Vol 2 No 2 (2020) *e*-ISSN: 2682-8383

Journal of Engineering Technology and Applied Physics

VHDL Modelling of Low-Cost Memory Fault Detection Tester

Quek Wei Chun, Pang Wai Leong*, Chan Kah Yoong, Lee It Ee and Chung Gwo Chin
Faculty of Engineering, Multimedia University, Cyberjaya, Malaysia.
*wlpang@mmu.edu.my
https://doi.org/10.33093/jetap.2020.2.2.3

Abstract **– Memory modules are widely used in varies kind of electronics system design. The capacity of the memory modules has increased rapidly since the past few years in order to satisfy the high demand from the end-users. The memory modules' manufacturers demand more units of automatic test equipment (ATE) to increase the production rate. However, the existing ATE used in the industry to carry out the memory testing is too costly (at least a million dollars per ATE tester). The low-cost memory testers are urgently needed to increase the production rate of the memory module. This has inspired us to design a low-cost memory tester. A low-cost memory fault detection tester with all the major fault detection algorithms that used in industry is modelled using Very High Speed Integrated Circuit Hardware Description Language (VHDL) in this paper to support the need of the low-cost ATE memory tester. The fault detection algorithms modelled are MATS+ (Modified Algorithm Test Sequence), MATS++, March C, March C-, March X, March Y, zero-one and checkerboard scan tests. PERL program is used to analyse the simulation results and a log file will be generated at the end of the memory test. Extensive simulation and experimental test results show that the memory tester modelled covers all the memory test algorithms used in the industry. The low-cost memory fault detection tester designed provides the 100 % fault detection coverage for all memory defects.**

Keywords—VHDL, Memory tester, March test

I. Introduction

Memory is one of the crucial modules used in most of the electronic systems. Memories have dominated the chip area and the area consumed increases linearly throughout the year. Memories are one of the sensitive parts with high potential of defects. The size of the memory cell is shrinking and the memory cells are built closer to each other in order to increase the density of the memory circuitry. The memory module with high density and complexity is fault-prone. A lot of works were carried out to examine the memory defects [1-5], but none of the work is to design low-cost memory tester.

There are various memory test algorithms were introduced to test the memory devices [6, 10-12]. The conventional test algorithms are zero-one and checkboard scan tests. The conventional test algorithms are widely used to test memory modules. New fault detection algorithms are proposed to detect memory faults, which are single-cell fault and two cells fault models. Some of the popular fault detection algorithms are Modified Algorithmic Test Sequence (MATS), MATS+, MATS++, Marching-1/0, March A, March B, March C, March C-, Extended March C-, March G, March X and March Y [10].

There are many types of memory testers available in the market but the tester price is too costly for the small and medium industry operators. This has inspired us to design a cost-effective memory tester. Various design techniques such as the full custom, ASIC, and system modelling can be used to design the memory tester. In this paper, VHDL is used to model the memory tester. VHDL modelled the behaviour of the memory test algorithms and the design is realizable through the Field Programmable Gate Array (FPGA) [7]. The designer has high flexibility in designing the memory testing algorithms and FPGA is a reprogrammable device. It is a promising solution to design an effective low-cost memory tester.

MMU MULTIMEDIA UNIVERSITY
This work is licensed under the Creative Commons BY-NC-ND 4.0 International License.
Published by MMU PRESS. URL: https://journals.mmupress.com/index.php/jetap/index

The remaining of this paper is organized as follows. A literature review is illustrated in section 2. The methodology is illustrated in section 3. Extensive simulations are discussed in section 4. Finally, a conclusion is drawn in section 5.

II. Memory Fault Detection Algorithms

Various fault detection algorithms are introduced to detect the faults that may occur in the memory devices. The faults in the memory cells can be divided into two types, i.e. a single cell and two cells fault models. For single-cell fault model, it consists of Stuck-At Fault (SAF), Transition Fault (TF), and Data Retention Fault (DRF) [10-12].

SAF contributes up to 50 % of the memory's faults. The data of the memory may be stuck at zero (SA0) or stuck at one fault (SA1). TF fault occurs when the memory cell failed to update the data stored in the memory cell either from 1 to 0 or 0 to 1. TF may occur once the memory cell encounters either SA0 or SA1 that prohibits the logic 1 or logic 0 transitions respectively. DRF occurs when the memory cell fails to retain the data stored in it. The data in the memory cell with DRF may change from 0 to 1 or 1 to 0 after a delay time.

Two cells fault model is mainly contributed by the Coupling Fault (CF). CF model involves two memory cells, i.e. aggressor cell (a-cell) and victim cell (v-cell). A change of the data stored in a-cell forced the change of the data stored in v-cell. The data in v-cell may be changed from 0 to 1 or 1 to 0.

There are four types of CF models, i.e. 1) Fault type Idempotent CF (CFid), 2) State CF (CFst), 3) Inversion CF (CFin) and 4) K-Coupling Fault (k-CF). CFid occurs when a write operation applied to the a-cell and it triggered a logic flip on the data stored in v-cell. CFst occurs when the data stored in a-cell and it caused the data stored in v-cell stuck at either 0 or 1. CFin occurs when writing data to a-cell and it toggles the data stored in v-cell. The k-CF also called a pattern sensitive fault. The k-CF fault only occurs when the v-cell is triggered by the surrounding cells under a certain kind of data pattern.

Another common fault on the memory module is Address Decoder Fault (AF). AF occurs when the memory array assigned the wrong memory address. AF can be divided into four categories, i.e. 1) no memory cell assigned to a memory address, 2) multiple cells assigned to a memory address, 3) no address assigned to a memory cell and 4) multiple addresses assigned to a memory cell.

March algorithm is the most popular memory test algorithm used in the industry to detect the memory faults discussed. The March algorithm will test the memory cell one by one. The March test will only begin to test the next memory cell once the current memory cell is completed the test. March test is modelled in this paper to test the high-density memory modules. Table I shows the fault coverage of the March algorithms.

The memory test algorithm is begun by writing a value to the memory cell located at an address in the memory array. Followed by, reading back the data stored in the memory cell. If the data value read and the data value written to the memory cell are the same, then this memory cell passes the test. If the values differ, then this memory cell fails the memory test. The memory fault coverage and the total test time required for each algorithm are different. Table II summarized the memory test notations used in the memory test algorithms.

Table I. Fault coverage and complexity of various March test algorithms.

Algorithm	AF	SAF	TF	CF	Complexity
MATS	✕	✓	✕	✕	4n
MATS+	✓	✓	✕	✕	5n
MATS++	✓	✓	✓	✕	6n
Marching-1/0	✓	✓	✓	✕	14n
March A	✓	✓	✓	✓	15n
March B	✓	✓	✓	✓	17n
March C-	✓	✓	✓	✓	10n
March X	✓	✓	✓	✓	6n
March Y	✓	✓	✓	✓	8n

Table II. Memory test notations.

Notation	Action
r0	Read a zero from the selected memory location
r1	Read a one from the selected memory location
w0	Write a zero to the selected memory location
w1	Write a one to the selected memory location
↑	Increasing memory address
↓	Decreasing memory address
↕	Either increasing or decreasing memory address

Table III. Test sequence of March algorithms.

Algorithm	Test Sequence
MATS+	↕(*w*0); ↑ (*r*0,*w*1); ↓ (*r*1,*w*0)
MATS++	↕ (*w*0); ↑ (*r*0,*w*1); ↓ (*r*1,*w*0, *r*0)
MARCH X	↕ (*w*0); ↑ (*r*0,*w*1);↓ (*r*1,*w*0); ↕ (*r*0)
MARCH C	↕(*w*0); ↑ (*r*0,*w*1); ↑ (*r*1,*w*0); ↕ (*r*0); ↑ (*r*0,*w*1); ↓ (*r*1,*w*0); ↕ (*r*0)
MARCH C-	↕(*w*0); ↑ (*r*0,*w*1); ↑ (*r*1,*w*0); ↓ (*r*0,*w*1); ↓ (*r*1,*w*0);↕ (*r*0)
MARCH A	↕(*w*0); ↑ (*r*0,*w*1,*w*0,*w*1); ↑ (*r*1,*w*0,*w*1); ↓ (*r*1,*w*0,*w*1,*w*0); ↓ (*r*0,*w*1,*w*0)
MARCH Y	↕ (*w*0); ↑ (*r*0,*w*1, *r*1); ↓ (*r*1,*w*0, *r*0); ↕ (*r*0)
MARCH B	↕ (*w*0); ↑ (*r*0,*w*1, *r*1, *w*0,*r*0, *w*1); ↑ (*r*1,*w*0, *w*1);); ↓ (*r*1,*w*0, *w*1, *w*0); ↓ (*r*0,*w*1,*w*0)

Table III summarized the test sequences for the various March algorithms. An example of the MATS+ algorithm is shown as follows.

↕(w0); ↑ (r0,w1); ↓ (r1,w0)

↕ (w0): Write the 0 to the memory cells in either the ascending or descending order of the memory address (starting at the lowest/highest address).

↑ (r0, w1): Read the data in the memory cells (expect to read a 0 value from the memory cell, r0) and then

write a 1 to the memory cell in the ascending order of the memory address (starting at the lowest address).

↓ (r1, w0): Read the data in the memory cells (expect to read a 1 value from the memory cell) and then write a 0 to the memory cell in the descending order of the memory address (starting at the highest address).

The zero-one scan test is used to test the memory array with different data backgrounds such as solid zero, solid one, checkerboard, complement checkerboard, row stripes, complement row stripes, double row stripes, complement double row stripes, column stripes, complement column stripes, double column stripes and complement double-column stripes are shown in the following figures (Figs. 1–6). The zero-one scan test is modelled to examine the interaction between memory cells in the form of data pattern and cover the other common memory faults.

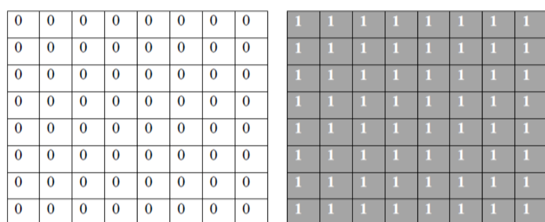

Fig. 1. Solid data background.

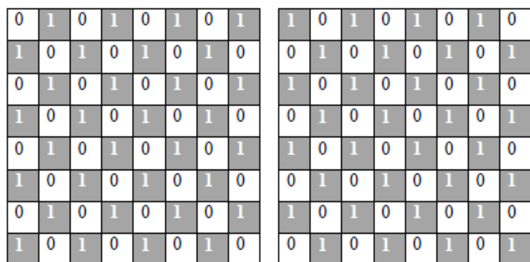

Fig. 2. Checker Board data background.

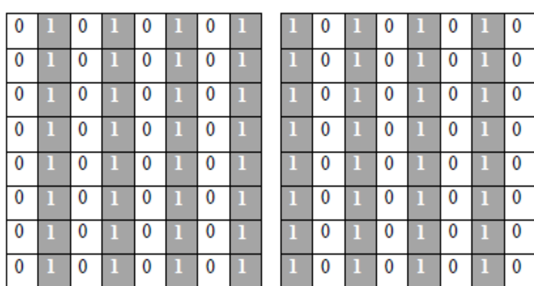

Fig. 3. Column stripes data background.

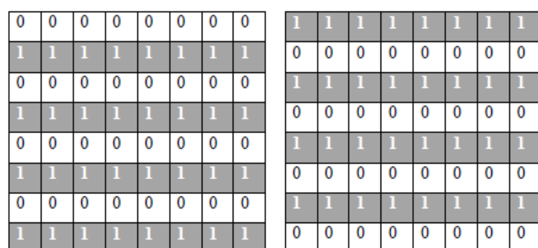

Fig. 4. Row stripes data background.

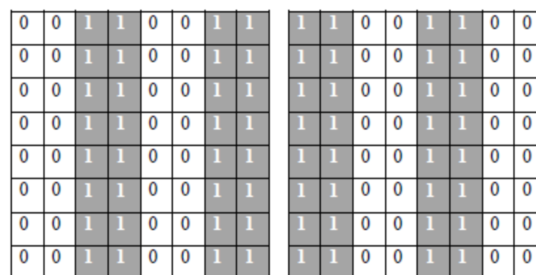

Fig. 5. Double row data background.

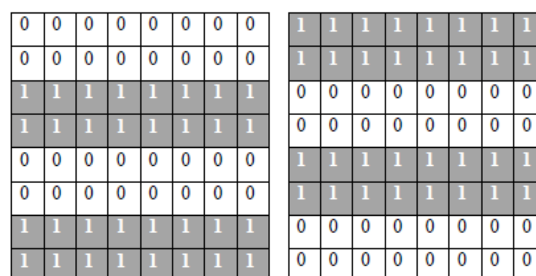

Fig. 6. Double row stripes data background.

III. Design Methodology

A. Simulation Configurations

Extensive simulation is carried out through the Quartus II software. Three types of Altera FPGAs, i.e. Stratix IV GX (High-end FPGA), Cyclone IV (Mid-range FPGA) and Apex20k (low-end FPGA) are used as the target devices in the simulations. The fault detection coverage of the MARCH algorithms is shown in Table IV. The combination of the test algorithms in Table IV will give the 100 % memory fault detection coverage for all commonly known memory defects.

Table IV. Fault detection coverage of the test algorithms.

Fault	MATS +	MATS ++	MARCH X	MARCH Y	MARCH C-	MARCH C
SAF	100%	100%	100%	100%	100%	100%
TF	0.2%	100%	100%	100%	100%	100%
SOF	100%	100%	0.2%	100%	0.2%	0.2%
AF	100%	100%	100%	100%	100%	100%
CFin	75%	75%	100%	100%	100%	100%
CFid	37.5%	37.5%	50%	50%	100%	100%
CFst	50%	50%	62.5%	62.5%	100%	100%
Complexity	5*n*	6*n*	6*n*	8*n*	10*n*	11*n*

The memory sizes that selected for the performance analyses are as follows, i.e. 4 by 4 array, 8 by 8 array and 16 by 16 array. These memories with different sizes are used as the device under test for the memory tester proposed. Furthermore, the memory sizes selected are able to fit into the FPGA chips that are used as the testbed for the simulation analyses. The sizes selected are adequate to show the current trend of the existing memory sizes available in the market. The memory fault detection algorithms modelled in this paper is able to detect all the common memory defects with 100 % fault detection coverage.

The memory test operation is shown in Fig. 7. The input data in the testbench are generated according to the test sequences listed in Table III. The tester will access the testbench that consists of the following information.

- Address
- Input data
- Memory cell enable
- Memory cell set
- Memory cell reset

The enable signal is available on each memory cell. The tester enables or disables the memory cells by controlling the enable signal. The memory cell is enabled to carry out the memory test and the other memory will be disabled. The faulty cell is disabled and the other memory cells will continue to carry out the memory test. For a read operation, the data stored in the memory cell is read and forwarded to the output file. The output file contains the following information.

- Input data from the testbench
- Time
- The data stored in the memory cell

The data stored in the memory cell must be the same as the input data from the testbench. The output file will also show the time, and the test sequence to complete the memory fault detection test.

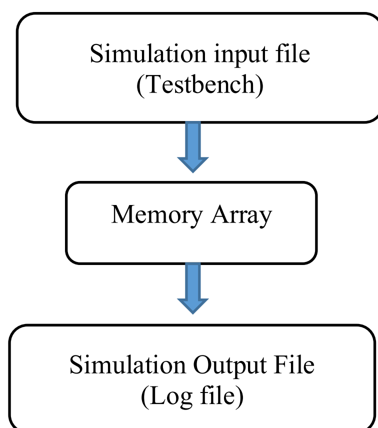

Fig. 7. Memory test operation.

B. Hardware Configurations

The 4 by 4 memory array that used as the device under test is shown in Fig. 8. The 4 by 4 memory array is modelled in the FPGA using the D flip-flops and all the flip-flops are arranged in the grid arrangement. The testbed modelled in FPGA is shown in Fig. 9. It consists of the tester and the memory array. The block diagram of the memory tester is shown in Fig. 10. Row decoder and column decoder are used to decode the row and the column numbers of the memory cell.

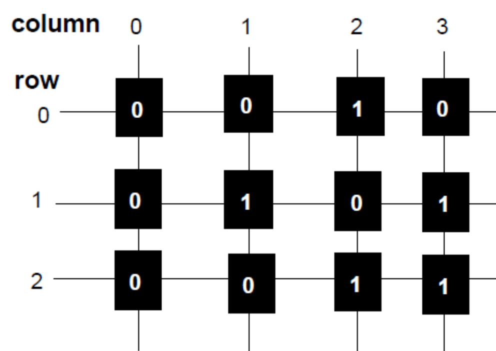

Fig. 8. 4 by 4 Memory array configuration.

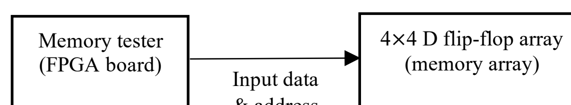

Fig. 9. Memory tester testbed configuration.

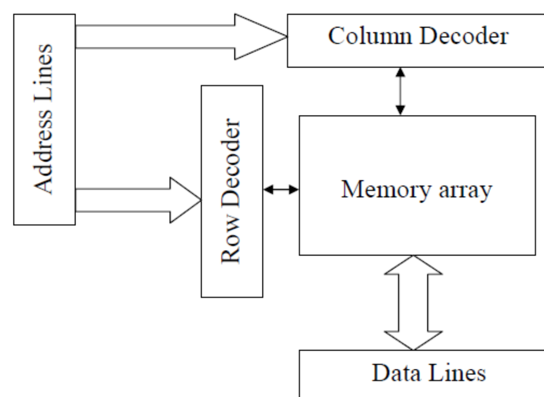

Fig. 10. Block diagram of the memory address decoder.

The processing speed of the FPGA board is slower compared to the existing ATE memory tester used in the industry. However, the cost of the FPGA is much lower than the ATE tester. The proposed memory tester is the simplified low-cost FPGA tester. The low-cost memory tester that realized in this paper is shown in Fig. 11. Only the row decoder is modelled, and the column decoder is replaced with the switching transistors in order to reduce the resources consumption. The FPGA board is realized as the memory tester. When one of the rows of D flip-flops is activated by the memory tester, the switching transistors that connected in that row will enable the D flip-flops. This allows the memory tester to write the data to the D flip-flops that are enabled. The data stored in D flip-flops will be shown through the LEDs connected to the D flip-flops respectively. If the LED is switched on when a 1 is stored in the D flip-flop and vice versa. The set and reset terminals of the D flip-flop can be used to simulate the error in order to test the response of the test algorithms.

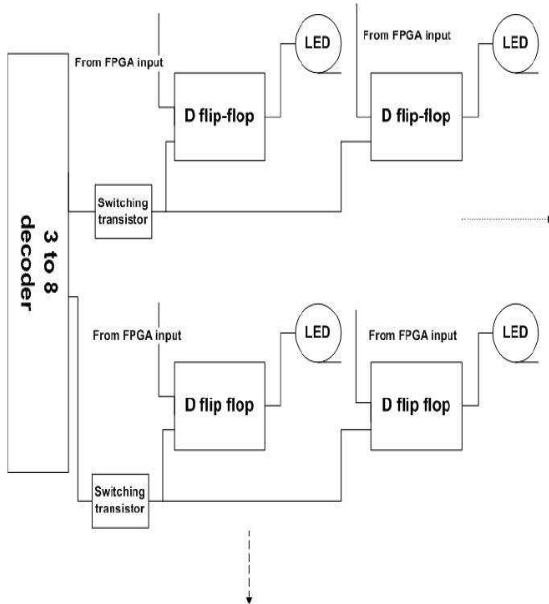

Fig. 11. FPGA realization of the memory tester.

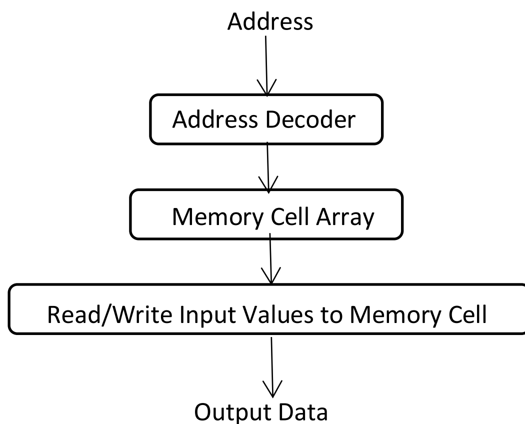

Fig. 12. Memory test procedure.

The block diagram of the memory test procedure is shown in Fig. 12. The address decoder decodes the corresponding row number from the input data and enables the memory array cells in the corresponding row to be accessed by the memory tester. The test program will test the memory as per defined by the corresponding test algorithms to write to and read from the memory cells. The test processes will be repeated over and over again until all the tests completed.

IV. Simulation Results

Two major tests are carried out in the simulation analyses, i.e. zero-one scan test and March memory tests. The zero-one scan test covers all the 12 types of data test patterns and the March tests cover all the 6 test algorithms stated in Table IV.

A. Zero-One Scan Test Simulation Results

Zero-one scan test has been implemented in the memory tester, the scan test is executed in several configurations to perform the actual memory test program that used in the industry. The scan test is evaluated in three different memory array sizes, i.e. 4 by 4, 8 by 8 and 16 by 16 in which all the three memory sizes are modelled in three different FPGAs to evaluate the effect of the FPGA maximum operation speed in the memory testing. All the memory tests are executed 10 times to perform the stability test that applied in the industry to assess the credibility of the test carried out on the memory array. The number of test looping can be increased in order to increase the confidence level and the intensity of the memory fault detection tests.

All these memory fault detection tests are modelled according to the memory test program that used in the industry of semiconductor to test the memory module during the final test. The memory fault detection tester is expandable to support the memory with higher capacity. The memory fault detection tester is realized in 3 different types of FPGAs. The performance of the zero-one scan tests is shown in Fig. 13, Fig. 14 and Fig. 15. All the test time for the zero-one scan test is the same except the solid zero data pattern. The test time for all the memory arrays and tests are the same since all the tester will write the test data to the memory cells and then read the data that stored in the memory cells in parallel. The test time is the same neglecting the size of the memory. However, for the solid zero tests, since all the data in the memory cells are 0 at the beginning of the test, it saves one clock cycle to write the 0 data into the memory cells.

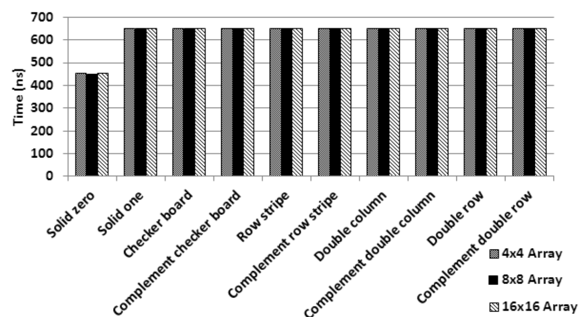

Fig. 13. Performance of APEX chip.

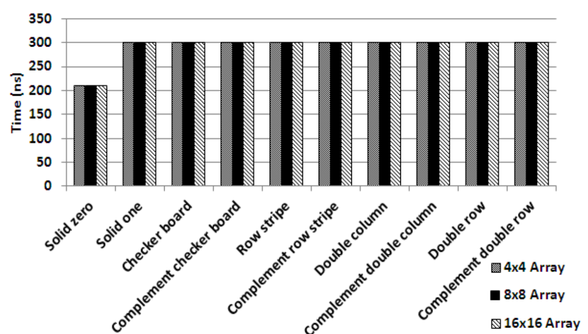

Fig. 14. Performance of Cyclone chip.

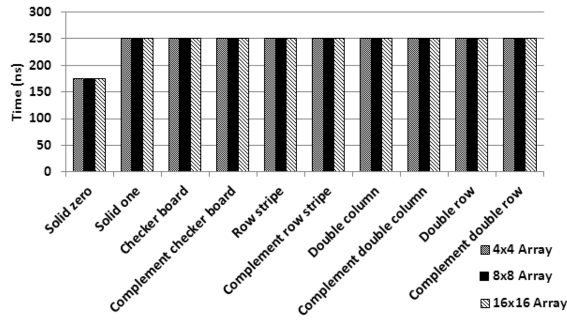

Fig. 15. Performance of Stratix chip.

As shown in the simulation results, Stratix can support the highest testing speed with 17 % faster than Cyclone, but the cost of Stratix chip is nearly 5 times higher than the Cyclone chip. APEX spends the longest time to complete all the test. The processing speed of APEX is more than 100 % slower than the Cyclone and Stratix FPGAs, but the cost of the APEX is the cheapest when compared to the other two FPGAs.

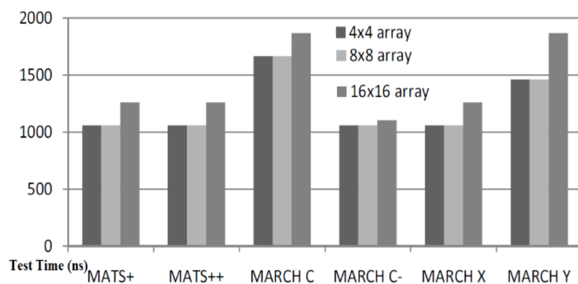

Fig. 16. Test performance of APEX chip.

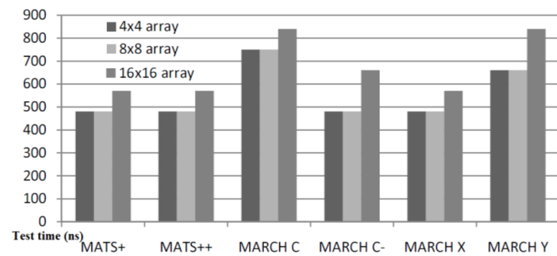

Fig. 17. Test performance of Cyclone chip.

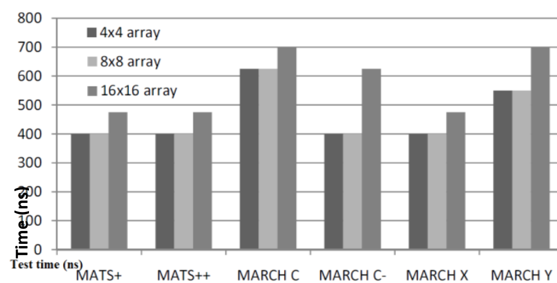

Fig. 18. Test performance of Stratix chip.

B. March Algorithms Simulation Results

Each of the March tests is repeated 10 times to increase the confidence and the intensity level of the March tests. The simulation results are shown in Fig. 16, Fig. 17 and Fig. 18 for APEX, Cyclone and Stratix FPGA respectively. It is proven that the complexity of the test algorithm is directly proportional to their test time.

As shown in the simulation results, March C algorithm is the slowest compared to the other algorithms. Followed by March Y algorithm with its performance is slightly better than March C. The performance of the other 4 algorithms (MATS+, MATS++, March C- and March X) are almost equal although their complexities are different. March C- needs a longer time to complete the test for the 16×16 memory array. Since March C- has a higher complexity compares to the other 3 algorithms.

The memory fault detection tester modelled has covered all the test algorithms that currently being used in the ATE tester to test the memory modules in the semiconductor industry. March C- holds the advantage over the other March tests. It covered more fault models compared to the other algorithms and yet it is just 10n in complexity which saves a lot of the test time. It meets the industry requirement that demanded the shortest test time in order to increase the production rate. The combination of the MATS+ and March C- provide the full coverage of all the memory faults detection stated in Table IV. The total test time required to complete the MATS+, March C- and zero-one scan test that provides the 100% of the memory faults detection is shown in Fig. 19.

The low-cost memory fault detection tester is modelled successfully and it can carry out all the test programs used in the ATE tester to test the memory. It shows that the performance of the APEX is the lowest (with the longest test time), and the performance of the Stratix is the highest (with the shortest test time). However, the performance of the Cyclone is comparable with Stratix. The total test time of the Cyclone is 17 % higher than the Startix but the cost of Cyclone is 5 times lower than the cost of Stratix. The suitable FPGA can be selected according to the performance required and the budget allocation for the low-cost memory tester.

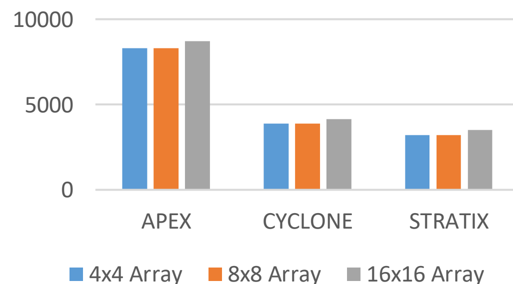

Fig. 19. Total test time to complete the MATS+, MARCH C- and Zero-one scan test.

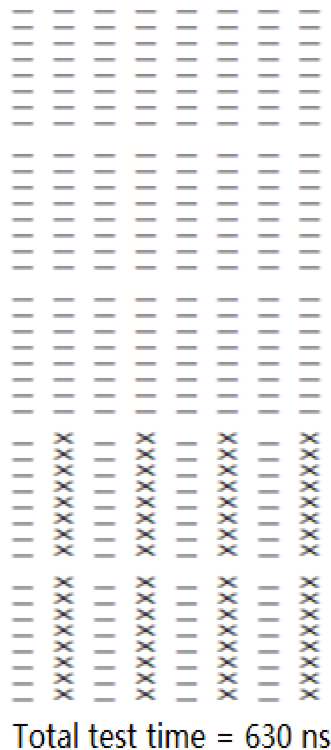

Fig. 20. Test datalog for the memory tests of the 8×8 memory array.

A PERL program is used to generate the data log after the memory tests completed. The test datalog is generated by the PERL program for the 8×8 memory array is shown in Fig. 20. The test datalog shows the test results of each memory cells in the 8×8 memory array for 5 iterations.

The '_' symbol represents the memory cell passed the memory test with the data in the memory cell is equal to the expected value after the memory test. The 'X' indicates that the memory cell failed the memory test.

V. CONCLUSION

An effective low-cost fault detection memory tester that provides the full memory faults detection coverage is modelled using VHDL. The memory tester modelled is a promising solution to provide complete memory faults detection. It modelled all the zero-one scan tests and March algorithms used in the ATE memory tester. The memory tester is realizable using FPGA and the functionality of the memory tester modelled is similar to the ATE memory tester that is used in the semiconductor industry. The memory tester is successfully modelled in three FPGA, i.e. Stratix, Cyclone and Apex. Three memory arrays with the sizes of 4×4, 8×8 and 16×16 are used to evaluate the performance of the memory testers. The performance of the Stratix and Cyclone is 61 % and 53 % faster than the Apex. The performance of the Startix is 17 % faster than the Cyclone, but the cost of the Stratix is 5 times higher than Cyclone. The designer needs to select the FPGA wisely in order to get a balance between the cost and performance of the memory tester. The PERL program is used to analyse the test result and to generate the test datalog at the end of the memory test.

REFERENCES

[1] N. Mukherjee, W. Or, A. Pogiel and J. Rajski, "High Volume Diagnosis in Memory BIST based on Compressed Failure Data," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 3, pp. 441-453, 2010.

[2] M. Melanie and H. Wunderlich, "BISD: Scan-based Built-in Self-diagnosis," in *Proceedings of the Conference on Design, Automation and Test in Europe*, vol. 1, pp. 1243-1248, 2010.

[3] M. Carvalho, P. Bernardi and M. S. Reorda, "Optimized Embedded Memory Diagnosis," in *International Symposium on Design and Diagnostics of Electronic Circuits & Systems*, vol. 14, pp. 347-352, 2011.

[4] P. Daniel and R. Chandel, "A Flexible Programmable Memory BIST Architecture," *IETE Journal of Education*, vol. 51, pp. 67-74, 2010.

[5] P. Bernardi and L. Ciganda, "An Adaptive Low-Cost Tester Architecture Supporting Embedded Memory Volume Diagnosis," *IEEE Trans. on Instrumentation and Measurement*, vol. 61, pp. 1002-1018, 2012.

[6] A. J. V. Goor, *Testing Semiconductor Memories: Theory and Practice*, John Wiley & Sons, 1998.

[7] W. L. Pang, K. Y. Chan, S. K. Wong and C. S. Tan, "VHDL Modeling of Booth Radix-4 Floating Point Multiplier for VLSI Designer's Library," *WSEAS Trans. on Systems*, vol. 12, pp. 678-688, 2013.

[8] J. H. Meza, S. Ostendorff and H. D. Wuttke, "A Configurable Test-Processor for Board-Level Testing," in *Euromicro Conference on Digital System Design*, pp. 22-29, 2016. doi: 10.1109/DSD.2016.81.

[9] I A. Grout, *Integrated Circuit Test Engineering*, Springer, 2006.

[10] G. Harutyunyan, S. Martirosyan, S. Shoukourian and Y. Zorian, "Memory Physical Aware Multi-Level Fault Diagnosis Flow," *IEEE Trans. on Emerging Topics in Computing*, doi: 10.1109/TETC.2018.2789818.

[11] T. Koshy and C. S. Arun, "Diagnostic data detection of faults in RAM using different march algorithms with BIST scheme," in *2016 International Conference on Emerging Technological Trends (ICETT)*, Kollam, pp. 1-6, 2016. doi: 10.1109/ICETT.2016.7873754.

[12] A. Johnsen, K. Lundqvist, K. Hänninen, P. Pettersson and M. Torelm, "Experience Report: Evaluating Fault Detection Effectiveness and Resource Efficiency of the Architecture Quality Assurance Framework and Tool," in *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*, Toulouse, pp. 271-281, 2017. doi: 10.1109/ISSRE.2017.31.