International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 04 Issue 13
October 2017

Implementation And Verification Of Generic Universal Memory Controller Based On Sv

P NAVYA SREE [1], W POORNIMA [2], P SRIKANTH [3]

1. P. NAVYA SREE, M.tech student, Dept of Ece, Mallareddy Engineering College For Women (MRECW),
2. Guide details: W.POORNIMA, M.tech, Assistant Professor, Mallareddy Engineering College For Women(MRECW)
3. Guide details: P.SRIKANTH, M.tech, working as senior verification engineer at wipro in vlsi domain

Abstract— This paper presents a coverage driven constraint random based functional verification method based on the System Verilog (SV) for Generic Universal Memory Controller Architecture. It improves the performance of the existing memory controllers through a complete integration of the existing memory controllers features in addition of providing novel features. It also reduces the power consumption by using different power levels supported to fit all power scenarios.

Some of the coverage points have been covered to verify the validation of the integrated features which makes the proposed universal memory controller replaces the existing controllers on the scene as it provides all of their powerful features in addition of novel features to control two of the most dominated types of memory; FLASH and DRAM, SRAM through one memory controller.

Index Terms— Universal Memory Controller, low power Memory Controller, Flash, DRAM, UVM, eMMC, ONFI, One-NAND, UFS, HMC, WideIO, SSD, Verification Environment.

I. INTRODUCTION

The main aim of the memory controllers is to provide the most suitable interface and protocol between the host and the memories to efficiently handle data, maximizing transfer speed, data integrity and information retention. To improve this communication as a solution for the memory bottleneck, the memory cores and memory controllers can be improved. The most famous existing memory controllers–based solution is to improve the controller architectures and scheduling algorithms. Part of the idea behind the solution is to unload low-level memory management from the host processor, freeing up resources. The proposed memory controller optimizes the features of the existing memory controllers on the scene and integrates them in only one generic universal memory controller. This generic universal memory controller can replace the existing memory controllers due to its powerful specifications which can be summarized in these points: (i) Two dominated types of memory: FLASH and DRAM and SRAM are supported. (ii) All the most major, powerful and important features for any existing memory controller are supported and designed to be optionally enabled or disabled according to the manufacturer desire. (iii) High control of power consumption due to different power levels supported to fit all power scenarios.

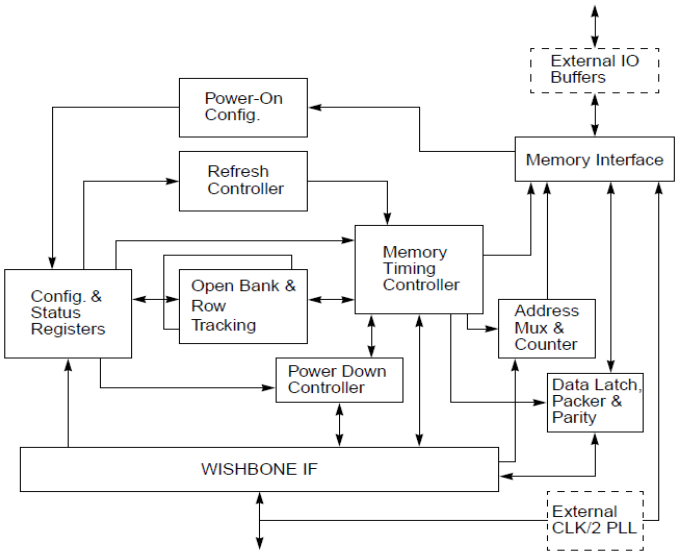

Fig.1 Memory Controller architecture

This generic universal Memory Controller core supports a variety of memory devices, flexible timing and predefined system startup from a Flash or ROM memory. Microprocessors communicate with memory cores through memory controllers.

Some of the main features are:

1. SDRAM, SSRAM, FLASH, ROM and many other devices supported. 2. 8 Chip selects, each uniquely programmable. 3. Flexible timing to accommodate a variety of memory devices. 4. Burst transfers and burst termination. 5. Supports RMW cycles. 6. Performance optimization by leaving active rows open. 7. Default boot sequence support. 8. Dynamic bus sizing for reading from Async. Devices. 9. Byte parity Generation and Checking. 10. Multi Master memory bus support. 11. Industry standard WISHBONE SOC host interface. 12. Up to 8 * 64 Mbyte memory size. 13. Supports Power Down Mode.

Our key contribution in this paper is to provide some implementation and verification details which are used in creating this universal memory controller by using sv. The rest of paper is organized as follows. In Section II, some of the implementation details are presented. In section III, the architecture of the SV verification environment is provided. Results are discussed in section IV. Conclusions are in section V.

II. SOME OF THE IMPLEMENTATION DETAILS ARE PRESENT

These specifications covered the most important types of memories are FLASH and DRAM and SRAM in one device. This architecture has extremely simple design which can utilize many applications.

Some of the different memory controllers are present

DRAM CONTROLLER:-

Dynamic random-access memory (DRAM) is a type of random-access memory that stores each bit of data in a separate capacitor within an integrated circuit.

Double data rate (DDR) memory controllers are used to drive DDR SDRAM, where data is transferred on both rising and falling edges of the system's memory clock. DDR memory controllers are significantly more complicated when compared to single data rate controllers, but they allow for twice the data to be transferred without increasing the memory cell's clock rate or bus width. Slower in access, charge leakage are the disadvantages among which lower cost is the advantage.

FLASH CONTROLLER:-

It manages the data stored on flash memory and communicates with a computer or electronic device. Flash memory controllers can be designed for operating in low duty-cycle environments like SD cards, Compact Flash cards, or other similar media for use in digital cameras, PDAs, mobile phones, etc. Flash controllers can also be designed for higher duty-cycle environments like solid-state drives (SSD) used as data storage for laptop computer systems clear up to mission-critical enterprise storage arrays.

SRAM CONTROLLER:-

A memory controller is usually used to shield the synchronous system from SRAM. It is responsible for generating the properly timed signals and making the SRAM look 'synchronous'. Its performance is measured by the number of memory accesses that can be completed in a given time period. Designing a memory controller that is optimal is non-trivial.

The host can handle the power consumption efficiently. Through a comparative study of these protocols, an important key result is reached, although that the industry always looking forward to improving the performance

The main objective is to reduce the consumed power by the device. Thus the major objective also of this common architecture is to conserve the energy. The interface between the host processor and the proposed universal memory controller consists of six buses which enable the host to send/receive to/from the memory controller serially . The host processor can also select the desired memory core (FLASH or DRAM or SRAM) through the memory core select signal.

This universal memory controller proposes six different partitions which are boot, enhanced, system code, high speed, temporary and the user data area partition to benefit from the permanent storage of the controlled memory.

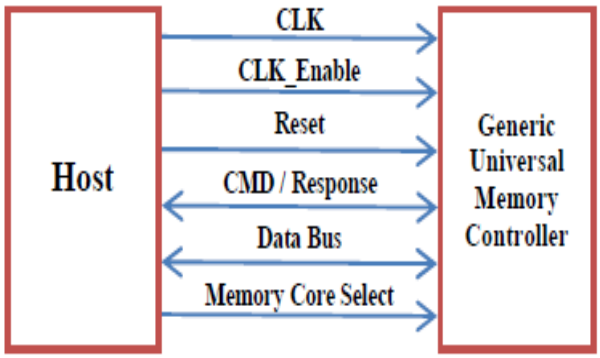

Fig.2 The proposed interface of the universal memory controller.

Therefore this Universal Memory Controller core supports a variety of memory devices like FLASH, DRAM, SRAM, SYNC CHIP SELECT..

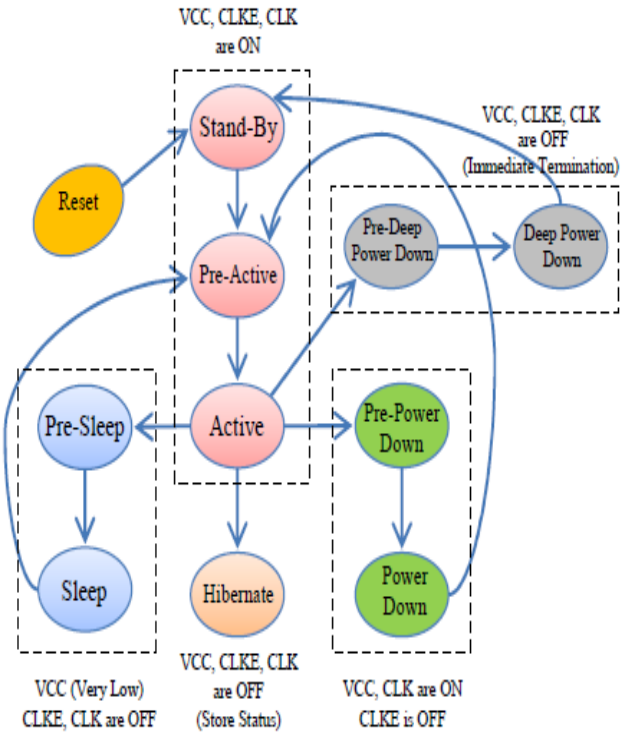

Fig.3 The Proposed power levels of the Universal Memory Controller are stand-by, active, power-down, sleep, hibernate and deep power down

The interface between the host processor and the proposed universal memory controller consists of six buses which enable the host to send/receive to/from the memory controller serially as shown in Fig.2. The host processor can also select the desired memory core (FLASH or DRAM) through the memory core select signal

III. THE ARCHITECTURE OF THE SV VERIFICATION ENVIRONMENT

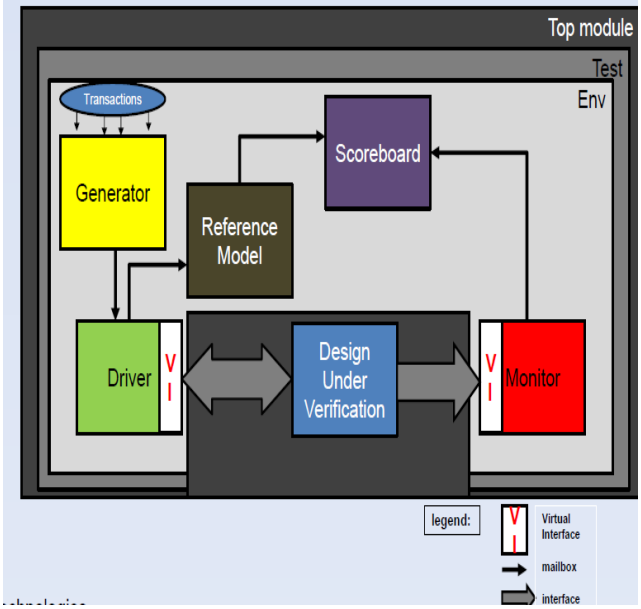

Fig.4 The Proposed Architecture of the SV verification environment for Generic Universal Memory Controller

BUILDING BLOCKS OF THE SV TESTBENCH ARCHITECTURE:

Transactions: All the inputs and outputs of the Design Under Verification (DUV) are written inside the transaction class, excluding the clock and reset signals that are generated in the top module. The transaction class can optionally have constraints for the inputs.

Generator: This is a component of the testbench which generates constrained random stimuli (transactions) for the DUV. The generator sends the generated stimuli to the driver through a mailbox. Its code is written inside a class.

Driver : This is a component which converts transactions into pin-level activity at the inputs of the DUV. It receives the generated transactions from the generator through a mailbox and drives it to the DUV through a virtual interface according to the DUV protocol. It also sends the received transactions from the generator to the reference model (described later) through another mailbox. Its code is written inside a class.

Monitor : This is a component which converts pin-level activity into transactions at the outputs of the DUV. It collects the outputs of the DUV through a virtual interface and sends it to the scoreboard through a mailbox. Its code is written inside a class.

Reference Model : This is a 'golden' model that mimics the functionality of the DUV and generates reference results used for comparison against simulation results.

Scoreboard: This is the component that collects the transactions (expected results) from the reference model through a mailbox.

- It collects the transactions (actual results) from the monitor through another mailbox.
- It compares the expected transactions with the actual transactions and generates a report.
- Its code is written inside a class

Environment : This is the component of the testbench which instantiates the generator, driver, monitor, reference model and scoreboard

- Here, the various sub-components are built and properly connected.
- Its code is written inside a class.

Test : This is the component where different test cases are written and run. Here the environment is instantiated and built. Its code is written inside a class.

Top : This is the top most module of the SV testbench architecture where control signals like clock and reset are generated. Its code is written inside a module and the interface, the DUV and the test are instantiated in it.

IV. MEMORY CONTROLLER FUNCTIONAL VERIFICATION

The Verification of Memory Controller is done as below.

- Initially the memory controller specification is read and features are extracted from it.
- The verification plan document is then developed with the information containing the feature based testcases like registers Read/Write, performance based testcases, interrupt testcases, low power testcases etc.
- Architecture components that fits to verify. Coverage to be covered, assertions for protocol verification. Scoreboard and checker information, interfaces, configuration parameters etc.
- The testbench and it's components is developed using System verilog and different testcases are written.
- The different stimulus is generated using randomized based sequences and configuration settings and given to the DUT through interface bus.
- The input and output from DUT is monitored and comparison is done
- In the scoreboard for input data and expected output data from DUT.
- The coverage report is analyzed and more number of randomized and direct testcases is written to achieve

the 100% code and functional coverage which is the main criteria for the verification completion.

QUESTASIM TOOL:-

QuestaSim is part of the Questa Advanced Functional Verification Platform and is the latest tool in Mentor Graphics tool suite for Functional Verification. The tool provides simulation support for latest standards of SystemC, SystemVerilog, Verilog 2001 standard and VHDL. This tool is an advancement over Modelsim in its support for advanced Verification features like coverage databases, coverage driven verification, working with assertions, SystemVerilog constrained-random functionality.

V. RESULTS

The implementation and verification of a universal memory controller is done using system verilog. Waveform showing the writing data to particular address location in the burst transaction and same data is read from the written address location. And both data should be same. This type read and writes sequences are done for the SRAM, FLASH memories to verify it.

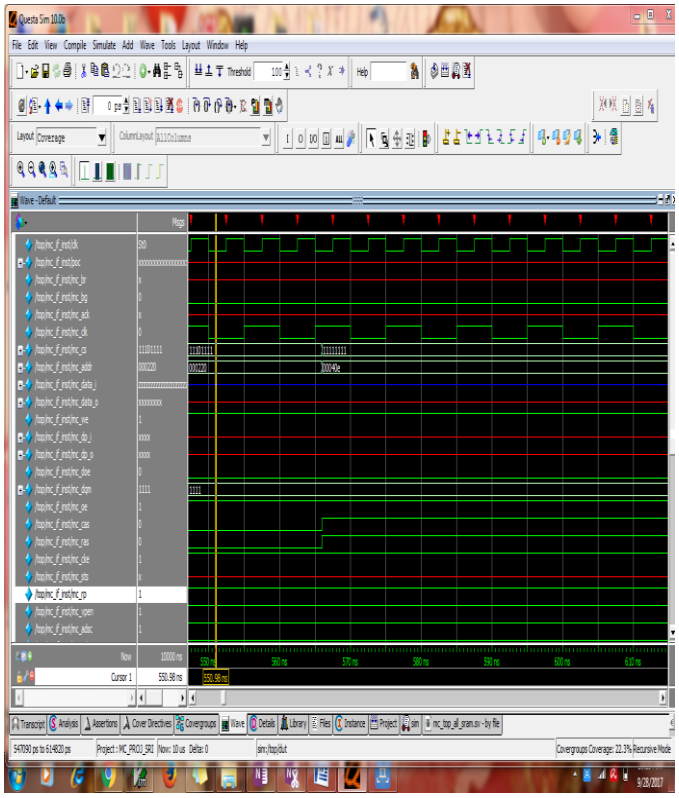

Fig 6 Waveforms of the interesting signals of the proposed interface

Though the tests plan, more than 200 coverage points have been covered. Since all the signals are statistically available, the interesting data sings of universal memory controller are added to the wave viewer and looking at their values over time while the simulation is running as shown in Fig6.

The wr_rd represents the read and write data where if wr_rd is 1 then it represents write the data and if 0 , it represent read data. In the below analysis the data is written at particular address location of memory.

This below analysis shows that whatever the data written in particular address is same at the reading the data at same addresses. This represents the complete verification of a universal memory controller using system verilog.

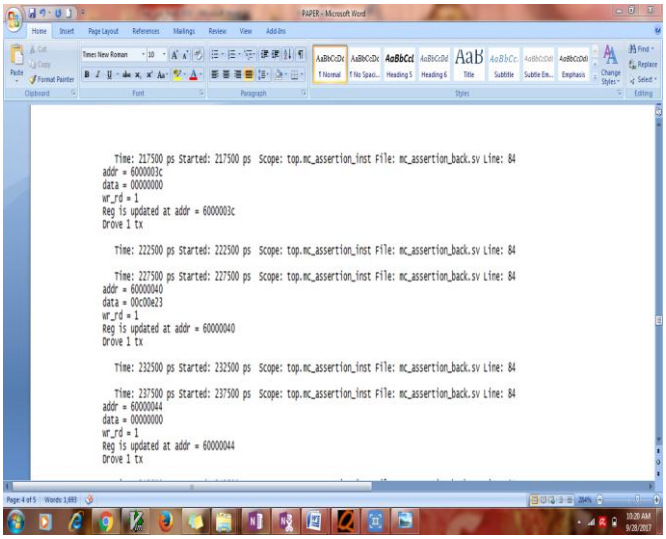

Fig 7 write data analysis

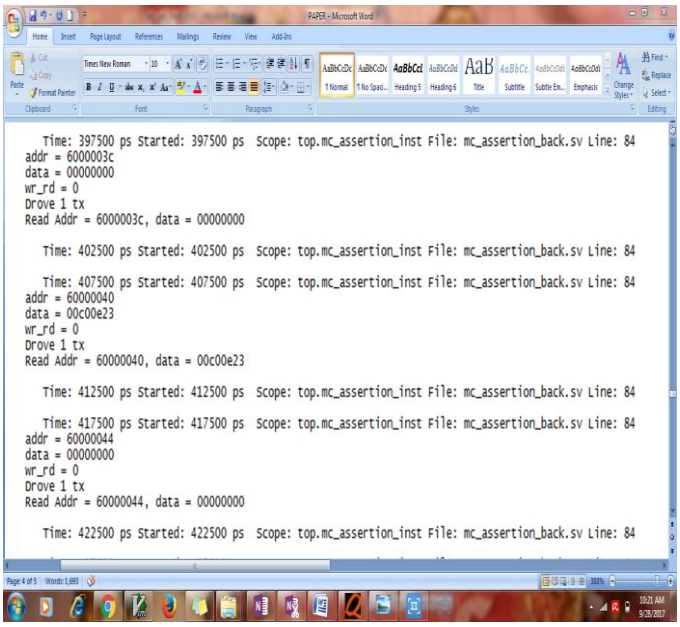

Fig 8 Read data analysis

VI. CONCLUSION

This paper proposed an implementation of a system level architecture of a GENERIC UNIVERSAL MEMORY CONTROLLER verified by using SYSTEM VERILOG . It supports both FLASH and DRAM memory types to be controlled.

It also grants to the host processor the ability of high power consumption control due to its proposed different power levels supported to fit all power scenarios, so the controller is keeping up with the global trend to save more power and reduce its impact on the performance.

Memory controller improves the memory access efficiency in applications requiring small burst and random addressing accesses such as image processing and networking.

.

VII. REFERENCE

- El-Ashry, S., memory controller architecture
- K Khalifa, H Fawzy Computer, Telecommunications and CON), 2014 11th…
- Systemverilog 1800-2012 IEEE Standard for System Verilog-Unified Hardware Design, Specification, and Verification Language
- http://www.verilog.net/
- http://www.asic-world.com/verilog/veritut.html.