International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 05, May-2020

A System Verilog Approach for Verification of Memory Controller

Gagana P, Mythili M
Students: Electronics and Communication Engineering
RV College of Engineering
Bengaluru, India

Dr. Kariyappa B S
Professor: Electronics and Communication Engineering
RV College of Engineering
Bengaluru, India

***Abstract*— The major bottleneck of the computer system to improve the overall performance, is the memory performance. An effective control of data between processor and memory can be done using memory controller. Verification plays a vital role in any design flow as it is completed before silicon development. In this work a memory controller for interfacing Synchronous Static Random Access Memory (SSRAM), Synchronous Dynamic Random Access Memory (SDRAM), Read Only Memory (ROM) and FLASH which is Electrically Erasable Programmable Read-Only Memory is designed and a Constraint random verification environment which is coverage driven, is built for the designed memory controller. The code is written in Verilog HDL and the verification is done using System Verilog. Using Questa Sim 10.0b software tool the simulation is done, and coverage results are obtained. The verification environment built in this work, gives a functional coverage of 96.8% and assertion success of 100% with 0% assertion failures. Simulation results show that the designed controller gave good performance and full filled all the system specifications.**

Keywords*— Wishbone interface, functional coverage, verification environment, Assertion.

I. INTRODUCTION

A circuit that can manage the flow of data from the computer's main memory to the processor and back to it, is a Memory Controller [1]. This can be integrated into another chip or as an integral part of a microprocessor which is called Integrated Memory Controller (IMC). In this paper, the universal Memory Controller core supports a variety of memory devices, SDRAM, SSRAM, FLASH, ROM and flexible timing.

II. LITERATURE SURVEY

The recent literature shows that the design of memory controller is getting optimized day by day. In [2] the author specifies about the specifications required for the design of memory controller like the configuration of registers and the address range for each of them. It also gave the information about how to connect various memory devices to the Memory Controller. It also specified the clock frequency and other constraints pertaining to memory, for example the refresh cycle time period for SDRAM. In [3] the author explains the basic functionality of Standard Single Data Rate-Synchronous Dynamic Random Access Memory (SDR SDRAM) devices, such as the command structure. In addition to this, Double Data Rate Synchronous Dynamic Random-Access Memory (DDR SDRAM) has some enhancements like Source synchronous operation, double data rate, differential clocks and low voltage signaling [4].

Heterogenous multicore systems have high latency. The author in [5] proposes a shared memory approach between CPU and GPU, so that only the address can be exchanged rather than transferring the content hence reducing the latency. Different degrees of memory parallelism are proposed that is achieved through different number of memory controllers which is applied to different types of region, hence it is called region aware MC. In [6] the author discusses the design and verification aspect of DRAM cache controller (DCC) by modelling DCC using interacting state machines with 87% coverage. The design controls both the timing aspects of DRAM system as well as the functional aspects of cache (insertion, access, writeback) that is checked using formal verification. The data structures such as Binary decision diagram and Satisfiability solvers are used in formal verification to perform equivalence check [7].

Prince Gurha [8] describes how SystemVerilog Assertions can be used in order to verify a system by considering AMBA-Advanced High-performance Bus (AHB). The AMBA AHB RTL is modeled using Verilog (3 masters and 4 slaves). The design verification is then carried out by making use of assertions and the binding construct using ModelSim [9]. Binding enables addition of assertions to design without modifying the RTL design files. The various functionalities of AMBA-AHB and the corner cases are considered and simulated using ModelSim. The coverage metrics are calculated. 80% functional coverage is achieved by using around 20 assertions. The system is designed using SystemC-TLM 2.0, and SystemVerilog assertions are used for verification.

In summary, there are different variations of the memory controller architecture as discussed in the literature. The universal memory controller designed in this paper, is derived by combining the different architectures such as NAND flash memory based, SDRAM controller, SRAM based memory controller discussed above with certain modifications so as to get the optimized design.

III. VERIFICATION ENVIRONMENT FOR MEMORY CONTROLLER

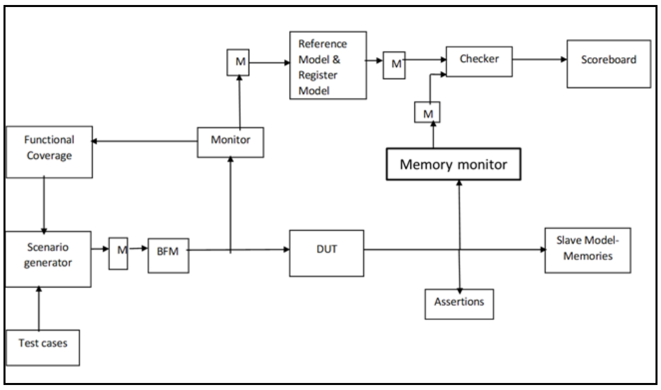


Fig. 1.Verification Environment for Memory Controller

The functionality of the blocks in Fig. 1 is:

- **Scenario generator:** The scenario generator stimulates the device in ways designed to exercise interesting behaviors. Mailbox is used for communication between the blocks.
- **Driver (BFM):** The driver converts transaction-level stimulus into pin wiggles on a bus or pin-level interface.
- **Monitor:** The monitor is the complement to the driver. It observes signals on a bus or pin-level interface and converts them to transactions.
- **Slave Model Memories:** These are the peripheral memory devices, SRAM, SDRAM, FLASH, ROM.
- **Assertions:** Assertion checks and corresponding binding files are placed in this directory
- **Functional Coverage:** It observes activity on monitors and keeps track of which functions have been observed.
- **Scoreboard:** It contains a reference model and does the comparison of the operation of the DUT with the reference [10]. Any mismatch between the DUT and the reference is known as a “scoreboard failure”. A scoreboard failure represents a problem in either the DUT or the scoreboard that must be investigated.

IV. TEST PLAN

The Fig. 2 shows the test plan of how the verification is carried out. The first testcase which is run, is for checking the registers if they are performing reset, write, read operations correctly, that is without any error. Then only one type of memory is connected to the chip selects of the Memory Controller and checked for correct memory reads after performing memory writes, that is, connecting only SRAM, then, only DRAM, followed by FLASH and ROM. After verifying that each memory is working correct when connected individually, all the four types of memories are now connected to the different chip selects of the Memory controller. Now, this is verified for correct data and timing configurations as the expected ones. The suspend and resume operation is also checked for this connection. Lastly, functional coverage obtained for chip selects with all the four memories, SRAM, SDRAM, FLASH, ROM is analyzed.

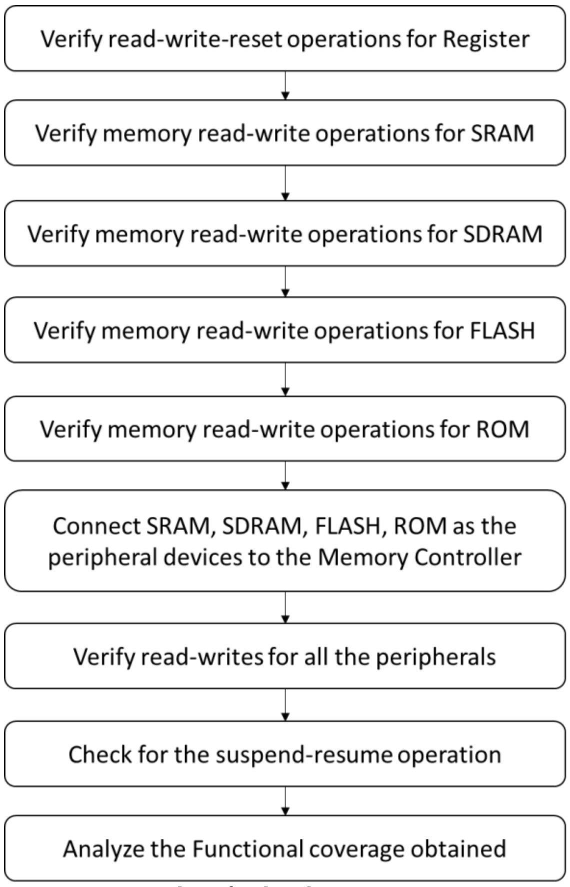

Verify read-write-reset operations for Register
Verify memory read-write operations for SRAM
Verify memory read-write operations for SDRAM
Verify memory read-write operations for FLASH
Verify memory read-write operations for ROM
Connect SRAM, SDRAM, FLASH, ROM as the peripheral devices to the Memory Controller
Verify read-writes for all the peripherals
Check for the suspend-resume operation
Analyze the Functional coverage obtained

Fig. 2.Flow of coding the testcases

The following are the list of testcases implemented:

A. Test_register_reset

In this testcase, the reset signal is applied and released so as to verify if the register contents are actually reset on applying a synchronous reset signal as shown in Fig. 3.

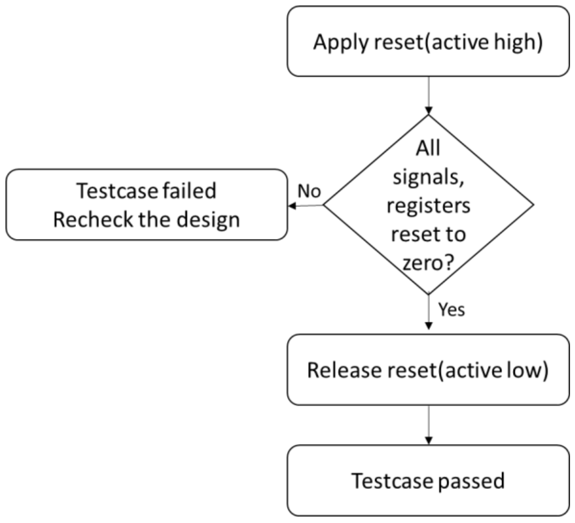

Apply reset(active high)
All signals, registers reset to zero?
No
Testcase failed Recheck the design
Yes
Release reset(active low)
Testcase passed

Fig. 3.Register reset testcase

B. Test_register_write_read

In this testcase, write and read operations are performed to the registers to check for data corruption, that is, if the data written to a particular address is the same when read back from the same location as shown in Fig. 4.

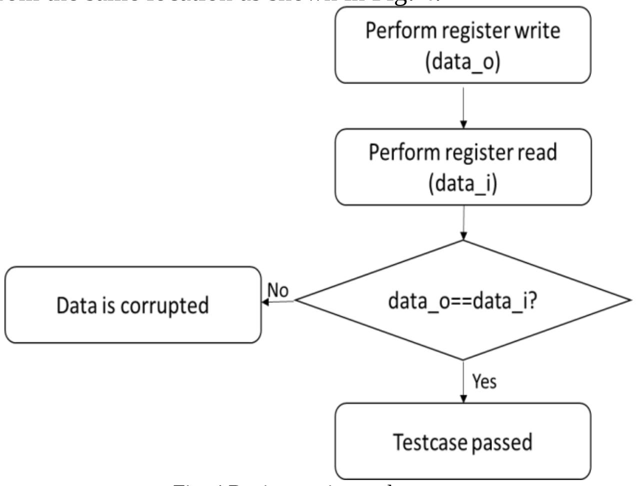

Perform register write (data_o)
Perform register read (data_i)
data_o==data_i?
No
Data is corrupted
Yes
Testcase passed

Fig. 4.Register write-read testcase

C. Test_all_chipselects_sram

In this testcase, write and read operations are performed to the SRAM chips connected so as to check for data corruption only after configuring the registers to operate for SRAM as shown in Fig. 5, similarly the write-read test cases can be carried out for SDRAM, Flash and ROM.

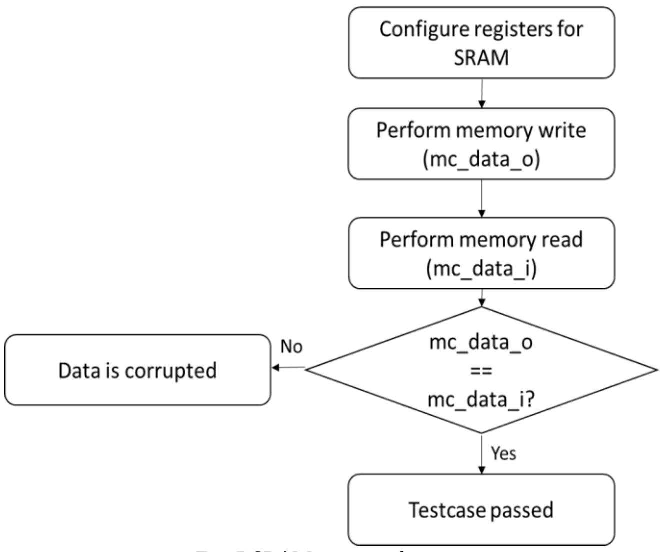

Configure registers for SRAM
Perform memory write (mc_data_o)
Perform memory read (mc_data_i)
mc_data_o == mc_data_i?
No
Data is corrupted
Yes
Testcase passed

Fig. 5.SRAM write-read testcase

D. Test_all_chipselects_sdram

In this testcase, write and read operations are performed to the SDRAM chips connected so as to check for data corruption only after configuring the registers to operate for SDRAM.

E. Test_all_chipselects_flash

In this testcase, write and read operations are performed to the FLASH to check for data corruption after configuring the registers to operate for FLASH.

F. Test_all_chipselects_rom

In this testcase, write and read operations are performed to the ROM chips connected so as to check for data corruption only after configuring the registers to operate for ROM.

G. Test_all_chipselects_different_memories

In this testcase, write and read operations are performed to all the four different memory chips, that is, SRAM, SDRAM, FLASH, ROM connected so as to check for data corruption only after configuring the registers to operate for different memories as shown in Fig. 6.

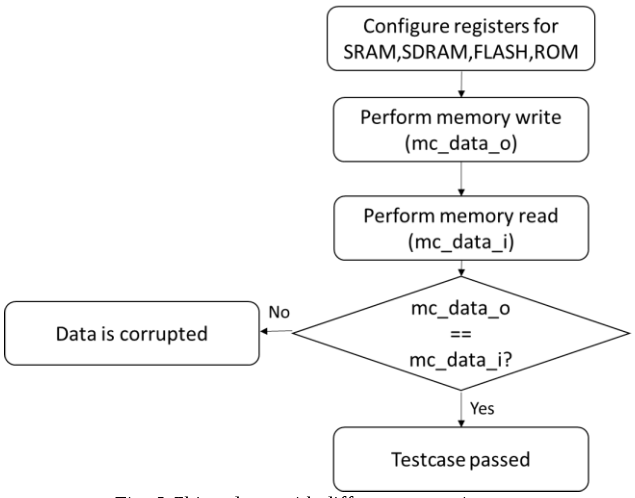

Configure registers for SRAM,SDRAM,FLASH,ROM
Perform memory write (mc_data_o)
Perform memory read (mc_data_i)
mc_data_o == mc_data_i?
No
Data is corrupted
Yes
Testcase passed

Fig. 6.Chip selects with different memories testcase

H. Test_suspend_resume

In this testcase, after configuring the registers to operate for different memories, the suspend signal is pulled high after one write operation and after that read operation is performed. This testcase cannot read the data, as the operation which was suspended is not resumed as shown in Fig. 7.

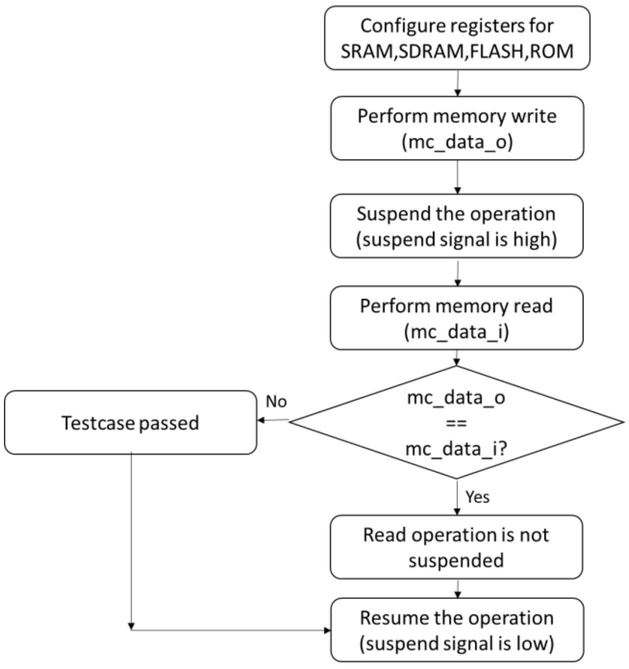

Configure registers for SRAM,SDRAM,FLASH,ROM
Perform memory write (mc_data_o)
Suspend the operation (suspend signal is high)
Perform memory read (mc_data_i)
mc_data_o == mc_data_i?
No
Testcase passed
Yes
Read operation is not suspended
Resume the operation (suspend signal is low)

Fig. 7.Suspend and Resume testcase

V. SOFTWARE

Questasim 10.0b tool is used for the simulation of the project. The Questa® Verification Solution converts verification, intensely growing verification efficiency and handling resources further proficiently constructed on numerous pre- vailing technologies and strongly associated with Veloce® emulation Questa answers the difficulties of increasing complexity of SoCs [11].

VI. RESULTS

The Fig. 8, shows the waveforms obtained when there is write-read operation done to all the memory modules connected to the chip selects. In the above testcases, all the chip selects were connected to the same type of memories. Fig. 8, shows the write and read operation of a memory after completion of write cycles when all the chip selects are connected to different type of memories. From the figures, we can infer that there is no data corruption since the data written (mc_data_o) and the data read (mc_data_i) from a specific address match.

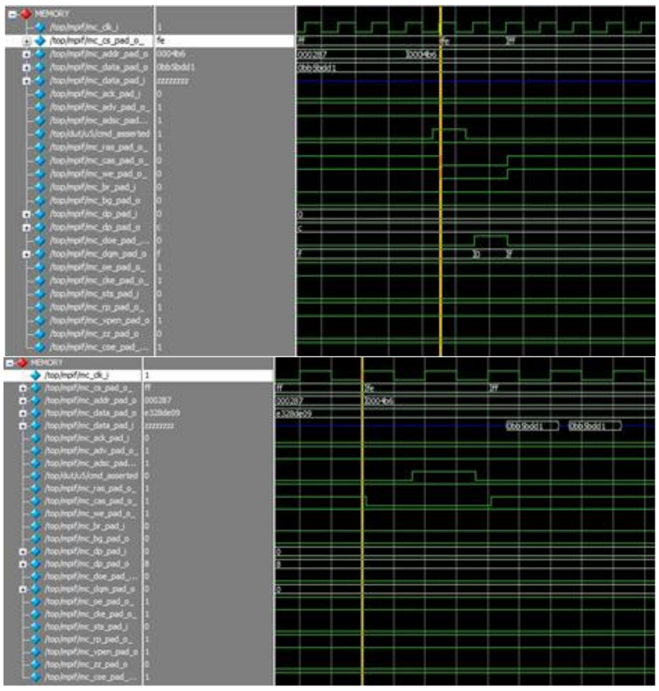
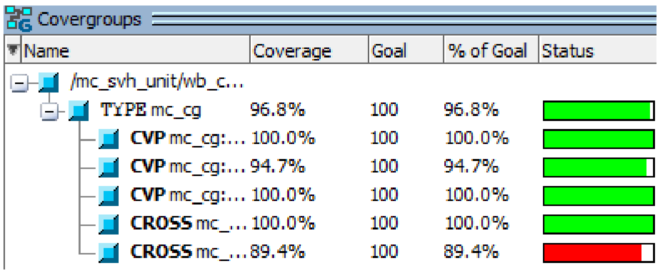


Fig. 8.Ten continuous Write and Read transactions when all the memories are connected to the chip selects of the controller

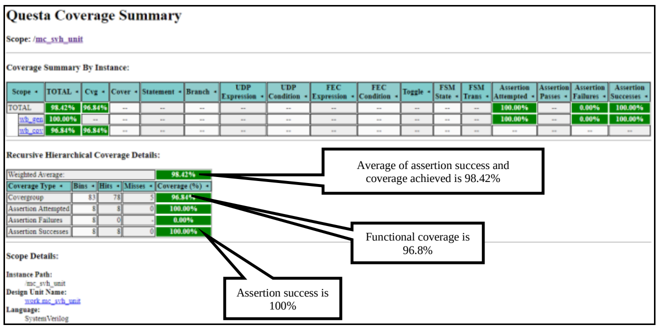


Covergroups

Name	Coverage	Goal	% of Goal	Status
/mc_svh_unit/wb_c...				
TYPE mc_cg	96.8%	100	96.8%	🟩
CVP mc_cg:...	100.0%	100	100.0%	🟩
CVP mc_cg:...	94.7%	100	94.7%	🟩
CVP mc_cg:...	100.0%	100	100.0%	🟩
CROSS mc_...	100.0%	100	100.0%	🟩
CROSS mc_...	89.4%	100	89.4%	🟥

Fig. 9.Functional Coverage achieved using covergroups

The functional coverage analysis of the environment is shown in the Fig. 9, where there are bins written for each of the registers, operations and the chip selects. Assertions are also implemented as a part of the environment and the coverage and assertion report are obtained with the help of the tool as shown in Fig. 10. The assertion success got by the html report is 100%, indicating there is no assertion failure.



Questa Coverage Summary

Scope: /mc_svh_unit

Coverage Summary By Instance:

Scope	TOTAL	Cvg	Cover	Statement	Branch	UDP Expression	UDP Condition	FEC Expression	FEC Condition	Toggle	FSM State	FSM Trans	Assertion Attempted	Assertion Passes	Assertion Failures	Assertion Successes
TOTAL	98.42%	96.84%	--	--	--	--	--	--	--	--	--	--	100.00%	--	0.00%	100.00%
wb_gen	100.00%	--	--	--	--	--	--	--	--	--	--	--	100.00%	--	0.00%	100.00%
wb_cov	96.84%	96.84%	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Recursive Hierarchical Coverage Details:

Weighted Average: 98.42%

Coverage Type	Bins	Hits	Misses	Coverage (%)
Covergroup	83	78	5	96.84%
Assertion Attempted	8	8	0	100.00%
Assertion Failures	8	0	-	0.00%
Assertion Successes	8	8	0	100.00%

Scope Details:

Instance Path:
/mc_svh_unit
Design Unit Name:
work.mc_svh_unit
Language:
SystemVerilog

Average of assertion success and coverage achieved is 98.42%
Functional coverage is 96.8%
Assertion success is 100%

Fig. 10.Coverage and assertion report

The functional coverage achieved for this design is 96.8%. It is because of the reserved fields in the register, which is according to the design. Also, there is power on configuration register, which is configured once in the design & is write protected. Thus, it is not 100%. Also, in the functional coverage Fig. 9, CP_REG (coverpoint for register) & CP_REG_X_WR_RD (coverpoint for register crossed with write-read signal) are the only 2 cover points not covered 100%. This justifies the reason behind the coverage not being 100%.

VII. CONCLUSION

The memory controller architecture supports SRAM, SDRAM, FLASH, ROM and any synchronous or asynchronous memory devices. A re-scalable, re-configurable and re-usable environment which is a coverage driven constraint random-based model are thus presented to verify the functions of a memory controller. Questa Sim 10.0b tool is used for simulation. The verification environment built in this work, gives a functional coverage of 96.8% and assertion success of 100% with 0% assertion failures. This verification method for memory controller is proved to be more time-efficient than the directed test method, which is time consuming.

REFERENCES

1. 2nd Generation Intel® Core™ Processor Family Desktop, Intel® Pentium® Processor Family Desktop, and *Intel® Celeron® Processor Family Desktop* Datasheet, Volume 1, June 2013.
2. Rudolf Usselmann, "Memory Controller IP Core", *OpenCores Revision.1.7 January 21, 2002.*
3. Barbara Johnson, "MSC711x Memory Controller Usage Guidelines Supporting Double Data Rate (DDR) SDRAM Devices", *NXP, Revision. 1, 3/2007*
4. L. Molnar and A. Gontean, "Functional simulation methodes," in *2016 12th IEEE International Symposium on Electronics and Telecommunications (ISETC),* IEEE, pp. 198–202, 2016.
5. M. D. Marino and K. Li, "RAMON: Region-Aware Memory Controller," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 4, pp. 697-710, April 2018.
6. D. Sahoo, S. Sha, M. Satpathy, M. Mutyam, S. Ramesh and P. Roop, "Formal Modeling and Verification of Controllers for a Family of DRAM Caches," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2485-2496, Nov. 2018.
7. C. Yu, W. Brown, and M. Ciesielski, "Verification of arithmetic datapath designs using word-level approach—a case study," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS),* IEEE, pp. 1862–1865, 2015.

8. P. Gurha and R. Khandelwal, "SystemVerilog assertion based verification of amba ahb," in *2016 International Conference on Micro-Electronics and Telecommunication Engineering (ICMETE),* IEEE, pp. 641–645, 2016.
9. K. Khalifa, H. Fawzy, S. El-Ashry, K. Salah, "Memory controller architectures: A comparative study", *8th IEEE Design and Test Symposium*, Dec. 2013.
10. V. Jusas and T. Neverdauskas, "Stimuli generator for testing processes in vhdl," in *2014 NORCHIP*, IEEE, pp. 1–4, 2014.
11. J. Papanuskas, "IEEE vhdl 1076.1: Mixed-signal behavioral modeling and verification in view of automotive applications," in *Proceedings VHDL International Users' Forum. Fall Conference,* IEEE, pp. 252–257, 1997.