is a process in which the verified RTL is converted in to an optimized, technology mapped Gate-Level Netlist. The CAD tool used for Synthesis is Design Compiler from Synopsys. Synthesis is the phase in which the design would be translated and mapped into actual hardware in the form of logic gates interconnected through nets. The mapping of the RTL is done taking the technology libraries as reference. The technology libraries used here are at 90nm node. The libraries along with the verified RTL and a set of constraints would constitute the necessary files required by the Design Compiler to synthesize the design. The tool would generate the Gate-level Netlist, timing and area reports, delay files(.SDF) and constraints(.SDC) files [5]. The top-level schematic of the design after logic synthesis is shown in figure 5.

B) Physical Design Flow (Place and Route):

The Physical Design Flow also called the Place and Route is a process in which the Gate-Level Netlist generated in the Synthesis phase would be converted into the physical layout of the design. In this phase the standard cells present in the physical libraries would be placed and routing is done accordingly. The CAD tool used for Place and Route is IC Compiler from Synopsys. A typical Place and Route phase would undergo the following hierarchical sequence:

i) Floor planning.
ii) Placement.
iii) Clock Tree Synthesis.
iv) Routing.

After passing through above sequence of steps, one would have completely routed layout of the design along with the delay information. The tool accepts the Gate-Level Netlist, physical libraries, TLU+ files which contain the electrical characteristics of the interconnect wires and the required constraints file generated from synthesis.

Before proceeding to the Floor Plan phase, the Milky-Way library should be setup which serves as the design environment. The physical libraries, technology files, TLU+ files are also defined [6].

i) Floor Planning:

Floor panning is the first step in the hierarchy. In this phase the basic skeleton of our block would be established. Floor planning is the process of identifying structures and it should be placed close together. It allocating space for them in such a manner as to meet the sometimes conflicting goals of available space (cost of the chip), required performance, and the desire to have everything close to everything else.

Floor planning also decides the IO structure, aspect ratio of the design, utilization of the core area, power and ground rails would be laid and the necessary power straps are laid over running right through the core area. A bad floor plan will lead to wastage of die area and routing congestion. Figure 6 illustrates the Floor planning result of the controller.

ii) Placement:

Placement is the predominant phase in physical design which would decide the efficiency of the routing process that follows. Both placement and routing go hand-in-hand iteratively. The standard cells required would be placed initially and coarse routing is done. The In-Placement optimization would perform iterations of set-up fixing, incremental timing and congestion driven placement. To fix setup, hold and max transition time Post-placement Optimization (PPO) is done before and after CTS. PPO before CTS would take ideal clocks into consideration whereas PPO after CTS would consider the propagated clock preserving useful skew if presents [3]. Sufficient care would be taken such that cell overlap is avoided and also the empty spaces that result after placement would be filled using filler cells which protect the block during the fabrication process. The placement is carried out using the *place_opt* command. Figure 7 shows the controller design post placement.

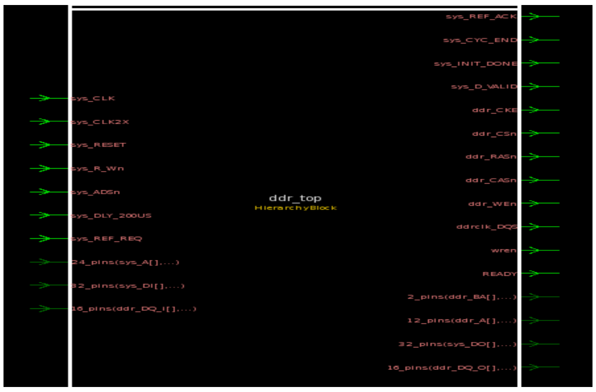


Fig 5. *Top level schematic of the Controller*

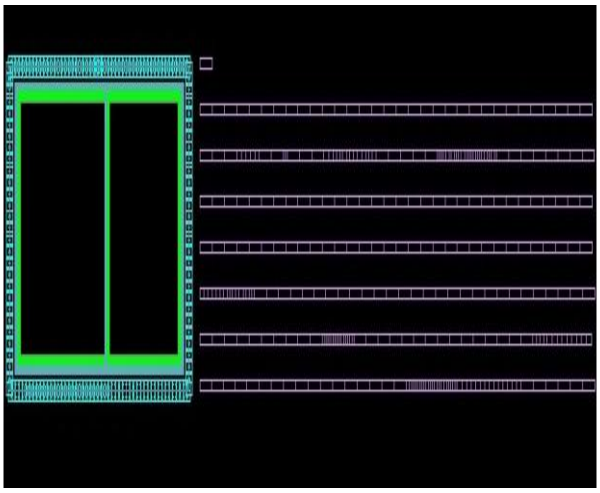


Fig 6. *Post Floorplan schematic of the controller*

iii) Clock Tree Synthesis (CTS):

Until now in the design the clocks that were used were ideal clock sources defined during the synthesis phase. CTS is the phase during which the clock would be actually propagated through the design. The main reason to perform CTS is to reduce clock skew and insertion delays. As the name suggests the clock would be synthesized as a H-tree and is fed to all the sequential elements present in the design. This tree is synthesized using buffer cells called clock buffers and placed at appropriate positions to reduce the skew and insertion delay. Hold slack if any is improved after CTS. *clock_opt* is the command that would perform CTS. Figure 8 shows the controller design post CTS.

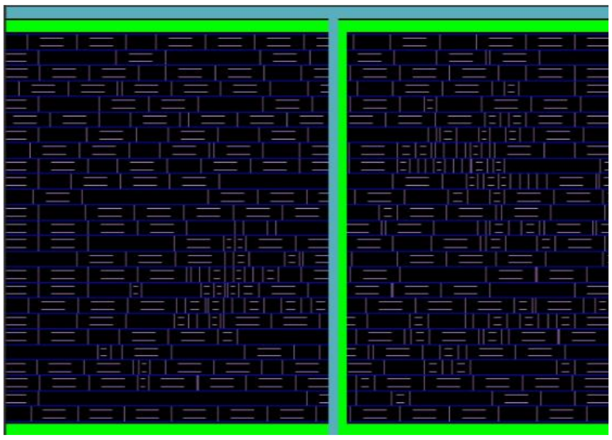


Fig 7. *Post placement schematic of the controller*

iv) Routing:

Routing is the final phase in the place and route of the controller design. All the placed standard cells would receive actual physical interconnections between them depending on the netlist. Routing is done in two steps which are Global routing and detailed routing. Global routing would provide routing resources that provide the connections. Specific metal layers and routing tracks within the global routing resources for detailed routing assigns routes. Cross talk reduction is also process during this. The routing is done using the *route_opt* command. Figure 9 shows a sample routed area of the controller design.

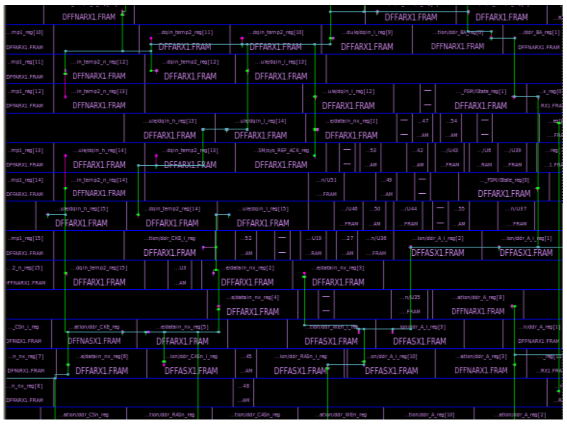


Fig 8. *Post CTS structure of the controller*

5. RESULTS

The design is simulated using Synopsys VCS to verify the functionality. Logic synthesis of the verified RTL was performed using Synopsys Design Compiler and the Place and Route was performed using Synopsys IC Compiler.

Figures 10 and 11 shows the internal schematic of the DDR SDRAM controller showing all the modules after synthesis and the completely routed controller post routing. The reports related to synthesis and the physical design phase are generated and tabulated in Table 1 and Table 2 respectively.

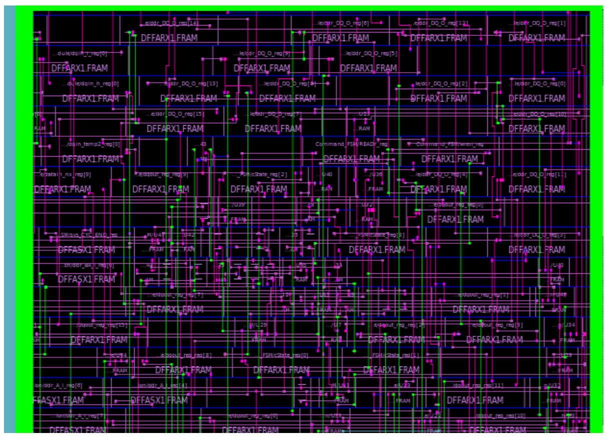


Fig 9. *Sample routed area of the DDR SDRAM controller*

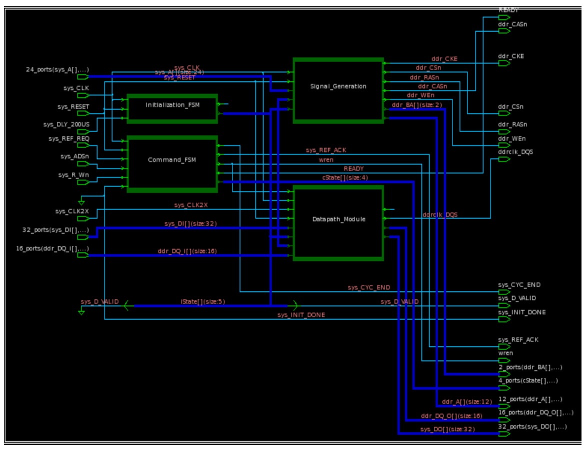


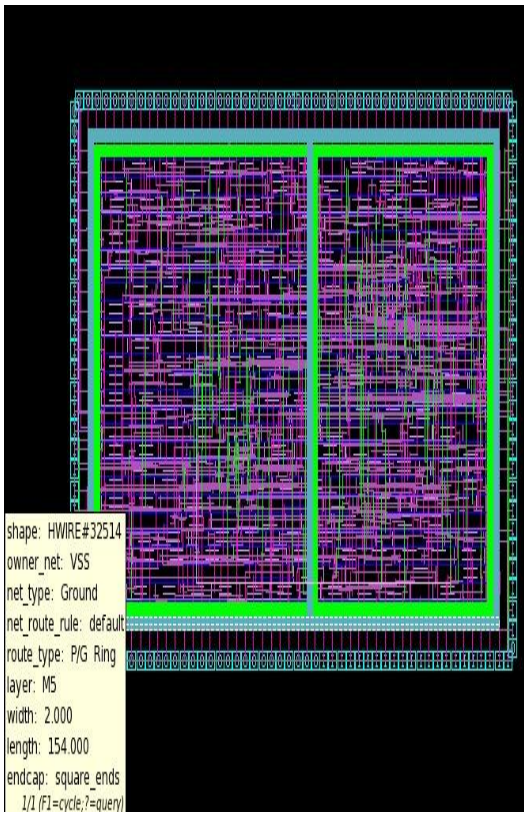
Fig 10. *Internal schematic of the DDR SDRAM controller*

Table 1. *Synthesis results of the controller*

Design Area (nm^2)	7611.81
Cell count	356
Dynamic power (μw)	101.36
Leakage power (μw)	28.66
Clock period (ns)	1.20

Table 2. *Physical design results of the DDR SDRAM memory Controller*

Design Area (nm^2)	7081.13
Dynamic power (mW)	5.9898
Leakage power (μw)	28.82
Net Length(nm)	12879



shape: HWIRE#32514
owner_net: VSS
net_type: Ground
net_route_rule: default
route_type: P/G Ring
layer: M5
width: 2.000
length: 154.000
endcap: square_ends
1/1 (F1=cycle;?=query)

Fig 11. *Completely routed DDR SDRAM Memory Controller*

6. CONCLUSION

Logic Synthesis and Physical Design Implementation of 32-bit DDR SDRAM Memory Controller is done which forms a part of the ASIC Design Flow at 90 nm technology node. The CAD tools used for Synthesis were Synopsys Design Compiler and Synopsys IC Compiler for Physical Design Flow. Future work may include performing the same for a smaller technology node.

REFERENCES

[1] DDR SDRAM Controller white paper, Lattice Semiconductor Corporation, Reference Design: RD1020, April 2004.

[2] Amit Bakshi, SudhanshuShekhar Pandey, Tribikram Pradhan Ratnadip Dey, “ASIC Implementation of DDR SDRAM Memory Controller”,2013 IEEE International Conference on Emerging Trends in Computing, Communication and Nanotechnology (ICECCN 2013).

[3] http://www.freescale.com/files/32bit/doc/app_note/AN2910.pdf

[4] http://userwww.sfsu.edu/necrc/files/synopsys%20tutorials/ASIC%20Design%20Flow%20Tutorial.pdf.

[5] Synopsys Design Compiler® User Guide Version D-2010.03-SP2, June 2010.

[6] Synopsys IC Compiler® User Guide Version B-2008-09, 2008.