International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:4, No:1, 2010

A New Efficient Scalable BIST Full Adder using Polymorphic Gates

M. Mashayekhi[1], H. H. Ardakani[2] and A. Omidian[3]
[1,2,3]Islamic Azad University, Karaj Branch, Karaj, Iran
{mashayekhi, hassan.hatefi, omidian}@kiau.ac.ir

***Abstract*—**Among various testing methodologies, Built-in Self-Test (BIST) is recognized as a low cost, effective paradigm. Also, full adders are one of the basic building blocks of most arithmetic circuits in all processing units. In this paper, an optimized testable 2-bit full adder as a test building block is proposed. Then, a BIST procedure is introduced to scale up the building block and to generate a self testable n-bit full adders. The target design can achieve 100% fault coverage using insignificant amount of hardware redundancy. Moreover, Overall test time is reduced by utilizing polymorphic gates and also by testing full adder building blocks in parallel.

***Keywords*—**BIST, Full Adder, Polymorphic Gate

I. Introduction

VLSI has had a dramatic impact on the growth of digital technology which has increased the complexity of the circuits. There are, however, potential problems which may retard the effective use and growth of future VLSI technology. Among these is the problem of circuit testing, which becomes increasingly difficult as the scale of integration grows. Because of deep submicron technology that allows hundreds of millions of transistors to be integrated on the same chip, conventional testing approaches are often ineffective and insufficient for VLSI circuits. On the other hand, the traditional Automatic Test Equipment (ATE) based solution is becoming increasingly expensive and inaccurate [1]. For any such alternative, the following goals are desirable: high and easily verifiable fault coverage, minimum test pattern generation, minimum performance degradation, at-speed testing, short testing time, and reasonable hardware overhead.

Built-In Self-Test (BIST) provides a feasible solution to partially meet the above demands. First, BIST significantly reduces off-chip communication to overcome the bottleneck caused by the limited input/output access. Further, it eliminates much of the test pattern generation and simulation process. Testing time can be shortened by testing multiple units simultaneously through test scheduling. Hardware overhead can be minimized by careful design and through the sharing of test hardware. It also reduces the reliance on traditional, high-cost, full-feature testers [1].

Classical BIST approaches suffer some problems such as, inducing additional delay to the circuitry and requiring a relatively long test application time, increasing hardware complexity and also incomplete fault coverage [2]. In particular, one major problem with the classical BIST implementation is due to the fact that the Test Pattern Generator (TPG) is implemented by Linear Feedback Signature Register (LFSR) that pseudo randomly generates test patterns and often do not guarantee a sufficiently high fault coverage (especially in the case of large and complex designs). It also demands very long test application times. It is not uncommon to have a pseudorandom test sequence that is more than 10 times longer than the deterministic test sequence with similar efficiency [3]. Moreover, many faults will never be detected with pseudorandom test vectors alone. Therefore, several proposals have been made in [4-6] to deal with the mentioned problems. Nevertheless, they cannot thoroughly handle different aspects related to the test cost, like test length, area overhead and tester memory requirements. Also, other attempts for improving the classical BIST scheme have been proposed. For example, in [7] a hybrid BIST is proposed that complement pseudorandom test patterns with deterministic test patterns, applied from the on-chip memory or, in special situations, from the ATE. As another example the fault coverage can be increased by modifying the Circuit under Test (CUT) by either inserting the test points [8] or by redesigning the CUT itself [9]. The drawback of these techniques is that they generally add additional logic to the circuitry that can degrade the system performance.

This work addresses the problem of BIST test time for high fault coverage by targeting test concurrency and test pattern minimization in full adders. The objective is to introduce a block of full adder that optimized for testing aims and to scale it up to generate n-bit BIST full adders. The optimized block has been enriched by polymorphic gates to minimize test pattern and then test time to achieve 100% fault coverage. All blocks can share some testing resources such as TPG and signature memory and therefore hardware redundancy will be negligible. They carry out the testing procedure concurrently to further reduction in testing time. Finally, the target design has been improved in terms of test pattern length, hardware complexity, tester memory requirements and also testing time.

The rest of this paper is organized as follows: optimized testable building block is proposed in Section I. Then, in Section III, our n-bit BIST full adder is introduced. Finally,

Morteza Mashayekhi is with the Computer Science Department, Karaj Islamic Azad University, Karaj, Iran (phone: +989126441807; e-mail: mashayekhi@kiau.ac.ir).
Hassan Hatefi Ardakani is with the Computer Science Department, Karaj Islamic Azad University, Karaj, Iran (phone: +989125523014; e-mail: hassan.hatefi@kiau.ac.ir).
Alireza Omidian is with the Computer Science Department, Karaj Islamic Azad University, Karaj, Iran (phone: +989122024124; e-mail: omidian@kiau.ac.ir).

conclusions and future directions are presented in Section IV.

II. Optimized Testable Building Block

Our method is based on a block of adder that optimized for testing aims and can be scaled up to produce larger adders. Test optimization can be done using specific gates called polymorphic gates. In this section, we describe how to generate optimized building block for a full adder.

A. Polymorphic Gates

In fact, polymorphic gates are multifunctional circuits that the change of their behavior comes from modifications in the characteristics of components involved in their circuit in response to controls such as temperature, power supply voltage, light, etc. Stoica et al [10,11] designed and implemented some basic polymorphic gates and have done some self-repair experiments in extreme environments. Some examples of the existing polymorphic gates are shown in Table I. For example NAND/NOR consists of six transistors and operates as NAND in mode 1 when Vdd is 3.3V and as NOR in mode 2 when Vdd is 1.8V. The circuit was fabricated in a 0.5-micron CMOS technology. It is stable for ±10% variations of Vdd and for temperatures in the range of 20°C to 200°C.

Table I
Examples of Existing Polymorphic Gates

Gate	Control Values	Control Line	# of transistor
AND/OR	27/125◦C	Temperature	6
AND/OR/XOR	3.3/0.0/1.5	ext. voltage	10
AND/OR	3.3/0.0V	ext. voltage	6
AND/OR	1.2/3.3V	Vdd	8
NAND/NOR	3.3/1.8V	Vdd	6
NAND/NOR/ NXOR/AND	0/0.9/1.1/ 1.8V	ext. voltage	11

Research papers indicate various areas in which polymorphic gates could be utilized. One of them is testing and diagnosing of electronic circuits [12-14]. By using polymorphic gates, we provide multi-way functionality for a circuit. Multi-way functionality enables the circuit to work in multiple modes of operation with probably different outputs in each mode. Since each mode can detect some faults, involving several modes in testing leads to higher fault detection than using only one mode like traditional circuits. Consequently, polymorphic gates and then multi-way functionality have the ability to improve testing of digital circuits.

B. Test Pattern Minimization

The most important feature of our building block is that it is minimized in term of test pattern length. Shortening in test pattern length is due directly to use of multi-way functionality and also to entail several configurations in testing. Consider circuit C that contains m polymorphic gates: $G_1 \ldots G_m$. A configuration is the set of doubles in the form of (G_i, M_j) that means polymorphic gate G_i is configured in mode j. For example, $\{(G_1, M_1), (G_2, M_1) \ldots (G_m, M_1)\}$ is the configuration in which all polymorphic gates operate in their first modes. By setting the polymorphic gates of circuit C in different modes, it can be put into several configurations. Let $n(G_i)$ denotes the number of modes in polymorphic gate G_i. We can calculate the number of possible configurations for circuit C as follows:

$$N = \prod_{i=1}^{m} n(G_i)$$

Let $R = \{C_1, C_2 \ldots C_N\}$ is the set of all possible configurations which C can operate as them. For each input vector v that is applied to the circuit when it configured as $C_i \in R$ and by comparing the output vector with the correct pattern, some of circuit faults can be detected. Therefore, if several configurations are involved in testing of C, the number of faults which can be detected by v will increase. Assume that we select some configurations from R and put them into set r ($r \subseteq R$). Configurations that belong to r are involved in testing of C and affect fault coverage. Let $F_v(\{c\})$ is the set of faults that can be detected by v in configuration $c \in R$. We can calculate $F_v(r)$, the set of all detectable faults on subset r, as follows:

$$F_v(r) = \bigcup_{c \in r} F_v(\{c\})$$

Examining a circuit in numerous configurations heighten the cost of testing. Therefore, we must find a set with few configurations that is satisfactory to achieve 100% fault coverage using a short test pattern. The problem is to find the best subset of R with k member(s) that can detect all stuck at faults with the minimum test pattern length. The problem is NP, but we can search its space state using an evolutionary approach to acquire a good k-member subset that satisfies the above requirements. Details of this algorithm are beyond the scope of this paper and we consider its results only. In this method testing time is directly proportional to k and test pattern length. Therefore, small subsets (with shorter k) can further reduce testing time of a circuit.

C. Target Building Block

Our building block is a 2-bit full adder containing several polymorphic gates to enhance its testability features as illustrated in Figure 1. It has been made up of two different types of polymorphic gates: AND/OR/XOR and NAND/NOR/NXOR/AND that are controlled by external voltages. By changing the control voltage of the polymorphic gates according to Table I, the circuit can switch between its different configurations. All of these configurations may be used during testing phase while one of them has the functionality of an ordinary 2 bit full adder, and they have been depicted in Table II. The set of all configurations is searched using the evolutionary algorithm mentioned in Section II.B to find a good subset that achieves 100% fault coverage through a minimized test pattern. The size of the target subset is considered to be three. The results are shown in Table II, where all three selected configurations of polymorphic gates are described.

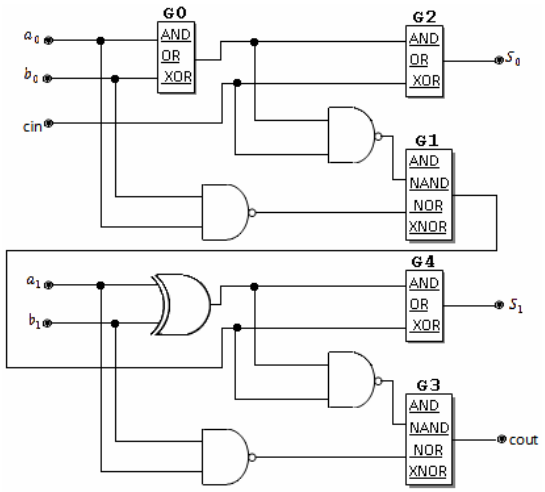

Fig. 1 2-bit full adder as the building block of our BIST design

The proposed building block has a distinct advantage in terms of test quality over traditional designs. Only two test vectors {V4, V27} among 32 possible ones will be adequate for detecting all stuck at faults on it. Therefore, instead of using a 5-bit LFSR to generate all possible input vectors, its TPG can be constructed using only one flip flop.

TABLE II
DETAILS OF CONFIGURATIONS AND OPERATION MODES OF POLYMORPHIC GATES IN EACH CONFIGURATION

Configurations		G0	G1	G2	G3	G4
Standard 2-bit FA		XOR	NAND	XOR	NAND	XOR
Selected Configuration for Test	1	XOR	AND	AND	NOR	AND
	2	XOR	XNOR	OR	AND	AND
	3	OR	NOR	OR	AND	OR

III. N-BIT BIST FULL ADDER

A typical BIST architecture consists of a test pattern generator (TPG), a test response analyzer (TRA), and a BIST control unit (BCU), all implemented on a chip. Examples of TPG are a ROM with stored patterns, a counter, or an LFSR. A typical TRA is a comparator with stored responses or an LFSR used as a signature analyzer. And finally a BCU is needed to activate test and analyze responses.

Our BIST architecture to build a self testable n-bit full adder is depicted in Figure 2. In this architecture we use 2-bit full adder that was introduced in Section II.C as the building block of the BIST architecture. The connections between the building blocks are as the same as the conventional n-bit carry propagate adder that are not shown in this figure. The building block inputs are either primary inputs (PIs) or test vectors that are supplied with a block of multiplexers controlled by the BCU over the states of the circuit i.e. testing mode or normal mode.

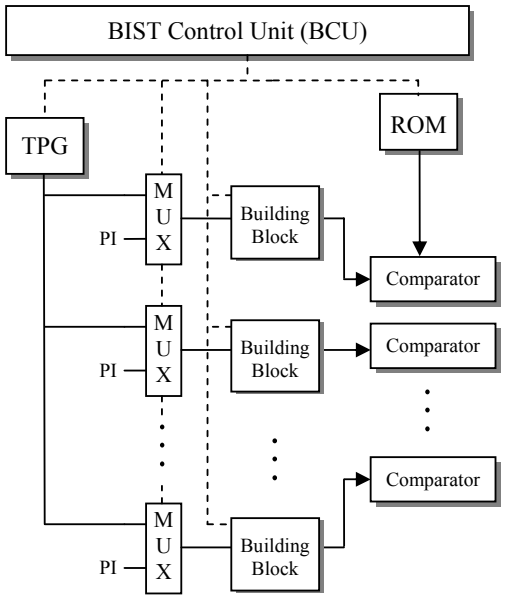

Fig. 2 Our BIST architecture

As mentioned in Section II.C our TPG can be constructed using only one flip flop that its complexity is much smaller than that of a usual 5 bit LFSR. The correct output pattern of the circuit in three different configurations is stored in a ROM. For each configuration, the correct output pattern (S_0, S_1 and C_{out}) of the building block for two vectors {V4, V27} must be saved in the ROM. Hence the total memory space will be 18 bits. The comparators are an array of XOR gates to compare the full adder outputs with the correct pattern stored in the ROM. The BCU issues control signals illustrated by dashed line in the figure to different parts of the circuit like the TPG, blocks of multiplexers, the ROM and building blocks. It has the ability to put the circuit into either normal or testing mode, to generate test patterns, to carry out them into building blocks, to switch building blocks between their configurations and to retrieve correct test patterns from the ROM.

Testing features become more important especially when the adder size increases. Hardware complexity and test time of our method for larger adders are shown in Table III. Since test pattern, correct signature and control signals for all building blocks are similar, hardware complexity of the TPG, the BCU and the ROM remains unchanged although the adder size has increased. Because the test pattern length and also number of circuit configurations are fixed, the test time also remains fixed. When the BCU activates testing mode, all building blocks will be checked using two test vectors on three configurations. If testing of a vector for each configuration takes one clock, the overall testing time will be 6 clocks, regardless of the adder size. Hence the proposed method is C-testable. However, like other BIST architectures, the number of comparators and multiplexer blocks increases proportionally.

TABLE III
TESTING FEATURES OF OUR METHOD FOR LARGER ADDERS

Size of Adder	TPG	#of MUX Blocks	TRA		Test Time (clock)
	#of FFs		ROM Space (bit)	#of Comparators	
2	1	1	18	1	6
4	1	2	18	2	6
8	1	4	18	4	6
16	1	8	18	8	6

IV. CONCLUSIONS

In this paper we proposed a 2-bit testable full adder building block which enriched by polymorphic gates to minimize test pattern and then test time to achieve 100% fault coverage. Then we scaled up this block to make an n-bit BIST full adder that had been improved in terms of test pattern length, hardware complexity, tester memory requirement and also testing time. It has been shown that hardware complexity and also testing time have no change when the full adder size increases. Using polymorphic gates and also test concurrency helped us to reach these enhanced results.

In future work, we will extend the proposed method to make more complicated circuits self testable using evolutionary approaches. We will also deal with the problems of utilizing our method to make low power version of BIST.

REFERENCES

[1] M.L. Bushnell and V.D. Agraval, Essentials of electronic testing for digital, memory, and mixed-signal VLSI circuits, Kluwer, Boston, 2002.
[2] G. Jervan, E. Orasson, H. Kruus, and R. Ubar, "Hybrid BIST optimization using reseeding and test set compaction", in Proc. 10th IEEE Euromicro Conference on Digital System Design Architectures, Methods and Tools, Lübeck, 2007, pp. 596 – 603.
[3] N.K. Jha, S. Gupta, Testing of digital systems, Cambridge University Press, Cambridge, 2003.
[4] S. Hellebrand, H.-J. Wunderlich, and A. Hertwig, "Mixed-mode BIST using embedded processors", Journal of Electronic Testing: Theory and Applications, 12 (1-2), pp. 127–138, 1998.
[5] M. Sugihara, H. Date, and H. Yasuura, "Analysis and minimization of test time in a combined BIST and external test approach", in Proc. 5th IEEE Design, Automation and Test in Europe, France, 2000, pp 134–140.
[6] N. Zacharia, J. Rajski, and J. Tyzer, "Decompression of test data using variable-length seed LFSRs", in Proc. 13th IEEE VLSI Test Symposium, Boston, 1995, pp. 426–433.
[7] E.J. Marinissen, Y. Zorian, "Challenges in testing core-based system ICs", IEEE Communications Magazine, 37 (6), pp. 104–109, 1999.
[8] N.A. Touba, E.J. McCluskey, "Test point insertion based on path tracing", in Proc. 14th IEEE VLSI Test Symposium, California, 1996, pp. 2–8.
[9] Z. Zhao, B. Pouya, N.A. Touba, BETSY: "Synthesizing circuits for a specified BIST", in Proc. IEEE International Test Conference, Washington, 1998, pp. 144–153.
[10] A.Stoica, R.S.Zebulum, and D.Keymeulen, "Polymorphic electronics", Proceedings 4th Springer International Conference on Evolvable Systems: From Biology to Hardware, Berlin, 2001, pp. 291–302.
[11] R.S.Zebulumand, A.Stoica, "Four-Function logic gate controlled by analog voltage", NASATechBriefs, vol.30,Mar 2006.
[12] H. Hatefi, M. Mashayekhi, "A self-testing method for combinational circuits using polymorphic gates", in Proc. 1st IEEE Asia Symposium on Quality Electronic Design, Kuala Lumpur, 2009, pp. 178 - 182.
[13] R. Ruzicka, L. Sekanina, P. Roman "Physical demonstration of polymorphic self-checking circuits", in Proc. 14th IEEE On-Line Testing, Rhodes, 2008, pp. 31-36.
[14] L. Sekanina, L. Stareček, Z. Kotásek, Z. Gajda, "Polymorphic gates in design and test of digital circuits", International Journal of Unconventional Computing, 4(2), pp. 125-142,2008.