International Journal of Electronics and Communication Engineering (IJECE)
ISSN(P): 2278-9901; ISSN(E): 2278-991X
Vol. 8, Issue 3, Apr - May 2019; 1-8


International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

# UVM BASED VERIFICATION OF DUAL PORT SRAM BY IMPLEMENTING BIST

***Sanket S. Nimbalkar[1] & R. Anitha[2]***

*Research Scholar, Department of Micro and Nanoelectronics, Vit University, Vellore, Tamil Nadu, India*

## *ABSTRACT*

*In this paper, we present a coverage driven functional verification based on UVM methodology using system Verilog language. As transistor scaled down, memory area on-chip & density of memory drastically increases. Which leads to an exponential increase in the problem of faults in memory? So, we present Built-in self-test which is used to detect and repair the faults in multi-ported memory. This BIST is design based on DFT concepts which use MARCH C test algorithm. We used layered UVM test bench architecture which allows engineer work independently and simplifies verification process by code re-usability feature.*

***KEYWORDS:*** *UVM (Universal Verification Methodology [9]), BIST (Built-In-Self-Test), Dual Port SRAM, Code Re-Usability, MARCH C*

***Article History***

***Received:*** *11 Mar 2019* **|** ***Revised:*** *23 Mar 2019* **|** ***Accepted:*** *01 Apr 2019*

## INTRODUCTION & BACKGROUND

With the scaling of technology to smaller size transistors, the possibility of manufacturing defects has increased. Nowadays most of the semiconductor manufacturers are rated based on their memory technology. So, the demand for memory is growing such that more and more area on the chip is acquired by the memory arrays. But it increases the probability of an increase in manufacturing faults. Due to the tight packaging of transistors for memory design, the probability of faults in-memory structures is more, compared to the logic structure in any chip.

DFT (Design for Testability) is always been the least concentrated field in VLSI (Very Large-Scale Integration). Compare to testing a Logic design, memory testing is difficult. Unlike logic testing, memory testing is more complicated due to its defect adaptive testing nature. Memory structures can't be represented in their gate level structures because of the large size of memory and the number of FF (flip-flop) required to model them. The memory structures are designed mostly in layered deeper on the chip. So insufficient test access leads to a noticeable problem in the memory array. As memory is structurally designed, observable and controllable at the array level, the test engineer can develop a test algorithm to identify the faults in the memory array.

There are different technologies for designing memory. Design constraints of memory such as area, power & speed decide which technology should use for designing memory. For each type of design technology, the potential fault model changes as per the physical structure of memory. By considering fault models test engineer creates a test plan. Which is support chip from inside like BIST or support from the external structure? BIST needed extra hardware resources to be inbuilt inside the chip. Also, there should not be any violations regarding power, area, cost, and speed. Testing time also plays an important role in considering an overall delay of the chip. So, test engineer should try to cover most of the

faults in given time complexity. Verification of functional correctness before the sign-off is more important than validating after manufacturing. So, verification is more time taking step (around 70%) in the chip manufacturing cycle. Design engineer needs more synthesis friendly language while verification engineer needs more simulation friendly language. So, for designing and verifying purpose System Verilog is introduced (standardized as IEEE 1800).

System Verilog is hardware description & hardware verification language used to model, design, simulate, test and implementation. It includes many features such as assertion, coverage to the verification environment. To improve the reusability and simplicity we use Universal Verification Methodology (UVM) which is the most widely used methodology. UVM ensures that the test bench is layered & separated into individual files. It will speed the verification process & helps in reusability.

In the given paperwork, we used a dual port RAM module as the DUT. Also, we design BIST for designed DUT model. After that, this model is verified with the help of UVM methodology.

## BIST ARCHITECTURE

Digital systems are tested & diagnosed many times in their life spam but this testing should be quick and have high fault coverage. A software implementation of the test at the highest level of the system gives flexibility but they have some of the disadvantages as poor coverage resolution, slow, long and expensive. So, it is better to build a self-test function in hardware. which gives benefits in term of reliability and the reduced maintenance cost, reduced test generation effort at all levels, reduced test effort at chip through system levels, improved system-level maintenance and repair, and improved component repair.

As we discuss earlier dual port memory is difficult to test compare to single port memory. Some of the marching algorithms are used to test ports separately but they are insufficient to check the compound coupling faults. Thus, we use such marching algorithm which can test concurrent operations. But it should know the limitation of memory operations. Basic BIST is trying to synchronize both clocks which can avoid clock crossing coupling. This can be achieved by using a multiplexer.

The BIST hardware contain the following blocks,

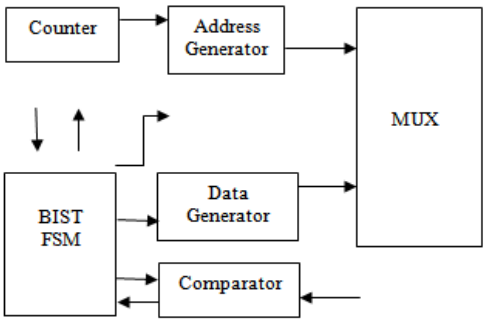


**Figure 1: BIST Architecture**

- **Counter**

  It used to change the address for marching, increment or decrements. Control signals for counter are passes from FSM of BIST.

- **Address Generator**

  This chooses the exact location of pins of individual pins.

- **Comparator**

  Compare the output of the data out port of the ram module with the compare word generated by the state machine and return the result to the state machine.

- **Multiplexer**

  It chooses between inputs of RAM & BIST nets which are connected to RAM. Select pins are from the BIST control signal.

- **BIST FSM**

  The main control part of BIST. Depends on its states BIST units perform the task. Is also control the RAM control signal. A state transition is depending on error value generated by counter and comparator block.

## MARCHING ALGORITHM

In this marching algorithm, there are 23 states which ensure that each port must see Read 0 Write 1 followed by Read 1 Write 0 and vice versa for the whole address range. It marched through each half of the address through two different ports. So, it takes half completion time. In this address, selector bus help us to avoid write collision.

In the 1$^{st}$ state, it writes default constant word through both ports. Next state will be selected after reaching the max half address. In that state, it reads the value from all addresses and compare it with constant word and then writes another value in all ports. This process is repeated till fault detected and then states are the move to Error state where they report the error. After all this operation memory is reset to zero for easier operation of the memory. After marching all the states BIST report success to the external interface. A state diagram for the given marching algorithm is shown in the following diagram.

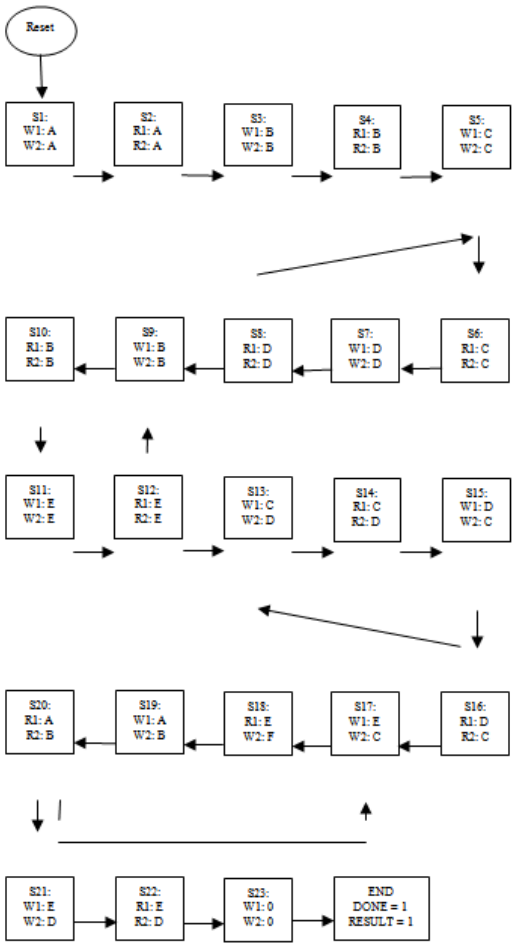


**Figure 2: Bist State Machine (Marching Algorithm)**

ERROR STATE: DONE= 1, RESULT =0

## UVM ARCHITECTURE

UVM is systematized methodology which is explicitly based on System Verilog language used for verifying basic and complex IC (integrated circuit). UVM is derived from OVM (Open Verification Methodology). With the use of UVM, we can build modular reusable verification components and test-benches. There are so many advantages of UVM over other verification languages like OVM, VMM, etc.

UVM gives us good control of the generation of stimulus. Sequence methodology gives us several ways to generates stimulus. Which includes virtual sequences, randomization, layered sequences. This increases the stimulus generation capability of UVM. UVM separates actual test bench hierarchy from the tests in term of stimulus. This gives reusability of stimulus across the whole project. In UVM, we can modify components easily by using a factory mechanism. It enables an override of components in any tests or environment without changing any of its base code. Configuration mechanism in UVM helps in configuring different components of test benches at any level in any environment without worrying about its hierarchy.

UVM phases are synchronized mechanism for the environment which represented as a callback method. There are different phases in UVM: build phase, connect phase, run phase, extract phase, check phase, report phase, end of the

elaboration phase, a start of simulation phase. The class which is derived from uvm_component should use some of the phases or all the phases and executes them in particular order. One more advantage of UVM is it provides Transition Level Modeling (TLM). This is a modeling technique which is used to design abstract models of components & systems. In this data it represents in the form of transactions which are flows in and out of components through special ports which called a TLM interface. UVM gives a set of different types of communication interfaces that are used tocommunicates between components such that data can be transfer in the form of packets.

UVM provides modularity in its test bench. Means in the UVM test bench there are different modular components like Driver, Sequencer, Monitor, etc. This enables the reuse of components at the unit level or chip level. It increases reusability of code & simplifies its execution.

In the following diagram, we gave the basic UVM architecture which we used in the given project.

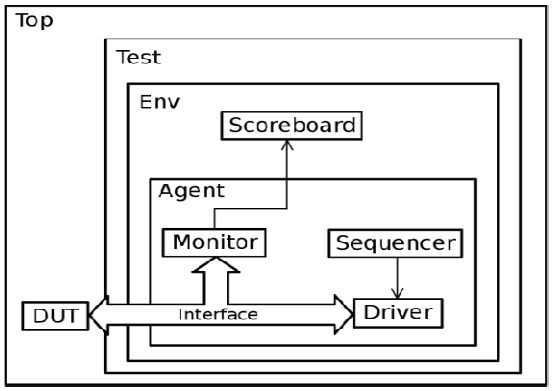


**Figure 3: UVM Testbench Architecture**

- **Driver and Monitor**

  This to components is mainly used for communicating with the DUT unit. Where Driver initiate the signal, which drives the inputs and Monitor is used to monitor the outputs.

- **Sequencer**

  This module receives simulated data and sends information to the driver block.

- **Agent**

  This block contains a monitor, sequencer and driver block in it. It provides a number of transition packets to driver and monitor.

- **Environment**

  This block instantiates all other modules in the build phase and connects them all in the connect phase.

- **Scoreboard**

  This block receives the current behavior from monitor block and it compares it with predicted outputs.

- **DUT**

  In our case, DUT (Design Under Test) is one dual two-port memory with 32-word size. It is synchronous dual port SRAM.

In this, all inputs are clocked with respect to the corresponding port's clock. It has a 7-bit address line which supports 128 locations.

## RESULTS

The dual port memory was successfully tested & verified using Built-in self-test and UVM methodology with the help of System Verilog language.

The DUT which id dual port ram's read-write output is shown in below fig. When chip select input & write enable pin for the specific port is high then write operation is carried out at that port. At high chip select pin if the write enables pin is low & output enable pin is high then reading operation is carried out.

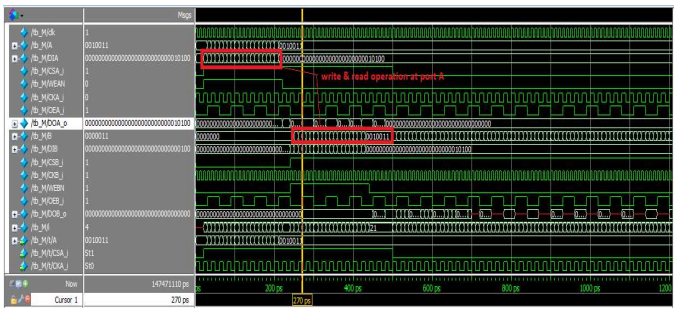


**Figure 4: RAM (DUT) Output**

BIST architecture is tested using normal Verilog test-benches. Testbenches are runs until done signal is enabled. The simulation is shown in the following fig.

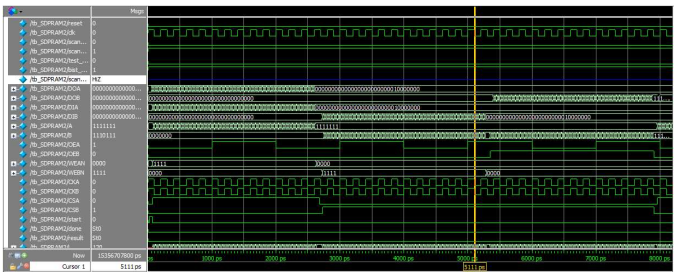

**Figure 5: BIST Output**

UVM testbench was designed using different components of UVM. It was a layered test-bench structure which used to get randomized stimulus. A result of UVM test-benches is shown in given fig.

**Figure 6: Test-Bench (UVM) Output**

Coverage of UVM testbench is shown in the following figure which is around 98%. Coverage report shows

separate coverages of each cover points.

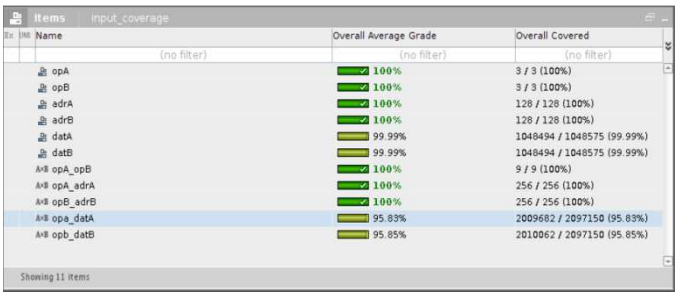


**Figure 7: Coverage**

## CONCLUSIONS

In this work, we introduced BIST architecture for synchronous dual port SRAM. This BIST architecture has 23 states which help to implement Marching pattern. Also, we did block level verification of dual port memory by using UVM methodology whose main advantages are code reuse, design test for design check and measuring functional coverage. UVM & System Verilog library can randomize data and operation of BIST. Which helps to ensure that memory verifies for each address and for different data. Also, we used half address method which reduced our verification time by reading and writing from both ports.

## REFERENCES

1. *M. L. Bushnell and V. D Agarwal, "Essentials of Electronic Testing" Kluwer academic Publishers, Norwell, MA, 2000.*
2. *Giuseppe Visalli, "UVM-based Verification of ECC Module for Flash Memories" 978-1-5386-3974-0/17/$31.00 c 2017 IEEE.*
3. *K. Khalifa and K. Salah, "Implementation and verification of a generic universal memory controller based on UVM," in 2015 10th International Conference on Design Technology of Integrated Systems in Nanoscale Era (DTIS), April 2015, pp. 1–2.*
4. *G. Zhong, J. Zhou, and B. Xia, "Parameter and UVM, making a layered testbench powerful," in 2013 IEEE 10th International Conference on ASIC, Oct 2013, pp. 1–4.*
5. *Y. Cai, G. Yalcin, O. Mutlu, E. F. Haratsch, A. Cristal, O. S. Unsal, and K. Mai, "Error analysis and retention-aware error management for NAND flash memory," Intel Technology Journal, vol. 17, no. 1, 2013.*
6. *Khaled Khalifa. "Extendable Generic Base Verification Architecture for Flash Memory Controllers Based on UVM," Proceedings of the 2017 IEEE 21st International Conference on Computer Supported Cooperative Work in Design, 2017 IEEE.*
7. *Khaled Salah, "A Unified UVM Architecture for Flash-Based Memory", 2017 18th International Workshop on Microprocessor and SOC Test and Verification, 2018 IEEE.*
8. *K. Salah, "A UVM-based smart functional verification platform: Concepts, pros, cons, and opportunities." Design & Test Symposium (IDT), 2014 9th International. IEEE, 2014.*

9. *Accellera, Universal Verification Methodology (UVM) 1.1 User's Guide, CA, May 2011.*

10. *Palakeeti Naveen Kalyan & K Jaya Swaroop, Verification of AMBA-AHB Based Verifying IP Using UVM Methodology, International Journal of Electronics, Communication & Instrumentation Engineering Research and Development (IECIERD), Volume 5, Issue 4, July-August 2015, pp. 21-28*

11. *H. Yokoyama, H. Tamamoto, and X. Wen, "Built-in random testing for dual-port rams," in Proceedings of IEEE International Workshop on Memory Technology, Design, and Test, Aug 1994, pp. 2–6.*

12. *Z. Zhang, Z. Wen, and L. Chen, "BIST approach for testing embedded memory blocks in system-on-chips," in 2009 IEEE Circuits and Systems International Conference on Testing and Diagnosis, April 2009, pp. 1–3.*

13. *C.-F. Lin and Y.-J. Chang, "An area-efficient design for programmable memory built-in self-test," in 2008 IEEE International Symposium on VLSI Design, Automation and Test (VLSI-DAT), April 2008, pp. 17–20.*