System Verilog based Verification of Write Operation in SDRAM using Memory Controller

Sandeep Raval[1]
[1]M. E. (VLSI and Embedded System Design)
[1]GTU PG School, Ahmedabad

Abstract---In this paper, a Write Operation is done into the SDR SDRAM using Memory Controller that is verified. Basically as the memory, the basic Write and Read operation is performed to it. To see this Functionality working correctly, it needs to be verified. For this the Verification Environment is to be created. The Interface, Program Block, Generator, Driver, Monitor and Scoreboard are the basic components of the Verification Environment. This Paper describes the basic Write operation to the SDRAM is Verified to check the functionality. This Operation is performed using System Verilog and Modelsim 6.3f. The Address and the Data size is 32bits and 8bit Memory Model is used to perform this operation.

Keyword: SDRAM, Verification, System Virology, Modelsim 6.3f, Write Operation.

I. INTRODUCTION

Synchronous DRAM (SDRAM) has become a mainstream memory of choice in embedded system memory design. For high-end applications using processors the interface to the SDRAM is supported by the processor's built-in peripheral module. However, for other applications, the system designer must design a controller to provide proper commands for SDRAM initialization, read/write accesses and memory refresh. SDRAM controller located between the SDRAM and the bus master reduces the user's effort to deal with the SDRAM command interface by providing a simple generic system interface to the bus master. Figure 1 shows the relationship of the controller between the bus master and SDRAM. The bus master can be either a microprocessor or a user's proprietary module interface. [5]

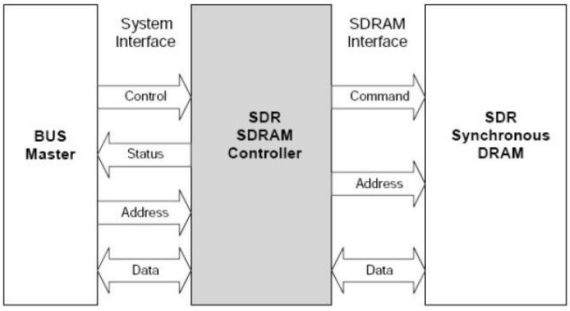

Fig.1: Interfacing of SDRAM with BUS

II. SDRAM

SDRAM (shown in Figure 2) is high-speed Dynamic Random Access Memory (DRAM) with a synchronous interface. The synchronous interface and fully pipelined internal architecture of SDRAM allows extremely fast data rates if used efficiently. SDRAM is organized in banks of memory addressed by row and column. The number of row and column address bits depends on the size and configuration of the memory. [1]

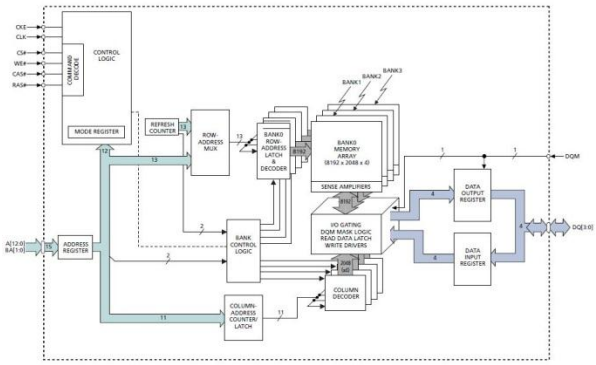

Fig. 2: SDRAM Structure Source: Micron

III. FUNCTIONAL DESCRIPTION

A. Wish Bone Bus Handler: This block handles the Protocol handshake between wish bone master and custom SDRAM controller. This block also takes care of necessary clock domain change over. This block includes; Command Asynchronous FIFO, Write Data Asynchronous FIFO, Read Data Asynchronous FIFO.

B. SDRAM Controller: This block includes four sub blocks:

1) SDRAM Bus convertor: This block convert and re-align the system side 32bit into equivalent 8/16/32 SDR format.
2) SDRAM Request Generator: To generate the Request in bidirectional.
3) SDRAM bank Controller: This module takes requests from SDRAM request generator; checks for page hit/miss and issue precharge/activate commands and then pass the request to SDRAM Transfer Controller.
4) SDRAM Transfer Controller: This module takes requests from SDRAM Bank controller, runs the transfer and controls data flow to/from the app. At the end of the transfer it issues a burst terminate if not at the end of a burst and another command to this bank is not available.

C. SDRAM Interface: Prior to normal operation, SDRAM must be initialized. The following sections provide detailed information covering device initialization, register definition, command descriptions and device operation. [2]

IV. SIGNALS TO WRITE THE DATA

Port	Direction	Description
wb_clk_i	Input	Master clock
wb_adr_i	Input	32 bit Lower address bits
wb_dat_i	Input	32 bit Data towards the core
wb_dat_o	Output	32 bit Data from the core
wb_we_i	Input	Write enable input
wb_ack_o	Output	Bus cycle acknowledge output
sdr_dqm	Output	2/4 bit SDRAM data bus mask
Dq	Output	16/32 bit SDRAM Data Output

All rights reserved by www.ijsrd.com

V. VERIFICATION PLAN

The Verification plan is the base of the verification which is intended to check that a Design meets a set of design specifications. The design is made of some features and based on that features we generate test plan which has the test cases which are to be tested in the verification environment that is created. The bugs will be found, if there is any in our design through these cases applied to our test cases. The Code coverage and the functional coverage are verified for the DUT which are explained in detailed below. [3]

A. Steps For Verification

The Specification along with the RTL is to be given by the Designer which consists of the features, functionality, and Configuration parameter.

In Verification plan, creation of the particular verification environment & generation the testbench of the components. Test cases will be applied and we shall see the pass/fail cases and fix the bugs. Regression/Simulation will check whether any error or not and all test should be passed with clean up. In the Coverage, the code coverage & the functional Coverage will be covered. It is shown in the Figure 3.

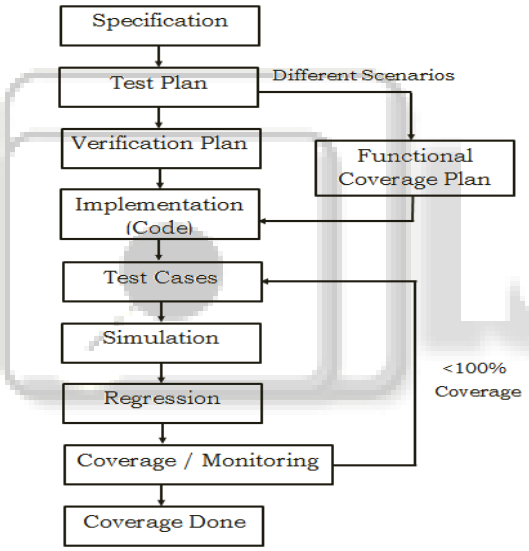

Fig. 3: ASIC Verification Flow

VI. VERIFICATION ENVIRONMENT

Typical Verification architecture looks as shown below in Figure 4. The main blocks in a testbench are base DUT (Design Under Test), transaction generator, driver, monitor and checker/scoreboard.

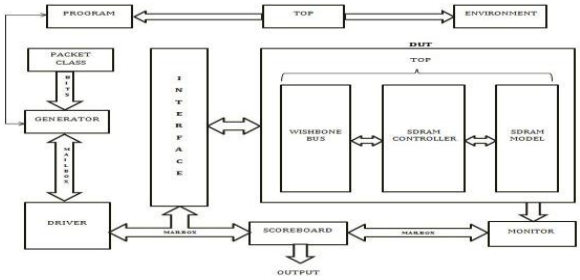

Fig. 4: Verification Environment

A. Verification Environment Component

1) Base Object: Base object is the data structure that will be used across the testbench. Normally we have some default constraints and some methods (functions) which could manipulate the objects in the base object.

2) Driver: Driver drives the base object generated by the transaction generator to the DUT. To do this, it implements the DUT input protocol.

3) Monitor: Monitor monitors the input signals to the DUT. Each ingoing packet is picked by the monitor and passed to the checker.

4) Checker/Scoreboard: Checker or Scoreboard basically checks if the output coming out of the DUT is correct or wrong. Basically scoreboards in e language are implemented using keyed lists. [6]

B. SDRAM (Memory) Model: Memory Model is nothing but the SDRAM file in which the number of data is to be written and read.

VII. WRITE OPERATION

SDRAM is controlled by bus commands that are formed using combinations of the ras,cas, and we signals. For instance, on a clock cycle where all three signals are high, the associated command is a No Operation (NOP). A NOP is also indicated when the chip select is not asserted.

SDRAM devices are typically divided into four banks. These banks must be opened before a range of addresses can be written to or read from. The row and bank to be opened are registered coincident with the Active command. When a new row on a bank is accessed for a read or a write it may be necessary to first close the bank and then re-open the bank to the new row. Closing a bank is performed using the Precharge command. Opening and closing banks costs memory bandwidth, so the SDRAM Controller Core has been designed to monitor and manage the status of the four banks simultaneously. This enables the controller to intelligently open and close banks only when necessary.

When the Write command is issued, the initial column address is presented to the SDRAM devices. The initial data is presented concurrent with the Write command. For the read command, the initial data appears on the data bus 1-4 clock cycles later. This is known as CAS latency and is due to the time required to physically read the internal DRAM and register the data on the bus. The CAS latency depends on the speed grade of the SDRAM and the frequency of the memory clock. In general, the faster the clock, the more cycles of CAS latency is required. After the initial Read or Write command, sequential read and writes will continue until the burst length is reached or a Burst Terminate command is issued. SDRAM devices support a burst length of up to 8 data cycles. The SDRAM Controller Core is capable of cascading bursts to maximize SDRAM bandwidth.

SDRAM devices require periodic refresh operations to maintain the integrity of the stored data. The SDRAM Controller Core automatically issues the Auto Refresh command periodically. No user intervention is required.

The Load Mode Register command is used to configure the SDRAM operation. This register stores the CAS latency, burst length, burst type, and write burst mode. The SDR controller only writes to the base mode register. [2]

All rights reserved by www.ijsrd.com

VIII. SIMULATION RESULT

Fig. 5: Write Operation Simulation Result

IX. CONCLUSION

- A directed test bench is designed to simulate the write Operation from master bus i.e. Wishbone Bus.
- The Write Operation into the SDR SDRAM is verified through Memory Controller using System Verilog.
- A Simulation Report of process is performed using Modelsim.

REFERENCES

[1] A Reference Manual for the SDR SDRAM Memory model by Micron available at http://www.micron.com /products/dram/sdram
[2] A SDRAM Controller Specification Reference Manual by Open cores available athttp://opencores.org /project,sdr_ctrl
[3] Mansour H. Assaf , Sunil R. Das, Wael Hermas, and Wen -B. Jone, Promising Complex ASIC Design Verification Methodology, Instrumentation, 1-3 May 2007
[4] Chris Spear, System Verilog for Verification: Springer, 2006.
[5] Joseph Gross, High-Performance DRAM System Design Constraints and Considerations, Faculty of the Graduate School of the University of Maryland.2010
[6] ASIC Verification Online Tutorials available at http://www.asic-world.com/tidbits/typical_ verification. html
[7] ASIC Verification Online Tutorials available at http://testbench.in/TS_03_FUNCTIONAL_VERIFICA TION_NEED.html

All rights reserved by www.ijsrd.com