An Effective Architecture of Memory Built-In Self-Test for Wide Range of SRAM

Tin Quang Bui, Lam Dang Pham, Hieu Minh Nguyen, Viet Thai Nguyen, Thong Chi Le, Trang Hoang
Electrical and Electronic Faculty
Ho Chi Minh University of Technology
Ho Chi Minh City, Viet Nam
{quangtin91, vietnguyen155}@gmail.com, {lamd.pham, hieumnguyen, chithong, tranghoang} @hcmut.edu.vn

***Abstract*— Together with highly increasing integration on one chip currently, usage of memories, mainly Static Random Access Memory (SRAM), applying to wide range of functions is inevitable. However, memory faults are greatly concerned due to purpose of achieving high yield. As a result, a memory built-in self-test (MBIST) has become essential in any system obviously. As regards MBIST, while increasing criteria associating with area, frequency as well as various test algorithms has posed, current approaches have not adapted both such silicon requirements and ability of covering errors with complex algorithms yet. In this work, an effective architecture of MBIST for different SRAM memories is proposed for ensuring high ability of detecting memory faults supported by the most popular algorithms namely MARCH C- and TLAPNPSF but also satisfy strict silicon criteria. Indeed, achieving great performance based on necessary experiments on 130nm technology with Application Specific Integrated Circuit (ASIC) design flow has confirmed strong competition to current designs.**

Keywords — MBIST; MarchC; TLAPNPSF; SRAM; ASIC

I. INTRODUCTION

Techniques applied for detecting error towards SRAM memory has currently based on effective algorithms integrated in MBIST architecture. Below Table I summarizing conducted survey in [1] presents current algorithms accompanying distinct types of errors. Some popular kinds of failure on SRAMs are defined such as Stuck at Fault (*SAF*), Transition Fault (*TF*), Address Decoder Fault (*ADF*), Coupling Fault (CF). From considerable statistics, mainly authors such as [1], [2] and [3] approach only group of MARCH algorithms with the main reason of decreasing chip area.

TABLE I. ALGORITHMS TO DETECTING MEMORY FAULTS

Test	Fault Coverage
MATS+	SAF
March Y	SAF, TF, ADF, some CFs, some linked TFs
MARCH C–	SAF, TF, RDF, IRF, ADF, CFs
March X	SAF, TF, ADF, some CFs
March LR	Also linked faults
March SR	SAF, TF, CFs, IRF,RDF, DRDF, SOF
March A	SAF, TF, ADF, some CF, some linked CFs
March B	SAF, TF, SOF, IRF, RDF, ADF, some linked TFs
March LA	SAF, TF, ADF, CFs, some linked faults
March AB	dRDF, dIRF, dDRDF, dCFds, dCFrd, dCFdrd, dCFir, static linked faults

TABLE II. ALGORITHMS ACCOMPANY TYPE OF MEMORY FAULTS [4][1]

Algorithms	Fault Memory							
	SAF	*TF*	*CFin*	*CFid*	*CFdyn*	*NPSF*		
						A	*P*	*S*
MATS	X							
MATS++	X	X						
MARCH C–	X	X	X	X	X			
MARCH A	X	X	X					
TLSNPSF	X					X		
TLAPNPSF	X	X				X	X	X
TDANPSF	X							X

However, in Table II, [1] and [4] indicate that such MARCH algorithms can't cover all memory faults that related to patterns of neighborhood cells. Through Table II, although most popular memory faults can be covered by only two algorithms including *MARCH C–* and *TLAPNPSF*, implementing *TLAPNPSF* on hardware has become critical issue due to the complex architecture dominating great chip area. Therefore, both area overhead and high flexibility are also discussed in [5].

As regards SRAM configuration, most approaches have limited ranges because of the ability of generating address decode as well as configuring test algorithm accompanying different SRAM memories. Particularly, while [7] only shows the results of one SRAM configuration 1Kx8, [6] proposes to verify several configurations as 8Kx32, 1Kx48, 4Kx16 by only basing one configuration of MBIST 8Kx48, but such architecture have problem related to the real-time issue during verifying.

From such circumstances, it is fact that the current MBIST designs have experienced a low ability of detecting memory faults as well as limited support towards different SRAM configurations simultaneously because of the exact trade-off between integral silicon requirements and complex test algorithms. As a result, authors propose the effective MBIST architecture by integrating *MARCH C–* and *TLAPNPSF* to fully cover memory faults, which not only adapts the crucial silicon requirements but also is compatible with wide range of SRAM memory configurations from 1Kx8 to 64Kx64.

The rest of paper is organized follows. Section II proposes MBIST architecture presenting how to implement *MARCH C–* and *TLAPNPSF* on hardware. Section III describes the positive results on 130nm technology witnessed by ASIC design flow. Finally, Section IV consults the paper and future work.

II. MBIST HARDWARE ARCHITECTURE

A. *MARCH C− model and hardware architecture*

MARCH C−, one of the most popular algorithms in MARCH group, can cover all simple failures of memory cell. Consequently, a state machine of *MARCH C−* is shown in Fig. 1.

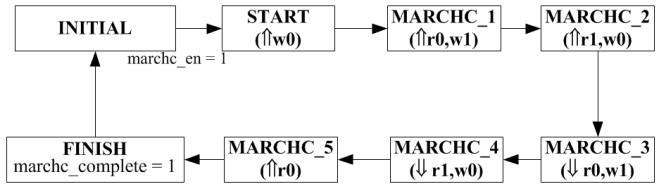

Fig. 1. MARCH C− state machine

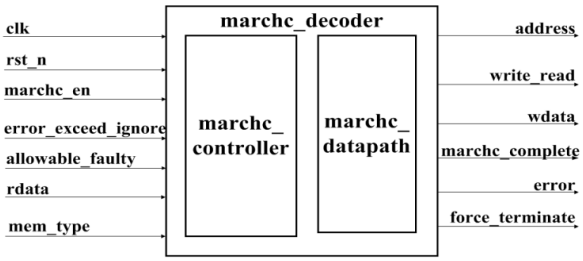

Fig. 2. MARCH C− hardware architecture

In order to adapt a wide range of SRAM configurations, only highest address parameter is firstly configured and address the decode function with the role of increasing or decreasing address index in order to cover all words of memory. From such constrains, perspective of *MARCH C−* hardware design is illustrated in Fig. 2, in which address and data are generated by *marchc_datapath* and *marchc_controller* keeps the role of state machine as Fig. 1.

B. *TLAPNPSF model and hardware architecture*

According to Table II, *TLAPNPSF* algorithm is developed to be able to cover the feasibly complex faults. It is vital that a cell is covered around by four neighborhood cells and four diagonal cells, but most authors in [8], [9] approach only patterns of memory faults relating to such neighborhood cells as presented Fig. 3, since it is more practical for such type of memory considered. Obviously, bit cell (A1) - base cell- is surrounded by four neighbor bit cells (n1, n2, n3, n4) and the fabrication issues may create a unexpected connection between two or many bit cells.

Instead of verifying every bit cell sequentially, the below approach as Fig. 4 reveals the effective solution, where all bit cells are classified into two categories. In every category, there are two types of bit cells called base cell namely *A1, A2, B1, B2, C1, C2, D1, D2* and neighborhood cells known as *n*. Based on such division, changing patterns of neighborhood cells can be applied to verify all base cells of one word at the same time since neighborhood cells are independent.

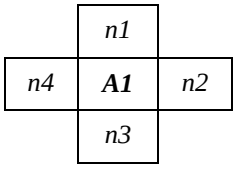

Fig. 3. Base cell and neighbor ones relating to unexpected faults

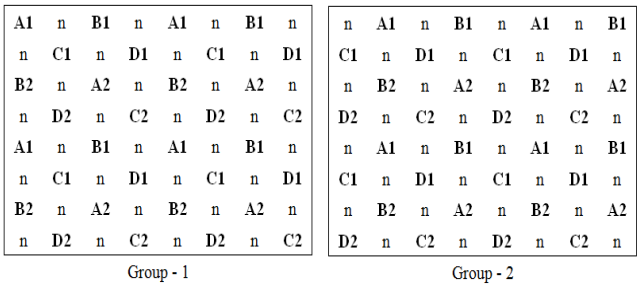

Fig. 4. Two groups of base cells and neighborhood cells.

After completely verifying all base cells (*A1*) at the first word of group 1, all base cells (*A1*) of the fifth word are verified and such verification is continued to accomplish all base cells (*A1*) belonging to the 4n word where *n=0, 1, 2, ...* in group one. Until all base cells of group one are finished, the second group is started.

From such circumstances, state machine as Fig. 5 are presented to clarify the transition of states. Fig. 6 also presents overall *TLAPNPSF* hardware together with state machine, which be similar to *MARCH C−* structure with *apnpsf_controller* and *apnpsf_datapath.*

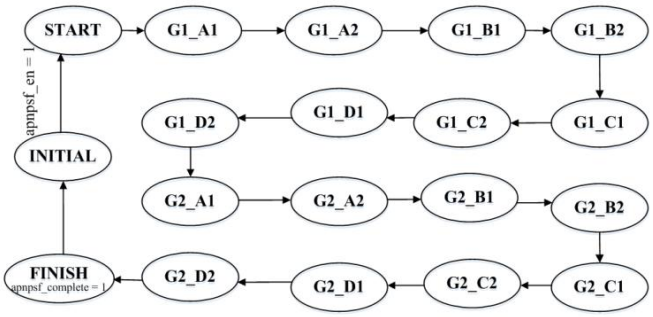

Fig. 5. TLAPNPSF state machine.

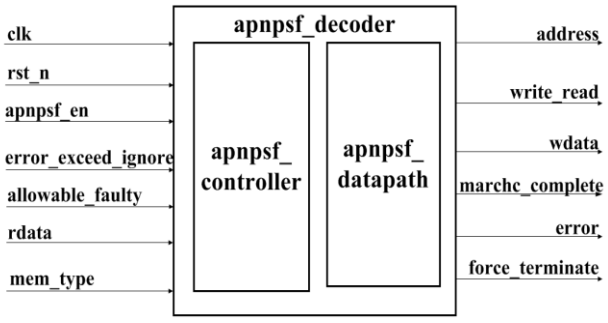

Fig. 6. TLAPNPSF hardware architecture.

C. MBIST architecture

MBIST hardware design in Fig. 7 will combine both algorithms *MARCH C–* and *TLAPNPSF* in *mbist_decoder*. Block *mbist_decoder* have *marchc_decoder* and *apnpsf_decoder* inside as said in section A and B. Controller block will control system to choose algorithm via the state machine as describing in Fig.8

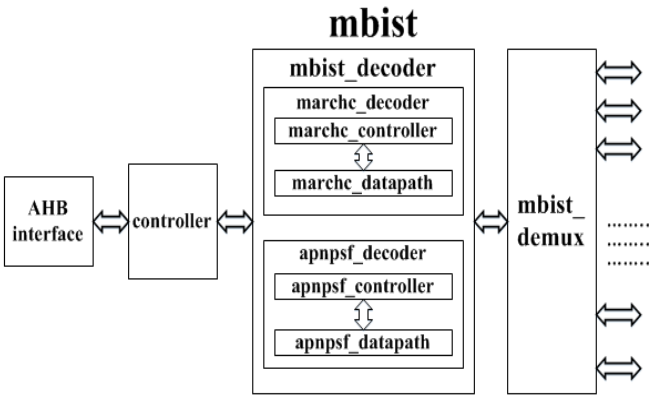

Fig. 7. MBIST hardware architecture.

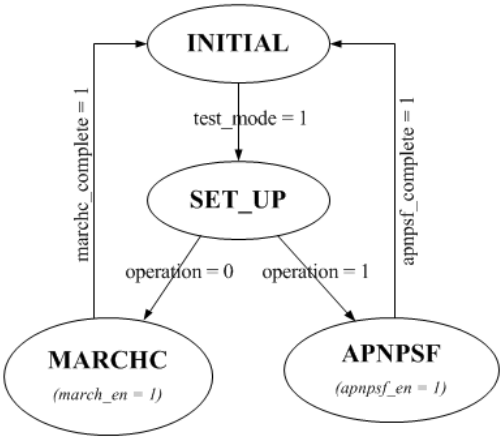

Fig. 8. State machine of *Controller* block.

However, the default configuration with both test algorithms can be independently activated by a trigger port instead of CPU commands through *AHB interface*. *AHB interface* is also applied to configure the depth of memory. One of the most important structures integrated in MBIST architecture is *mbist_demux* block to apply for many different SRAM configurations. To be more detail, *mbist_demux* is generated automatically in RTL develop step and supported by own tool on Perl script language to be able to adapt wide range of different SRAM memories.

III. IMPLEMENT AND RESULTS

ASIC design flow with 130nm technology is applied to design MBIST hardware architecture. Based on presented MBIST architecture, the ability of expanding and integrating other algorithms are ensured because of ability for generating *mbist_demux* structure in RTL design automatically. Besides, architecture with dynamic configuration of highest address parameter and the state machine design enables to reach the wide range of memory depth. To ensure all fault cases and all configurations are covered, an automatically verification environment is designed to make progress conveniently as defined in Table III.

TABLE III. TYPICAL FAULTS WITH CORRESPONDING CODE

File	Description
0101	Data are always fixed value
0201	The fourth bit of word always equals to 1
0301	The fourth bit is inverted if the fifth bit is changed from 0 to 1
0311	The fourth bit is kept constant (1'b1) if the fifth bit is changed from 0 to 1
…	…..
072F	If the neighbor bits (n1, n2, n3, n4) are changed from 4'b0000 to 4'b1111, base bit cell (A1) is kept 1'b0(n1n2n3n4 is from 0000 to1111, A1 equals to 1'b0)

With 100 test cases in the Table III, 100% faults related to SAF, TF, RDF, IRF, ADF, CFs, ANPSF, PNPSF, SNPSF are detected by the design hardware. In Fig. 9, a fault is detected by using TLAPNPSF algorithm.

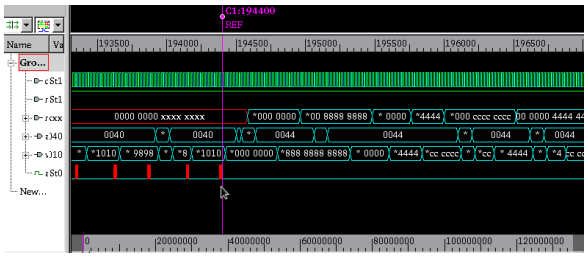

Fig. 9. Fault is detected by TLAPNPSF algorithm

In this paper, experiments on MBIST configuration supporting for a wide range of SRAM configurations from 1Kx8 to 64Kx64 has performed and confirmed by the physical superiorities in comparison to other references. To be more specific, Table IV presents collected data in order to compare the different performance among designs. Through such Table IV, although both *MARCH C–* and *TLAPNPSF* are integrated in the proposed design, result of 11207 gates has witnessed a low gap compared with other designs with only *MARCH* algorithm as 27469 gates in [12], 7942 gates in [5].

In term of frequency, the proposed design has experienced high frequency at 500 MHz opposed to others with same technology in [5] or [10] at 400 MHz and 300 MHz respectively. Most reference designs have referred to few configurations with only *MARCH* algorithms while this work supports a wider range with the ability of reconfiguration not only for the depth of memory but also the length of word.

In order to perform the timing consumption exactly, (1) and (2) which experienced from waveform simulation by VCS tool confirm how to calculate the number of cycles in every algorithm and the exact SRAM configurations. (1) and (2) together with Table V collects information to perform timing consumption of different SRAM configurations to achieve the high frequency at 500MHz. Through Table V, supported maximum configuration at 64Kx64 has witnessed real-time test applying two proposed algorithms.

$$T_{MARCHC-} = 51 \times 2^n \quad (1)$$

$$T_{TLAPNPSF} = 1645 \times 2^n \quad (2)$$

where *n* is number of address bits

TABLE IV. PERFORMANCE COMPARISION

Authors	Technology	Algorithm	Memory capacity	Area	Freqency
[5]	130 nm	March C+, March SS, March Y, Walking, Galloping	16.3Kx16	7942 gates	400 MHz
[7]	130 nm	March C+, March C–, Checkerboard	1Kx8	1582 gates	500 MHz
[10]	130 nm	March C+	16Kx16	6427 gates	300 MHz
[11]	90 nm	March C–, MATS+	8Kx32	3400 gates	500 MHz
[12]	250 nm	March C–	16M x8	27469 gates	344 MHz
Proposed MBIST	**130 nm**	**MARCH C–, TLAPNPSF**	**1Kx8 to 64Kx64**	**11207 gates**	**500 MHz**

TABLE V. TIMING CONSUMPTION WITH FREQUENCY AT 500 MHZ

Configuration	MARCH C–	TLAPNPSF
Maximum configuration (64Kx64)	0.006 s	0.215 s
Minimum configuration (1Kx8)	0.0001 s	0.003 s

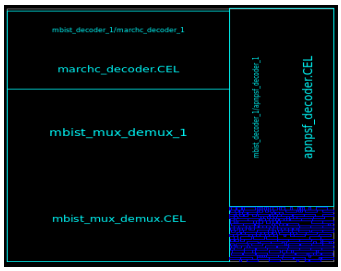

Fig. 10. Results of Floor Plan.

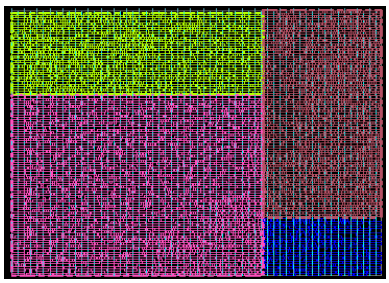

Fig. 11. Physical design results of Place and Route.

Finally, in Fig. 10 and Fig. 11, physical design phase is also implemented to confirm the exact performance of proposed MBIST architecture. Extracted information including post–layout netlist and timing format (SDF file) are verified again by all test cases in RTL verification step to confirm matching functions and performance as previous Table IV.

IV. CONCLUSION

In this paper, effective hardware architecture of MBIST is proposed to perform superiorities in comparison to others designs. Not only has such architecture confirmed the great ability of detected memory faults supported by two popular test algorithm but it also meet the strict silicon requirements. Proposed architecture can be generated the feasible MBIST to adapt any SRAM configurations. In future work, a complex compiler will be built to perform any algorithm followed user's purposes to be compatible any system as well as different types of memory.

REFERENCE

[1] B. Singh, A. Khosla and S.B. Narang, "Area overhead and power analysis of March algorithms for memory BIST," International Conference on Communication Technology and System Design, vol. 30, p. 930-936, 2012.

[2] G. Harutyunyan, S. Shoukourian, V. Vardanian and Y. Zorian, "A new method for March test algorithm generation and its application for fault detection in RAMs," IEEE Transction of Computer-Aided Design of Integrated Circuits and System, vol. 31, pp. 941-949, 2012.

[3] Y. J. Huang, C. W. Chou and J. F. Li, "A low cost built in self test scheme for an array of memories," 15th IEEE European Test Symposium, pp. 75-80, 2010.

[4] M. Riedel and J. Rajski, "Fault coverage analysis of RAM," 13th IEEE VLSI Test Symposium, pp. 227-234, 1995.

[5] Y. Park, H. S. Kim, I. Choi, and S. Kang, "A flexible progarmable memory BIST for embedded single port memory and dual port memory," Electronic and Telecomunications Research Institute Journal, vol. 35, pp. 808-818, October 2013.

[6] Q. L. Rao, C. He and Y. M. Jia, "A memory built in self test architecture for memories different in size," IEEE Circuit and System International Conference on Testing and Diagnosis, pp.1-4, 2009.

[7] C. F. Lin and Y. J. Chang, "An area-efficient design for programble memory built in self test," IEEE International Symposium on VLSI Design Automatic and Test, pp. 17-20, 2008.

[8] C. Huzum and P. Cascaval, "All multibackground March test for all static sample Neighborhood Patterns Sensitivity Fault in RAMs," 15th International Conference, pp. 01-06, 2010.

[9] Y.-J. Huang and J.-F. Li, "Testing ternary content addressable memories with active neighborhood pattern sensitive faults," IET Computers Digital Techniques, pp. 246-255, 2007.

[10] Y. Park , J. Park, and S. Kang, "An efficient BIST architecture for embedded RAMs," International Conference On Electronic Information and Communication (ICEIC), vol. E92D, pp. 2508-2511, 2009.

[11] A. J. van de Goor, S. Hamdoui and H. Kuhuner, "Generic, orthogonal and low cost March element based memory BIST," International Test Conference, pp. 1-10, 2011.

[12] Y. Park , M. H Yang, Y. Kim, D.Y. Lee, H. Yoon and S. Kang, "An efficient BIST architecture for embedded dual port memories," IEEE International Solid –State Circuits Conference, pp. 157 – 160, 2007.