International Academic Institute for Science and Technology

International Academic Journal of Science and Engineering
Vol. 4, No. 2, 2017, pp. 59-74.
ISSN 2454-3896

International Academic Journal of Science and Engineering

www.iaiest.com

Concurrent BIST for Embedded SRAMs in SoCs

Esmaeil Nik Maleki

Department of Electrical Engineering, Sheikhbahaee University, Isfahan, Iran,

Abstract

With the progress of deep submicron technology and increasing design complexity, hundreds of memory cores with different size and configuration are embedded in system-on-chips (SoCs). These memory cores occupy a noticeable silicon area and need an efficient and low-cost test methodology. Built-in Self-test (BIST) is a practical solution provides a certain degree of reliability and flexibility. However, designing a memory BIST circuit is a considerable task which imposes some limitations to the designer. These limitations can be grouped in three categories: area overhead, test and diagnosis time and fault coverage. This paper proposes an at-speed and parallel BIST scheme based on IEEE std 1500 for concurrent testing of all SRAM memories embedded in a SoC. A novel structure of IEEE std 1500 wrapper, supports at-speed and parallel testing which reduces the test time. Also by including an algorithm analyzer, we can apply every march algorithm to the BIST circuit which results to a high fault coverage. Finally, sharing method for homogeneous memories improves the area overhead of the design.

Key words: Syetem-on-chip (SoC), embedded memory cores, Built-in Self-test (BIST), march algorithm, fault-coverage.

1 Introduction

Today by the fast innovation in VLSI technology, system-on-chips (SOCs) are widely used in current high-performance applications [1,2]. These SoCs usually include a large amount of embedded memory cores with extremely high density which makes the memories more prone to manufacturing defects than other types of on-chip circuitries [3]. Therefore, designing an efficient test and diagnosis methodology for these embedded memory cores is very important. Since direct access to memory ports is virtually impossible and at-speed testing is difficult to achieve, BIST is a more practical method to test and diagnose embedded memories.

Although BIST is an effective approach, it has some limitations all of which directly influence the cost and quality of test. One of the drawbacks is BIST circuit area overhead in terms of both silicon area and routing. This overhead is usually small with respect to the memories under test. But an SOC contains a hundred of memory cores which most of them are small memories. If each of these memory cores has a self-contained BIST circuit, the area overhead of the whole BIST circuits can be very high [4]. Therefore, sharing a test circuit between multiple memory cores is essential in reducing the area overhead and cost. On the other hand, SOCs usually include a large amount of homogeneous memory cores which physically locate close to each other. The adjacency can help sharing the BIST circuits for these homogeneous memory cores.

Another essential factor in designing a memory BIST circuit is the test and diagnosis time. A serial BIST like BIST based on IEEE std 1500, has a long and unacceptable test and diagnosis time, because this standard can't implement a parallel test and manage multiple cores concurrently [5]. On the other hand, parallel interface to memory cores for implementing a parallel BIST, increases the routing overhead. Therefore, we have to evaluate the requirements and consider constraints to select the best choice between serial or parallel method.

The fault coverage of the BIST approach is another point which directly influences the quality of test. The conventional BIST approaches usually implement a limited number of march test algorithms or combine the operations to generate an algorithm which increases the area overhead. However, deep submicron process technologies and design rules introduce physical defects that are not screened when using the march algorithms developed for previous process generations. Therefore, BIST architectures should be programmable to support every march algorithm to increase the fault coverage and to find the most suitable algorithm for the manufacturing process [6]. On the other hand, implementing an at-speed test can increase the fault coverage. Because many defect types have a time-dependent nature. These defects can be screened out only when normal operating frequencies are applied during test.

Considering the above requirements and constrains, various BIST structures have been proposed [5]-[13]. In [12], a modular test wrapper for small wide memories has been proposed. The interfaces between the memories and BIST circuit are based on IEEE std 1500. This scheme allows for at-speed test at low area overhead. In [9], a BIST approach based on sharing the test circuit and parallel testing has been proposed. In [13], the BIST circuit is shared to reduce the area overhead. The test circuit consists of a BIST controller and a wrapper. The wrapper and the BIST controller are shared between the identical memories and the embedded memories in different functional blocks, respectively. Reducing the routing area overhead has been considered in [5]. The proposed hybrid BIST scheme uses a serial test interface to control test pattern generators of the memories under test. Each group of homogeneous memories has a self-contained test pattern generator which tests the memories in parallel. In [7,8,10] a serial test interface for small memories has been proposed. These serial interfaces are mainly based on IEEE std 1500. This standard can reduce the area overhead resulting from the routing and I/O connections, however it has some drawbacks. The area overhead of the IEEE std 1500 wrappers is high. Also the control signals based on IEEE std 1500 needs to connect to a centralized controller which increases the complexity of the routing. On the other hand, testing multiple memories using a serial BIST based on IEEE std 1500 requires a long and intolerable test time.

In this paper, we propose a parallel BIST scheme for concurrent testing of all SRAM memories embedded

in a SOC. We called this structure as Embedded SRAM Concurrent BIST or ESC BIST. ESC BIST is a hierarchical parallel scheme which reduces test time compared with the traditional parallel structures. The proposed structure adopts with IEEE std 1500, but uses a novel structure for 1500 wrapper which supports at-speed and parallel testing. These wrappers are partly shared between the identical memories to reduce area overhead. Also, ESC BIST uses a shift register to receive the march algorithm from ATE and an analyzer to analyze and decode the algorithm which can improve the programmability and fault coverage of the scheme. Considering the fact that identical memories physically locate in a group, we design a self-contained test pattern generator and comparator to locate adjacent to each group. This grouping approach reduces the complexity and overhead of the routings. Moreover, the area overhead of the BIST structure is improved compared to the similar schemes.

The organization of this paper is as follows: section 2 presents a brief explanation about the general specification of the scheme and then introduces the components of the structure in detail. Sections 3 and 4 describe the functionality of the design in test and diagnosis modes and the features of the structure respectively. In section 5, we present the results of simulation and synthesis on a group of embedded memory cores that confirm the proposed structure effectively reduces the test time and area overhead. The paper is concluded in section 6.

2 ESC BIST Architecture

To analyze ESC BIST structure, we use a benchmark as an example which is shown in figure 1. This case study contains twenty four memory cores in six different configurations which all of them have a small address space and wide data words.

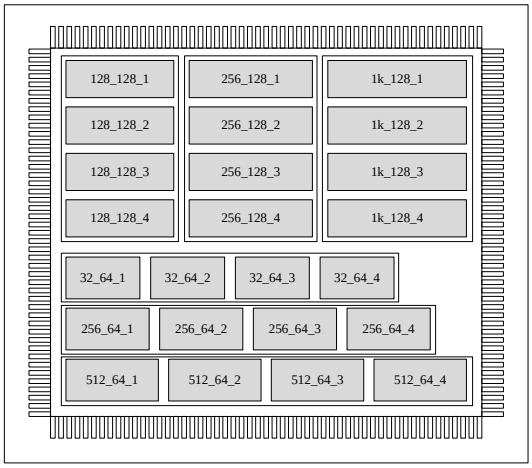


Fig 1. Floor plan of the benchmark

Figure 2 shows ESC BIST structure overall view based on the used benchmark. ESC BIST contains three shared parts including BIST controller, algorithm-analyzer and algorithm-register which are shared between all memories. The unshared parts consist of IEEE std 1500 wrappers, test pattern generators and comparators which is designed for each memory group separately. In the following subsections we explain each part of ESC BIST in detail.

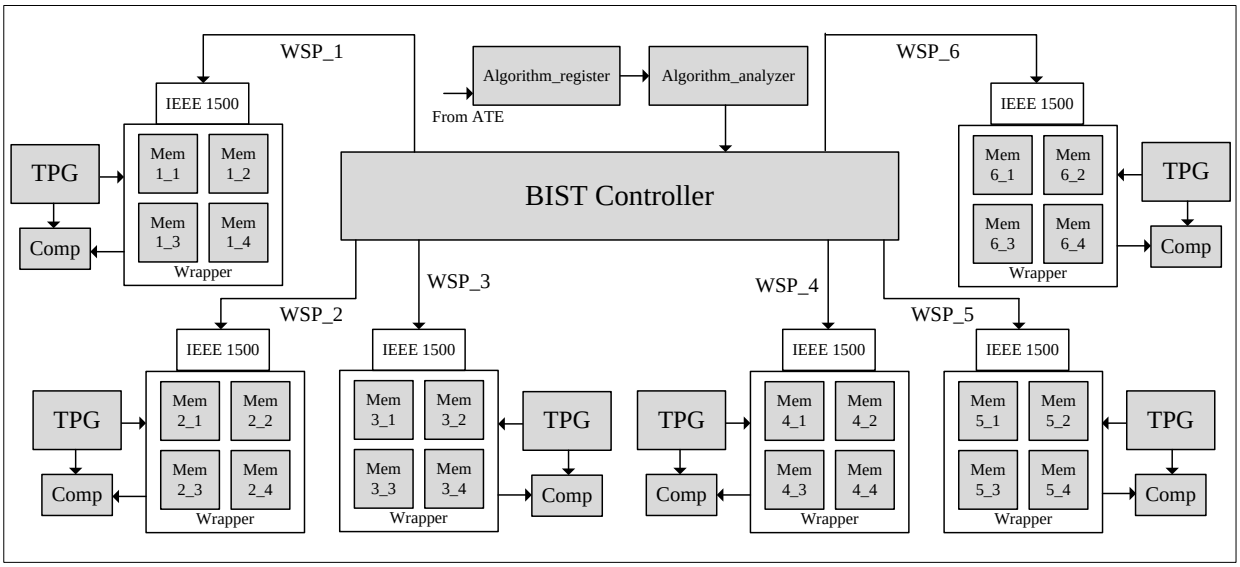


Fig 2. ESC BIST structure (TPG: Test Pattern Generator; Comp: Comparator; WSP: Wrapper Serial Port)

2.1 Algorithm-register

To prevent saving algorithms in the BIST structure which increases the area overhead and causes less programmability, we include a 128-bit shift register called algorithm-register. Algorithm-register receives the march algorithm code from ATE. Using such a structure, the test process can be repeated for any number of march algorithm which increases the design programmability.

Figure 3 depicts the structure of the algorithm-register. The register has a serial input from ATE and serial output to algorithm-analyzer, as well as three control signals from the BIST controller. When the BIST controller asserts AR_shift, the code is shifted from ATE to the algorithm-register in 128 clock cycles.

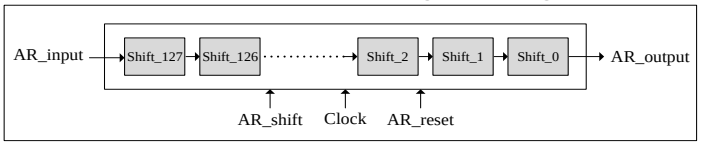


Fig 3. The structure of the algorithm-register

Figure 4 shows the shifted code from ATE which has three fields. The first field has one bit and represents the beginning of the shift process from ATE to the algorithm-register. The next field has four bits and contains the number of elements of the march algorithm. Finally the third field is composed of 123 bits and saves the algorithm code. To convert the march algorithm into a code, we use an approach which is defined in [14].

Field _1(1 Bit)	Field _2(4 Bit)	Field _3(123 Bit)

Fig 4. Fields of the serial data from ATE

2.2 Algorithm-analyzer

The algorithm-analyzer is a decoding circuit for march algorithm and extracts the required information to generate the test pattern. Figure 5 shows the block diagram of the algorithm-analyzer. In this structure, a decoder and a control state machine implement decoding of march algorithm. The results of the analysis are saved in four registers.

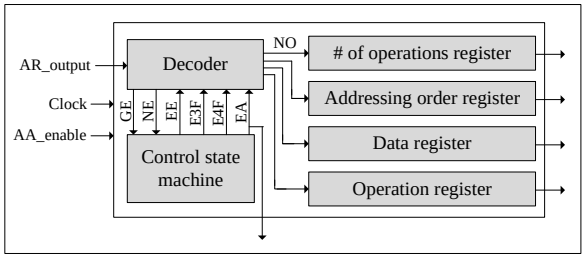


Fig 5. Structure of the algorithm-analyzer (GE: gets # of elements; NE: # of elements; E3F: end 3bit fields; E4F: end 4bit fields; EE: end element; EA: end analysis)

First, the algorithm-analyzer receives the march algorithm code sequentially from the algorithm-register within M clock cycles which M represents the number of code bits. After receiving the code, the algorithm-analyzer extracts the information like the addressing orders, operation types (read or write), test data (0 or 1) and the number of primitives and elements from the code. Finally, the analyzer sends extracted information to the BIST controller. Figure 6 shows the operation flow of the algorithm-analyzer. In this algorithm, after extracting the number of elements from the code (NO), decoding operation starts. In this operation, all three and four bit fields of the algorithm code [14] decode where each of them belong to a different primitive. The extracted information includes the addressing orders, operation types (read or write) and test data (0 or 1) which saves in three different registers. When all primitives decode, the EA signal informs the end of operation to the BIST controller.

2.3 Test pattern generator

In our proposed structure in figure 2, the test pattern is composed of memory address and data background. Each memory group has a dedicated test pattern generator which is physically located close to the group. Figure 7 shows a block diagram of the test pattern generator of ESC BIST which is composed of four main parts. The address generator block produces the memory addresses while the three other blocks serve as a data background generator.

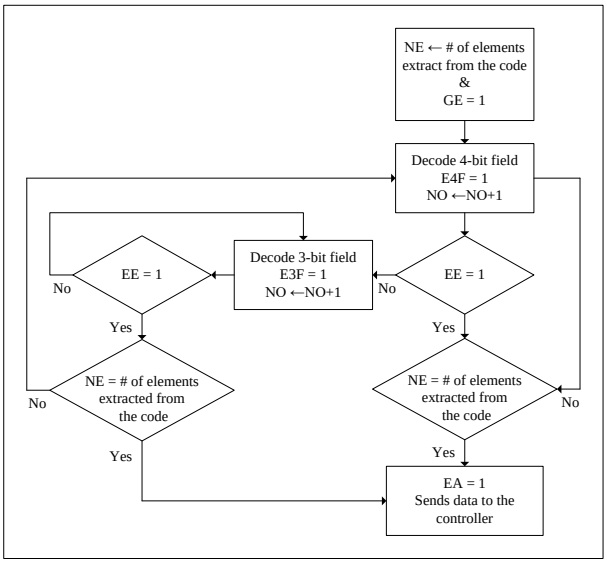


Fig 6. Operation flow of the algorithm-analyzer

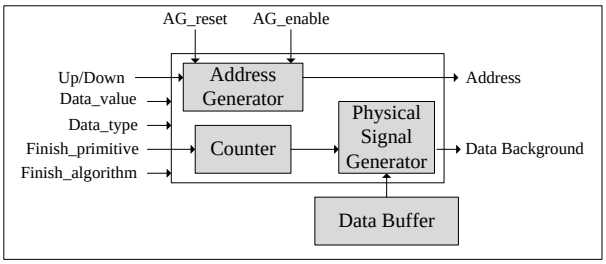


Fig 7. Block diagram of the test pattern generator

The address generator is a simple up/down counter. All address generator input signals come from the BIST controller. The output signal is the produced address which applies to all identical memories in a group concurrently. The reminder of the test pattern generator produces corresponded data background. We apply a systematic way of converting tests for bit-oriented memories into tests for word-oriented memories which is presented in [15]. Such a march test is a concatenation of two march tests: a modified bit-oriented march test and a march test for intra-word faults. A modified bit-oriented march test is a test where the bit-oriented operations $r0$, $r1$, $w0$ and $w1$ are replaced with the word-oriented operations r_D, $r_{\overline{D}}$, w_D and $w_{\overline{D}}$ whereby D demonstrates data bit and is set to all 0s. The march test for intra-word faults consists of a single march element with an arbitrary addressing order. Among the nine groups of known intra-word faults [15-16], tests for uCFdsts are at the top of the fault coverage hierarchy. The number of the data backgrounds and the read/write operations of the element which detects uCFdsts can be calculated with the equation (1) from [15].

$$\begin{aligned} d &= 3\beta + 3 \\ TL &= 7\beta + 6 \\ \beta &= \log_2^B \end{aligned} \tag{1}$$

In which d is the number of data backgrounds, TL is the number of read/write operations and B is the word length. This march element is shown below in which D_i represents the data backgrounds.

$$\updownarrow (w_{D_0}, r_{D_0}, r_{D_0}, w_{D_1}, r_{D_1}, r_{D_1}, w_{D_2}, w_{D_3}, r_{D_3}, r_{D_3}, w_{D_4}, r_{D_4}, r_{D_4}, w_{D_5}, w_{D_6}, r_{D_6}, r_{D_6}, \ldots, w_{D_{d-1}}, r_{D_{d-1}}, r_{D_{d-1}})$$

To produce a symmetric algorithm, we can add any number of read operations, without any impact on the fault coverage. Therefore, the proposed modified intra-word element can be shown as follows and the number of operation in this modified element can be calculated from equation (2):

$$\begin{aligned} &\updownarrow \left(\left\{ w_{D_i}, r_{D_i}, r_{D_i} \right\}_j \right) \quad i \in \{0,..., d\} \;\; , \;\; j \in \{0,..., 3d-3\} \\ &d = 3[3\beta + 3] - 3 = 9\beta + 6 \end{aligned} \tag{2}$$

So the structure of the march algorithm we use in the proposed BIST circuit is as follows which can enable applying any bit-oriented march algorithm to the BIST circuit without any modification:

$$\textit{Modified bit} - \textit{Oriented March} + \updownarrow \left(\left\{ w_{D_i}, r_{D_i}, r_{D_i} \right\}_j \right) \quad i \in \{0,..., d\} \quad j \in \{0,..., 3d-3\}$$

Since all of the memories in this scheme have 64-bit or 128-bit data word, we can save the data backgrounds of a 64-bit memory in an array which is accessible by all test pattern generators. Then the backgrounds of 128-bit memories can be produced using this array [15]. This approach reduces the additional operations for generating the data backgrounds and so the area overhead of the test pattern generators.

In figure 7, the data_type signal determines the background type. When this signal is 0, the data back-

ground belongs to the bit-oriented march algorithm and the data_value signal determines the value of the data background. If the data_type signal equals to 1, the test pattern generator produces the data background belongs to the march algorithm for intraword-faults. To generate the data background in this case, the test pattern generator uses data buffer, counter and physical signal generator with respect to the structure of the intraword-fault element.

2.4 ESC BIST Comparator

The comparator receives the memory responses and compares them with the correct responses come from the test pattern generator. To reduce the routing complexity, we design one comparator for each memory group which physically locates close to that group. Figure 8 represents the block diagram of a comparator in ESC BIST structure which is composed of a comparator, a delay generator to delay the correct responses and three sets of registers to save the diagnostic information.

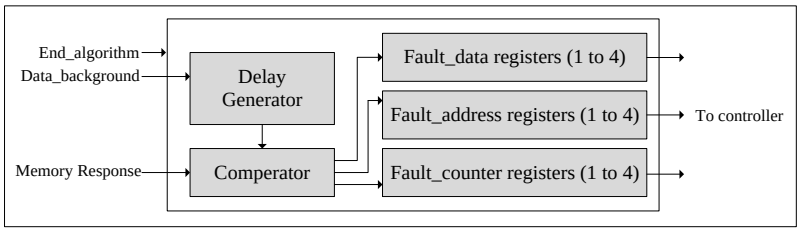


Fig 8. ESC BIST comparator Block diagram

The comparator receives the memory responses two clock cycles later than the correct responses from the test pattern generator. This delay causes by the circuits exist between the comparator and the test pattern generator. So the delay generator circuit delays the data background for two clock cycles. The results of comparison are saved in three sets of registers named fault-counter, fault-address and fault_data. The fault_counter register of each memory saves the number of detected faults. The addresses of these faults and the faulty word are also saved in the fault_address and fault-data registers respectively. The length of the fault_address and fault-data registers differs from each memory group to another but they are the same in each group. Every time a fault detects, the address of the fault and the faulty word overwrite to the fault_address and fault-data registers respectively. So at the end of the algorithm run, we have the number of the detected faults and the address and faulty word of the last fault for each memory of a group. This information is then used for the diagnosis process.

2.5 Modified IEEE std 1500 wrapper

Considering the fact that the IEEE std 1500 wrappers occupy a noticeable silicon area, we propose a modified wrapper structure around identical memories to support parallel and at-speed testing and reduce area overhead. Figure 9 shows the modified wrapper in which all parts except WBR registers are shared between the identical memories in a group. Using this sharing method, the same test procedure is applied to all identical memories concurrently. Each memory group has only one wrapper parallel input (WPI) port entered to WBR. The four least significant bits of WPI port come from the BIST controller and apply to enable input of each memory. The rest of the signals consist of the address, data and read/write signals which shared between all identical memories. This structure enables controlling the test process of each memory independently, as well as reducing the area overhead and routing complexity. The output signals of the wrapper are WPO port of each memory which applies to the comparator inputs.

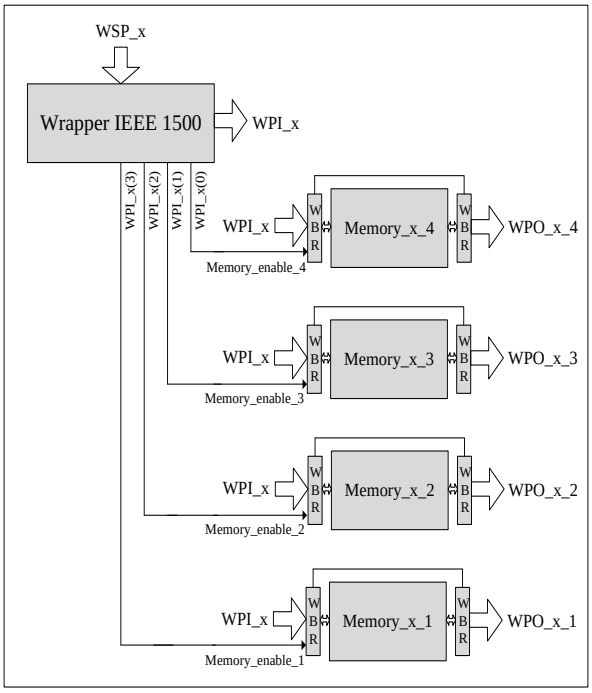


Fig 9. The structure of modified wrapper for each memory group

To enable parallel and at-speed testing in the modified wrapper, we design a wrapper boundary register (WBR) suitable for memory testing. Figure 10 shows the designed WBR which is composed of input cells, output cells, multiplexers and some glue logic around a memory core.

The wrapper serial port (WSP) port has an extra signal for parallel shift called WBR_P_shift and comes from the BIST controller. The three mandatory operation modes of IEEE std 1500 including functional, internal test and external test are shown in table 2. ESC BIST controller initializes all WSP ports concurrently and sets the memories in a suitable operational mode.

Table 2. Operational modes of IEEE std 1500 wrapper

Operational Mode	Mode
Functional	00
Internal Test	01
External Test	10

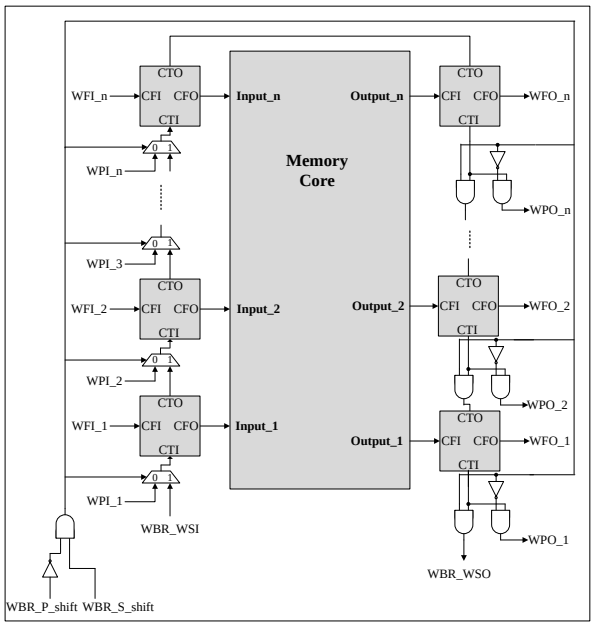


Fig 10. The modified wrapper boundary register

2.6 ESC BIST controller

Figure 11, represents a block diagram of ESC BIST controller. The circuit is composed of three main parts: an FSM machine, *Test Execution* circuit and *End Test/Diagnosis* circuit. The FSM machine controls the whole test and diagnosis procedures.

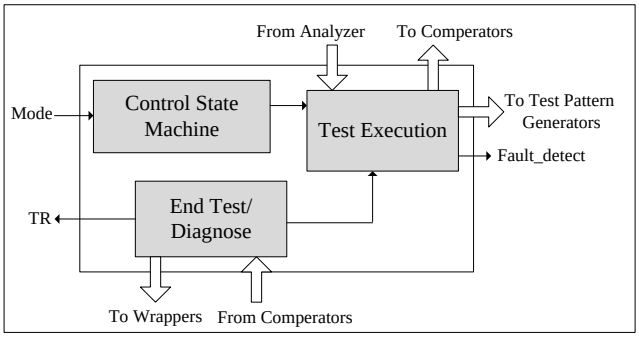


Fig 11.Block diagram of BIST controller (TR: test result)

The FSM machine receives a 2-bit signal named *Mode* to determine the operation as stated by table 2. According to selected modes, the BIST controller initializes WSP port of the wrappers. When internal test mode is selected, *Test Execution* circuit receives data from the algorithm_analyzer. Using the received data, the BIST controller generates required control signals for test pattern generators and comparators. *End Test/Diagnosis* circuit receives comparator outputs after two clock cycles and decides about the end of test process based on the received information.

When no fault is detected in the whole memories, the test process ends and the results shift out through TR signal which has 6 bits. Each bit of this signal belongs to a group of memories and sets to 1 at the end of test process of the group. Therefore, in the case of no faults, all bits of this signal set to 1 concurrently and the test

process ends without the need to a diagnosis procedure. But if each of the memories has a failure, the diagnosis procedure must be executed and so the test process rerun. *End Test/Diagnosis* circuit informs *Test Execution* circuit to rerun test process. Fault_detect signal send the diagnostic information to the corresponding wrappers.

3 ESC BIST functionality

Figure 12 shows the overall functionality of ESC BIST which is composed of three main procedures: initialization of the BIST circuit, test process and diagnosis procedure. In the following sections we describe the functionality of ESC BIST in each of these modes.

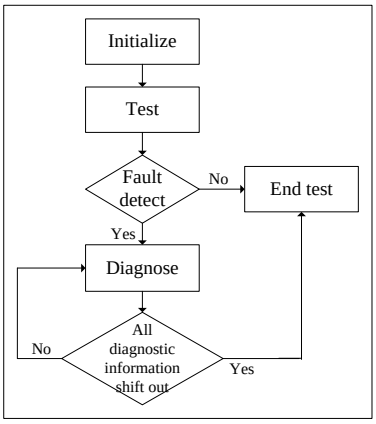


Fig 12. The overall functionality of ESC BIST

When we set the signal mode to 01, the BIST controller configures the wrappers according to the internal test mode [17] and initialization process starts. Figure 13 shows algorithm-register and algorithm-analyzer control signals, shown in figures 3 and 4, in initialization process. The BIST controller sets AR_shift to 1 which enables the algorithm-register to receive the algorithm code from ATE. After 128 clock cycles, AA_enable is set to 1 and the code shifts from algorithm-register to algorithm-analyzer. The analyzer needs M clock cycles for decoding the algorithm which equals to the number of code bits. After M clock cycles, End_analysis signal informs the end of the analysis to the BIST controller. In this step, the BIST controller deasserts AR_shift and AA_enable signals and enables the test pattern generators and the comparators.

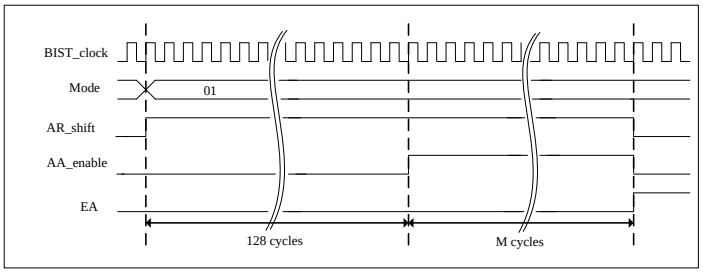


Fig 13. Control signals of the algorithm-register and algorithm-analyzer in the internal test mode

After initialization, test process begins and test pattern generators produce data background and memory address. Also the BIST controller generates read/write and enable signals of the memory. By concatenating all these signals, WPI port of each wrapper is produced which is shown in figure 9 as WPI_ x. Figure 14 represents

WPI fields for 32×64 memories. As shown in figure, WPI consists of 72 bit grouped in 4 fields. These fields include address, data and read/write bits of the memories as well as enable signals of each memory.

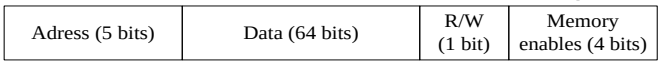


Fig 14. Structure of the WPI port of 32×64 memory group

All WPI ports have the same structure in different address and data length for each memory group. WPI ports are produced concurrently and applied to the wrappers in each clock cycle to implement at-speed testing. The comparators receive the memory responses after two clock cycles from the beginning of the test process. In the case of no fault in memories, the test process ends without any diagnosis procedure. TR signal of the BIST controller reports the end of test process.

When there is a mismatch between memory responses and data backgrounds, the counter of the comparator increments the number of faults of the respective memory. So the fault address and faulty word are saved in the registers of that comparator. By increasing the number of detected faults, address and data word of the last fault is overwritten to the previous ones repeatedly. Therefore, at the end of test process, the existing information is the number of faults and the address and data word of the last detected fault in each memory which are saved in the comparators of each memory group.

Diagnosis procedure starts after finishing the test process. The diagnostic information comes from the comparators and saves in six registers of *End Test/Diagnosis* circuit. At first step of the diagnosis procedure, the information of the last detected fault has to shift out from these registers. Each register saves the information of the four memory of the respective group. Figure 15 shows the structure of these registers in which x represents the number of the memory groups.

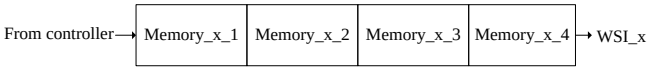


Fig 15. Structure of the diagnostic information register

The BIST controller sends the diagnostic information to the wrapper where it serially shifts from wrapper serial input to wrapper serial output through wrapper data registers. The diagnostic information register of each memory group has different length because of difference in number of data and address bits, so they need different number of clock cycles to shift the contents out. Each of the fields in figure 15, saves the diagnostic information of one memory core which represents in figure 16.

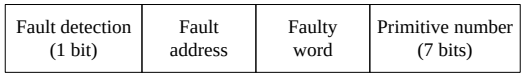


Fig 16. The contents of each field in the diagnostic information register

In this figure, the first field has one bit which is sets to 1 in the case of fault detection. When a fault detects, the second and third fields save the fault address and the faulty word respectively. So by increasing the number of detected fault, the fault address and the faulty word of the last detected fault is overwritten to the previous values. Finally the last field has seven bits and saves the primitive number of the march algorithm related to the last fault.

Figure 17 represents the flow of diagnosis procedure. At the beginning of diagnosis procedure, the diagnostic information of the last detected fault in each memory is shifted out. To shift the diagnostic information of other faults out, the algorithm has to run again, and the diagnosis procedure continues. In this step, the BIST controller compares the number of detected faults in each memory to find the biggest number. This value determines the number of algorithm runs in diagnosis procedure to shift the whole information out. On the other

hand, in order to save time and prevent executing ineffective primitives, the biggest primitive number of the last detected fault in each memory group, is the end point for the next run of that group.

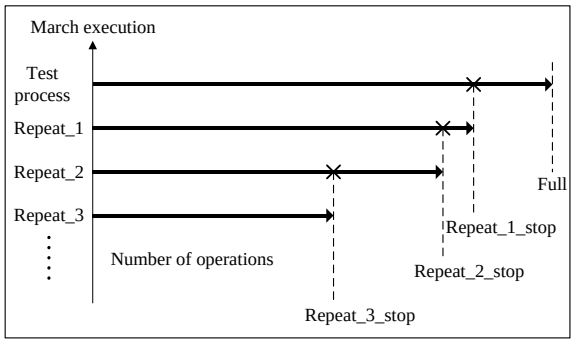


Fig 17. Flow of diagnosis procedure

The BIST controller can independently control the diagnosis procedure of each memory and disable the memories which their faults detected completely. To detect the end of the diagnosis procedure for a memory group, the BIST controller sets the corresponding bit in TR signal. This process continues until the whole bits of TR signal sets to 1. This diagnosis approach guarantees detecting all diagnosable faults with the minimum interface to the ATE.

4 The features of ESC BIST

Important features of the embedded SRAM concurrent BIST (ESC BIST) design involve parallel testing and diagnosis of all memories under test, compatibility with IEEE std 1500, at-speed testing and diagnosis, minimum area overhead compared to the similar schemes and a flexible and scalable architecture.

Parallel testing and diagnosis: we can perform parallel-testing techniques on a device under test and simultaneously test all of the memories. We obtain the test results from the serial output and use them to show whether each memory has passed test or not. Since we perform the test and diagnosis procedure of all memories concurrently, the test and diagnosis time is less than the traditional parallel BIST.

At-speed testing and diagnosis: Serial BIST such as BIST based on IEEE std 1500, needs many clock cycles to shift in test commands, update read and write operations to the memory, capture and shift out memory responses and compare outputs with correct responses [5]. So the serial BIST have a long test time and can't test the memory at-speed. But the proposed BIST structure uses a special design for the IEEE std 1500 wrapper and so can meet the requirements of at-speed and parallel testing. The proposed BIST structure can send a read or write command to the memory for each clock cycle and compare memory outputs in real time.

Compatibility with IEEE std 1500: IEEE std 1500 is a standard for testing embedded cores, however it has some drawbacks in the domain of at-speed and parallel testing. So we propose a modified structure of IEEE std 1500 wrapper to resolve the problems while using this standard as a modular and scalable background.

Minimized area overhead compared to similar schemes: we position each test pattern generator and comparator adjacent to the corresponding memory group to reduce the wire length of the interconnections. To further reducing area overhead, we propose a structure in the form of an algorithm-analyzer which decreases the BIST controller operations to generate test data. On the other hand, this structure produces a scalable structure for any type of march algorithm and any number of memory cores.

Flexible and scalable architecture: The proposed structure easily handles multiple embedded memories. When the number of memories changes, only slight modifications to the controller is necessary. We must add

another test pattern generator and comparator for the memories with different configuration but the test approach does not change.

5 Analysis and results

To evaluate the proposed BIST structure, it is simulated and synthesized for memories with different configurations. Figure 1 represents the used benchmark which has twenty four memories divided in six groups according to their size and configuration. Each group has a self-contained test pattern generator, comparator and modified IEEE std 1500 wrapper which shares one algorithm-register, algorithm-analyzer and BIST controller. The memories under test are high-density SRAMs generated by TSMC 0.18µm CMOS standard cell technology. Also the BIST circuit is synthesized using 0.18µm CMOS standard cell library. Table 3 summarizes the area overhead of the unshared BIST elements in terms of gate count. The first column of the table represents the memory configuration. Here the memory configuration N×W denotes a memory with N words and each word has W bits. The second column shows the area overhead of the memories under test in each group. Also the other columns represent the area overhead of comparators, test pattern generators and wrappers which are the unshared BIST elements.

Table 3 The area overhead of the unshared BIST elements in terms of gate count

Memory Configuration	Memory (mm^2)	Comparator (gate count)	Test pattern Generator (gate count)	IEEE std 1500 Wrapper
32 × 64	0.142 × 4	2413	1650	7526
256 × 64	0.336 × 4	2565	1723	7682
512 × 64	0.419 × 4	2615	1749	7734
128 × 128	0.336 × 4	4251	3674	14286
256 × 128	0.419 × 4	4303	3711	14338
1k × 128	0.308 × 4	4425	3767	14442

Table 4 represents the area overhead of the shared elements in terms of gate count. Similar to the previous table, the first and second columns represent the memory configuration and the area overhead of the memories respectively. Other columns show the area overhead of BIST controller, algorithm register and algorithm analyzer which are the shared elements of the design.

Table 4 The area overhead of the shared elements in terms of gate count

Memory Configuration	Memory (mm^2)	BIST Controller (gate count)	Algorithm Register (gate count)	Algorithm Analyzer (gate count)
32 × 64	0.142 × 4	19380	927	2644
256 × 64	0.336 × 4			
512 × 64	0.419 × 4			
128 × 128	0.336 × 4			
256 × 128	0.419 × 4			
1k × 128	0.308 × 4			

In table 5, we show the ESC BIST area overhead with and without IEEE std 1500 wrappers. The results show the overhead of the IEEE std 1500 wrappers is high, about two times of ESC BIST circuit. Refer to IEEE std 1500, we find that WBR register of the wrapper occupy a large amount of silicon area, because the number of I/O connections of the memory is directly proportional to the number of WBR cells. Thus, in the embedded

SRAMs which have a wide data word (for example 64 or 128 bits), the WBR area overhead is high. However, the proposed BIST structure is independent of the IEEE std 1500 wrappers. We can eliminate the IEEE std 1500 wrappers and implement a slight modification to the proposed design such as using simple multiplexers instead of WBR cells and compare the BIST circuit with the similar schemes.
Table 6 represents the comparison between the area overhead of ESC BIST with some similar schemes. The

Table 5 The area overhead of the scheme in two states

BIST Scheme	Area (gate count)	Memory Core Area (mm^2)	Area Overhead
ESC BIST	126.1k	7.84	32.14%
ESC BIST without IEEE 1500	60.1k	7.84	15.3%

Table 6 Comparison between ESC BIST area overhead with some similar schemes

BIST Scheme	Area Overhead
Hybrid BIST[4]	30.4%
Aitken Design[11]	34.5%
Brains[17]	33.5%
Commercial[3]	41.9%
Huang and Jone Design[9]	45%
Wang ad Xu Design[12]	34%
ESC BIST without IEEE 1500 Wrapper	15.3%
ESC BIST	32.14%

BIST proposed by Aitken, and by Wang and Xu, used the IEEE std 1500 serial interfaces, but not an independent structure as a WBR similar to that defined in standard. But the results in table 6 show even in this case, the proposed ESC BIST which includes IEEE std 1500 wrappers has the lower area overhead.

We also evaluate the test and diagnosis time of ESC BIST structure. In the conventional parallel BIST, the test time is dependent to the number of memory cores. In these structures, only the identical memories can be tested concurrently, whereas ESC BIST controls the test flow of each group of memories independently and is capable of testing all memories at the same time. Therefore the test time of the proposed BIST is constant and equals to the test time of the memory with the most number of data words. Equation (3) shows the number of clock cycles to test each memory:

$$t_T = 128 + P + PN_i + N_i\left(9\log_2^{M_i} + 6\right) \tag{3}$$

Where we define the parameters as follows:
i: the whole number of memory cores
N_i: the number of data words in each memory
M_i: data word length
A_i: address length
P: the number of operation in the march algorithm

In this equation, the first term shows the number of clock cycles needs to shift the algorithm code from ATE to the algorithm-register. To decode the algorithm in the algorithm-analyzer, p clock cycles need which is shown in the second term. Also the third and fourth terms represent the number of clock cycles for executing the march algorithm. Since all memories are tested concurrently and the test time is dominated by the memory with the most number of data words, we can modify the above equation in the form of equation (4) to calculate the total test time:

$$t_{T,total} = Max\left(128 + P + PN_i + N_i\left(9\log_2^{M_i} + 6\right)\right) \tag{4}$$

As an example we calculate the number of clock cycles needs to test the memories of the benchmark shown in figure 1 with march C- algorithm: $t_T = 128+10+10\times1000+1000\left(9\log_2^{128}+6\right)=79138$

Diagnosis time of ESC BIST depends on the most number of data words and detected faults and is composed of two terms, which are the algorithm execution and shifting the diagnostic information out. So we can calculate the number of clock cycles of the diagnosis procedure of each memory from equation (5):

$$t_D = \left(N_i \sum_{1}^{z=n_i-1} P_{i,j,z}\right) + \left(n_i+1\right)\left(n_g\left(1+A_i+7+M_i\right)\right) \tag{5}$$

Where n_i is the number of faults in each memory, n_g is the number of memories in each group and $P_{i,j}$ shows the primitive number in which a fault detects .

In this equation the first and second terms show the clock cycles of algorithm runs and shifting the diagnostic information out respectively. Since the total diagnosis time is dominated by the memory which has the longest time, so we can modify equation (5) in the form of equation (6) to calculate the total diagnosis time:

$$t_{D,total} = Max\left(\left(N_i \sum_{1}^{z=n_i-1} P_{i,j,z}\right) + \left(n_i+1\right)\left(n_g\left(1+A_i+7+M_i\right)\right)\right) \tag{6}$$

The concurrency of the diagnosis procedure in the proposed structure improves the diagnosis time. On the other hand, the used approach guarantees fault detection with the maximum diagnostic capability allowed by the march algorithm. This high fault coverage is achieved because the procedure is at-speed and doesn't stop between the faults to shift the information out.

6 Conclusion

As the number of embedded memory cores in SoCs increases, a low-cost and efficient test and diagnosis methodology for multiple embedded memory cores become more significant. This increase in number, as well as the wide data word of embedded memories causes the penalties of test time and BIST area intolerable. Moreover, the degree of programmability of the BIST circuit to detect new fault types impacts the efficiency of yield improvement. The proposed BIST structure provides a certain degree of programmability using an analyzer to analyze every required march algorithm. Also the novel structure of the IEEE std 1500 wrappers support at-speed and parallel testing of the whole memory instances, while test the memories in a modular and scalable method. Moreover, the sharing method for homogeneous memories improved the area overhead of the design.

References

[1] A. Agarwal, M. Levy, The kill rule for multicore,in proc. IEEE /ACM Design Automation Conf. , 2007 *pp.*750-753. [doi: 10.1145/1278480.12786 6].

[2] S. Borkar, Thousand core chips – a technology perspective, in proc. IEEE/ACM Design Automation Conf., 2007 pp. 746-749.

[3] M. Habibi, H. Poormeidani, A Hierarchical Defect Repair Approach for Hybrid Nano/CMOS Memory Reliability Enhancement, Microelectronics Reliability, 54(2) (2014) 475-484.

[4] Y. Huang, J. Li, A low-cost pipelined BIST scheme for homogeneous RAMs in multicore chips, Asian Test Symposium. ATS '08. 17th, 2008 pp.357-362. [doi: 10.1109/ATS.2008.51].

[5] L. M. Denq, Y. T. Hsing, C. W. Wu, Hybrid BIST scheme for multiple heterogeneous embedded memories, in proc. IEEE

Design & Test of Computers Conf., 2009 pp.64-73. [doi: 10.1109/MDT.20 09.37].
[6] B. Hong Fang, Embedded Memory BIST for Systems-on-a-Chip, MS Thesis, McMaster University, Hamilton, Ontario,Canada, 2003.
[7] B.Vadeau-Dostie, A. Silburt, V. K. Agarwal, A serial interfacing technique for external and built-in self-testing of embedded memories, *IEEE Design & Test of Computers*, 7(2) (1990) 56-64. [doi : 10.1109/ CICC.1989.56808]
[8] W. B. Jone, D. C. Huang, S. C. Wu, K. J. Lee, An efficient bist method for small buffers, in Proc. IEEE VLSI Test Symp. (VTS), 1999 pp. 246-251. [doi: 10.1109/VTEST.1999.766672]
[9] K.L. Cheng et al, Automatic generation of memory built-in self-test cores for system-on-chip, in Proc. Tenth IEEE Asian Test Symp. (ATS*)*, 2001 pp. 91-96. [doi: 10.1109/ATS.2001.990265]
[10] D. C, Huang, W. B. Jone, A parallel built-in self -diagnosis method for embedded memory arrays, *IEEE Trans. On Computer-Aided Design of Integrated Circuits and Systems*, 21(4) (2002) 449-465. [doi: 10.1109/43.002768]
[11]A.Benso et al., "Programmable Built-in Self-Testing of Embedded Ram Clusters in System-on-Chip Architectures",IEEE Comm.Magazine, 41(9) (2003) 90-97.
[12] R.C. Aitken, A Modular Wrapper Enabling High Speed BIST and Repair for Small Wide Memories,in Proc. Int'l Test Conf.(ITC04),IEEE CS Press, 2004 pp. 997-1005. [doi: 10.1109/TEST.2004.1387365].
[13] B. Wang, Q. Xu, Test/Repair area overhead reduction for small embedded memories, in Proc. 15th IEEE Asian Test Symp. (ATS), 2006 pp. 997-1005. [doi: 1 0.1109/ATS.2006.260990]
[14] P. McEvoy, R. Farrell, Built-In Test Engine for Memory, IEEE IC Test Workshop, 2004.
[15] Ad J. Van de Goor, I.B.S. Tlili, A Systematic Method for Modifying March Tests for Bit-Oriented Memories into Tests for Word-Oriented Memories, *Computers, IEEE Transactions on*, 52(10) (2003) 1320-1331. [doi: 10.1109 /TC.2003.1234529]
[16] Ad J. Van de Goor, I.B.S. Tlili, S. Hamdioui, Converting March Tests for Bit-Oriented Memories into Tests for Word-Oriented Memories, Memory Technology, Design and Testing. Proceedings. International Workshop on, 1998 pp. 46-52. [doi: 10.1109/MTDT.1998.705945]
[17]IEEE Computer Society, IEEE Standard Testability Method for Embedded Core-based Integrated Circuits- IEEE Std 1500™-2005, IEEE, 2005. [doi: **:** 10.1109/IEEESTD.2005.96465]
[18]C. Cheng et al, BRAINS: A BIST Compiler for Embedde Memories, Defect and Fault Tolerance in VLSI Systems, Proceedings. IEEE International Symposium, 2000 pp. 299 – 307.