Constructing Effective UVM testbench for DRAM Memory Controllers

Khaled Salah[1], Hassan Mostafa[2]
[1]Mentor, a Siemens Business, Egypt. [2] Electronics and Communications Engineering Department, Cairo University, Giza 12613, Egypt.
khaled_mohamed@mentor.com
hmostafa@uwaterloo.ca

Abstract—In this paper, a general verification architecture for DRAM memory controllers is proposed. The proposed verification architecture is based on universal verification methodology (UVM) which makes use of the common features between different DRAM memory controllers to generate common and configurable scoreboard, sequences, stimulus, different UVM components, payload and test-cases. The proposed verification architecture uses minimum number of macros, methods and classes. The proposed verification architecture provides high reusability for UVM tests.

Index Terms — UVM, DRAM, Verification, Architecture.

I. INTRODUCTION

As verification is now considered as the bottleneck of any complex VLSI design. So, improving the verification efficiency is a must. Due to the exponential growth in design complexity, verification is facing a new level of challenges. Mainly, there are two levels of verification, IP-level verification and SoC-level verification. For IP-level verification, we need to verify the functionality. For SoC-level verification, we need to verify the connectivity. The verification cycle is mainly driven by time to market. A host controller is interested in correct and successful communication with the device, high data through-put and in saving power.

Memory controllers are considered a vital component in many VLSI designs. They provide an efficient interface between the memory cores and the host in terms of maximizing transfer speed, processing data, ensuring data integrity. Memory controllers can be flash-based or DRAM-based. DRAM memory controllers' examples are DDRx, LPDDRx, HMC, HMB, and WIDEIO [1]-[6]. The architecture of these different controllers has many common features. TABLE I shows the main memory controllers features. TABLE II presents comparison between the most common architecture in terms of commands. TABLE III compares between different memories controllers.

Enhancing the verification environment is a challenge for DRAM memory controllers as they are very time consuming parts.

In this paper, generic UVM-based verification architecture is proposed to verify the DRAM memory controllers. The proposed architecture exploits the common features to generate common UVM components and tests. The rest of paper is organized as follows. In Section II, The proposed architecture is introduced. In Section III, results are analyzed. Conclusions are given in section IV.

TABLE I
MEMORY CONTROLLERS FEATURES

Features	Explanation
Topology	Point to point, or multi-master/ multi-slave.
Physical interface	Pins.
Initialization process	Start operation and negotiation.
Command Sets	• Read • Write • multiple read • multiple write • Activate • Refresh
Responses	Types and size.
Internal registers	Information about the memory controllers.
Data rate	DDR/ SDR.
Timing	The time between different commands, responses, and data.
Reliability	CRC/ECC.
Performance	In terms of clock frequency.

TABLE II
COMPARISON BETWEEN THE MOST COMMON ARCHITECTURE IN TERMS OF COMMANDS

Features	3D		2D	
	HMC	WideIO	LPDDRx	DDRx
Read		✓	✓	✓
Write	✓	✓	✓	✓
Command queuing	✓	✓	✓	✓
Retry	✓			
Power management	✓	✓	✓	✓
Sleep			✓	
Power down	✓	✓	✓	✓
Deep power down			✓	
self-refresh	✓	✓	✓	✓

TABLE III
COMPARISON BETWEEN DIFFERENT DRAM MEMORY CONTROLLERS

	HMC	WIDEIO	LPDDRx	DDRx
Number of banks	8/16 Banks	4 Banks	16 banks	8 banks
Technology	3D	3D	2D	2D
Memory Cells	DRAM	DRAM	DRAM	DRAM
# Memory partitions	1 partition	1 partition	1 partition	1 partition
Modes of operations	• Initialization • Sleep • Active • Power Down	• Idle • Active • Power Down • Deep Power Down	• Idle • Active • Power	• Idle • Active • Power
Data Integrity	CRC	ECC	CRC	CRC
Number of Registers	15	8	64	64
Size of registers (bits)	32	19	8	8
Number Of Pins	29	48	12	12
Transmission Type	Synchronous	Synchronous	Synchronous	Synchronous
Number of commands	64	32	20	20
Command length (bits)	48	4	6	6
# Responses	5	1	0	0
Command/Data Bus share same bus	No	No	No	No
Interface pins	• CLK • Reset • CMD Bus • Data Bus	• CLK • Command Bus • Address Bus • Data Bus • Data Mask • Reset	• CLK • Command Bus • Address Bus • Data Bus • Reset	• CLK • Command Bus • Address Bus • Data Bus • Reset
Interface Type	Parallel	Parallel	Parallel	Parallel
Booting	Optional	-	-	-
Clock(MHz)	200	200	250	250
Speed (MB/s)	200	200	100	100
Data Rate	SDR/DDR	SDR	DDR	DDR
# Timing Modes	1	1	Many	Many
Topology	Point to point	Point to point	Point to point	Point to point

II. GENERIC UVM-BASED VERIFICATION ARCHITECTURE

Mainly, there are two categories of memories: hard disk driver (HDD) and solid-state driver (SSD) which uses volatile memory and nonvolatile memory to store data.

The proposed verification architecture makes use of the common features between different DRAM memory controllers to generate common and configurable scoreboard, sequences, stimulus, different UVM components, payload and test-cases.

The proposed generic UVM-based verification architecture for DRAM-based memory controllers is shown in Fig. 1. A generic scenario is shown in Fig. 2.

UVM verification environment is composed of different components such as stimulus, scoreboard, driver, coverage, sequencer and monitor [7]-[23]. All these components can be highly reused [24].

The sequencer generates the data and the driver sends it to the DUT. The scoreboard compares the received data or response against the expected one. The monitor samples the data and responses.

The proposed testbench overcomes many verification challenges such as covering different levels of communication, covering different parameters, generating efficient random stimulus generation, coping with fast protocols evolution and revisions. Moreover, it provides a scoreboard that not only ensures transaction-level data correctness, but also automates common verification tasks, such as transaction order check [25]. Also, the scoreboard is generic enough to allow in order checking and out of order checking. Moreover, it supports UVM callback. UVM callbacks are used to add the new capabilities, without creating a huge OOP hierarchy.

III. RESULTS AND DISCUSSIONS

HMC, HBM, WIDE-IO, LPDDRx, and DDRx. Using the proposed generic verification architecture, a fast coverage closure is obtained as with each new protocol, we use the previous verification and add new scenarios that will be useful for the next protocol and so on.

For HMC, basic sequence generation is compared against the proposed methodology. Results show that the proposed verification exhibits a reduced simulation time as shown in Fig. 3.

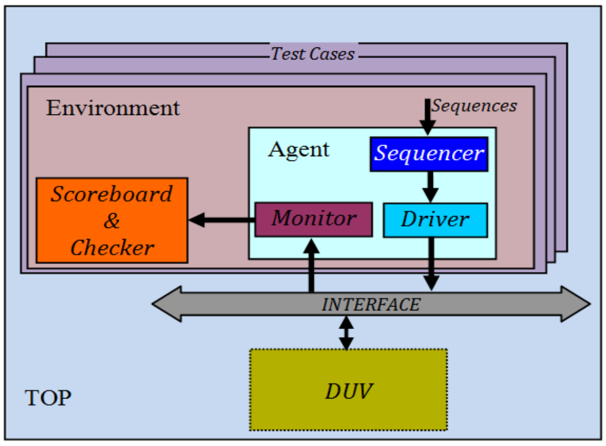

Fig. 1 The generic UVM architecture for DRAM memory controllers.

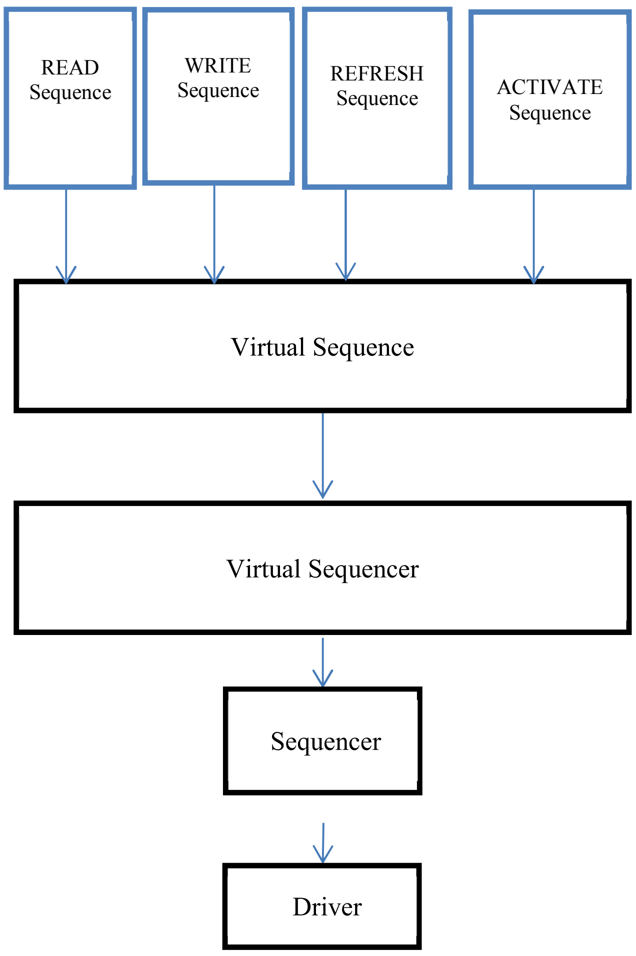

Fig. 2 A generic scenario.

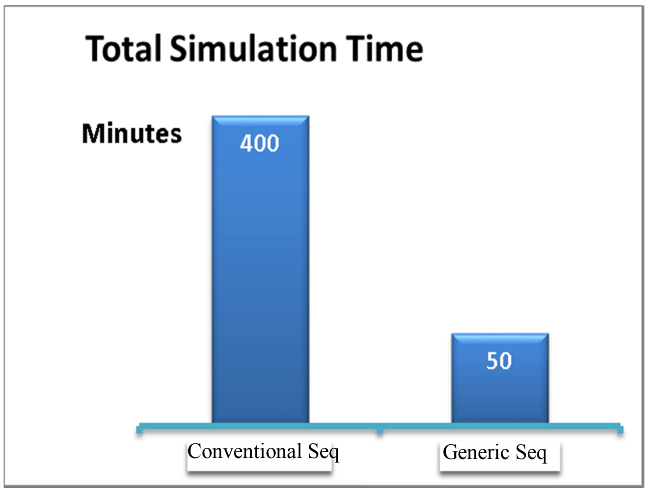

Fig. 3 The conventional tests generation method versus and the proposed method.

IV. CONCLUSIONS

In this paper, generalized UVM-based verification architecture for DRAM memory controllers is proposed. The proposed architecture uses the most common features between different DRAM memories to generate different configurable test scenarios. The final results show that the proposed architecture speedup simulation time 8x compared to conventional test generation methods.

REFERENCES

[1] DDR5 SDRAM Standard, JEDEC Standard.
[2] LPDDR5 SDRAM Standard, JEDEC Standard.
[3] Graphics Double Data Rate (GDDR6) SGRAM Standard, JEDEC Standard.
[4] High Bandwidth Memory (HBM) DRAM, JEDEC Standard.
[5] About Hybrid Memory Cube, Hybrid Memory Cube Consortium: http://hybridmemorycube.org/ technology.html.
[6] WIDE I/O Technical Report Revision 2.0, JEDEC Standard.
[7] J. Bromley "If SystemVerilog Is So Good, Why Do We Need the UVM? Sharing Responsibilities between Libraries and the Core Language" FDL, 2013.
[8] M.F. S. Oliveira, F. Haedicke, R. Drechsler, C. Kuznik, H.M. Le, W. Ecker, W. Mueller, D. Große, V. Esen "The System Verification Methodology for Advanced TLM Verification " ISSS, 2012.
[9] H. Zhaohui, A. PIERRES,H. Shiqing, C. Fang, P. ROYANNEZ, E. P. SEE, Y. L. HOON "Practical and Efficient SOC Verification Flow by Reusing IP Testcase and Testbench" ISOCC, 2012.
[10] S. Raghuvanshi, V. Singh "Review on Universal Verification Methodology (UVM) Concepts for Functional Verification" International Journal of Electrical, Electronics and Data Communication, ISSN: 2320-2084 Volume-2, Issue-3, March-2014.
[11] Y. Yun, "Beyond UVM for practical SoC verification", SoC Design Conference (ISOCC), pp158-162, 2011.

[12] S. Sutherland, D. Mills “Synthesizing SystemVerilog: busting the myth that SystemVerilog is only for verification”, SNUG Silicon Valley 2013.

[13] C. Spear and G. Tumbush, “SystemVerilog for Verification (2nd Edition)”, A Guide to Learning the Testbench Language Features, Springer, LLC, (2012).

[14] H. Sohofi, Z. Navabi “Assertion-Based Verification for System-Level Designs” ISQED, 2014.

[15] B. Vaidya N. Pithadiya “An Introduction to Universal Verification Methodology” journal of information, knowledge and research in electronics and communication engineering, volume – 02, ISSUE – 02, 2013.

[16] S. Sutherland and D. Mills “Can My Synthesis Compiler Do That? What ASIC and FPGA Synthesis Compilers Support in the SystemVerilog-2012 Standard" DVCon-2014, San Jose, CA.

[17] Y. Jin “System-Level Verification Platform using SystemVerilog Layered Testbench & SystemC OOP” International Journal of Control and Automation Vol.7, No.2 (2014), pp.221-230.

[18] Khaled Salah, Mohamed AbdelSalam. "Smart autocorrection methodology using assertions and dynamic partial reconfiguration", 2017 24th IEEE International Conference on Electronics, Circuits and Systems (ICECS), 2017.

[19] Accellera, ―Universal Verification Methodology (UVM) 1.1 User’s Guide, CA, May 2011.

[20] T. Vörtler, T. Klotz “Enriching UVM in SystemC with AMS extensions for randomization ad functional coverage”, DVCON Europe), October 2014.

[21] www.opencores.org.

[22] Bruce Wile, John C. Goss, Wolfgang Roesne “Comprehensive Functional Verification the Complete Industry Cycle” ELSEIVER, 2005.

[23] K. Salah “A Unified UVM Architecture for Flash-Based Memory”18th International Workshop on Microprocessor and SOC Test and Verification (MTV).

[24] K. Salah “A uvm-based smart functional verification platform: Concepts, pros, cons, and opportunities” IDT, 2014.

[25] B. Vaidya, N. Pithadiya “An Introduction to Universal Verification Methodology” Journal of Information, Knowledge and Research in Electronics and Communication Engineering, 2013.