An Effective BIST Architecture for Fast Multiplier Cores

A. Paschalis[1] D. Gizopoulos[2] N. Kranitis[1] M. Psarakis[1] Y. Zorian[3]

[1] Institute of Informatics & Telecommunications, NCSR "Demokritos", Athens, GREECE
{paschali|nkran|mpsarak}@iit.demokritos.gr

[2] 4PLUS Technologies, Athens, GREECE
dgizop@4plus.com

[3] LogicVision, San Jose, CA, USA
zorian@lvision.com

Abstract

Wallace tree summation in conjunction with Booth encoding are well known techniques to design fast multiplier cores widely used as embedded cores in the design of complex systems on chip. Testing of such multiplier cores deeply embedded in complex ICs requires the utilization of a BIST architecture that can be easily synthesized along with the multiplier by the module generator.

In this paper we introduce an effective BIST architecture for fast multipliers that completely complies with this requirement. The algorithmic BIST patterns that this architecture generates guarantee a fault coverage higher than 99%. The required Test Pattern Generator consists of a simple fixed-size binary counter, independent of the multiplier size. Accumulator-based compaction is adopted since multipliers and adders co-exist in most datapath architectures.

1. Introduction

Fast multipliers are widely used as embedded cores in both general purpose datapath structures and specialized digital signal processors. In order to accelerate multiplication, first, we reduce the number of partial products by using Booth encoding [1,2], and, then, we sum up these partial products accordingly by using Wallace tree summation [3] and carry look-ahead addition.

The low controllability and observability of fast multiplier cores deeply embedded in complex ICs impose serious testability problems. External testing of fast multiplier cores using scan techniques is not trivial at speed, a demand in today's high speed ICs. Furthermore, scan design is not always cost effective in high complexity designs with many deeply embedded cores. In such cases, Built-In Self Test (BIST), a method that puts the tester on the chip, provides an excellent solution.

The use of effective BIST architectures for embedded fast multiplier cores, as well as, for other multiplier cores [4,5] or RAMs [6], ROMs [7], FIFOs [8] and ALUs [9], is the best solution. It permits at-speed testing, provides very high fault coverage and drives down the testing costs for the overall IC.

An effective BIST architecture for fast multiplier cores must satisfy the following requirements:

- It must not apply Design For Testability (DFT) modifications to the multiplier's structure to avoid performance degradation in carefully optimized designs.
- It must guarantee very high fault coverage without requiring time-consuming iterative fault simulations.
- It must consist of a small number of regular test vectors independent of the multiplier size.
- The BIST hardware must be easily synthesizable and must impose little hardware and delay overhead in the multiplier design.

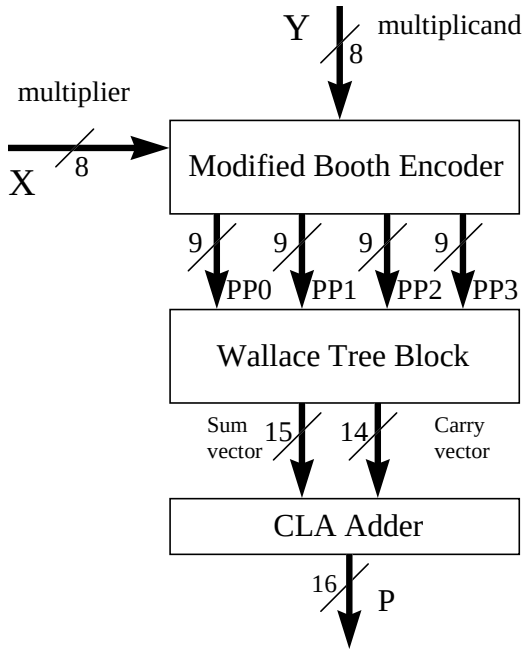

Figure 1: An 8x8-bit fast multiplier core

Open literature currently lacks BIST architectures for fast multiplier cores. In this paper, we present a generic BIST architecture for fast multiplier cores that complies with the requirements just listed. The BIST algorithm introduced in this paper provides fault coverage higher than 99% and is generated by a fixed size test set, regardless of the multiplier size. This means that both the cost of the BIST test pattern generation hardware and the test application time are constant. We use as test pattern generator a fixed width counter that needs very little design effort to be synthesized along with an existing multiplier design. The best known output data compaction techniques (accumulator-based compaction [10,11] and signature analysis [12]) provide excellent error coverage. In the case that an accumulator already exists at the multiplier outputs, accumulator-based compaction with single or multiple rotate carry requires negligible extra hardware and hence it is more efficient with respect to hardware overhead. Multipliers are accompanied by an accumulator in the majority of datapath architectures. If the multiplier is not accompanied by an accumulator there are two excellent alternative solutions. One solution is to add an accumulator with area optimized adder cells and exact pitch matching. This requires little design effort due to accumulator's modularity and imposes small hardware overhead. The other solution is to add a classical multiple input signature register (MISR). In this paper, we consider both alternatives.

2. Fast multiplier core architecture

The fast multiplier core architecture considered in this paper consists of three blocks (see Fig. 1) :

- the *Booth Encoder* for the multiplier recoding and the formation of the partial products,
- the *Wallace Tree Block* for the summation of the partial products which leads to two vectors, and
- the *Carry Look-Ahead (CLA) Adder* which adds the two vectors to obtain the final multiplication product.

The Booth encoder recodes the N_x-bit two's complement multiplier operand and generates $\lceil N_x/2 \rceil$ N_{y+1}-bit partial products PPi's (i = 0,1,2,.., $\lceil N_x/2 \rceil$-1) [5]. For example, the Booth encoder of an 8x8-bit fast multiplier core generates four 9-bit partial products (PP0, PP1, PP2 and PP3). This reduction by half of the number of the partial products is the main purpose of the Booth encoder in the multiplier architecture leading to speeding up the multiplication operation. Each partial product PPi is shifted by 2 bit positions with respect to its neighbours. The various required multiples can be easily obtained by a simple shift of the multiplicand. Negative multiples (in 2's complement form) can be obtained by inverting every bit of the multiplicand and adding "1" at the least significant bit position of the partial product. This is performed by adding the sign recoding signal (shown as S in Fig. 2) at the least significant bit position of the corresponding partial product. The Booth encoder consists of an array of two different cells. The *r-cells* which perform the recoding function and the *pp-cells* which perform the generation of the partial products PPi's to be added by the Wallace tree block. Since the Booth recoding is used for two's complement binary number representation, and the Wallace tree architecture adds the partial products in parallel, sign extension should take place. For proper addition, the sign bit of each PP_i must be extended to the sign bit position of the $\lceil N_x/2 \rceil$-st partial product which is the most significant one. We adopt the "add one method" for sign extension [13]. This method dictates the following simple operations:

- to invert every sign bit of each PPi,
- to add "1" in the sign bit position of the PP0,
- to add "1" in the bit position after the sign bit position of the PPi (i = 0,1,2,.., $\lceil N_x/2 \rceil$-1),

In Fig. 2 we illustrate Booth recoding and sign extension of an 8x8-bit fast multiplier core.

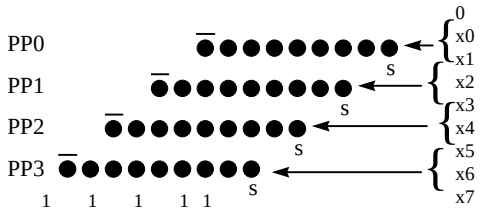

Figure 2: Dot diagram of an 8x8-bit fast mult. core

Booth encoding reduces the number of partial products by half and Wallace tree increases the speed of summing the partial products by means of increased parallelism [3]. The Wallace tree performs the addition of the partial products in each bit position (column) in parallel and independent of other columns. This tree structure which consists of *full adder cells* and *half adder cells*, reduces the partial product summands of each column to two vectors: a sum vector and a carry vector. These vectors represent the multiplication result in a redundant form. They must be added together by the succeeding CLA adder to obtain the final multiplication result in a non-redundant form.

The CLA adder is organized in a hierarchical approach in order to speed up carry propagation leading to fast addition. In the first level, 4-bit adder blocks are formed. First-level group carry generate and carry propagate signals are computed inside each 4-bit adder block. Following the hierarchical approach, 16-bit adder blocks are formed using four 4-bit adder blocks, along with second level carry generate and carry propagate signals, and so on. Since some of the adder inputs are always stable to "0", simplification has taken place to achieve irredundant implementations.

3. Test Strategy and Fault Model

Our testing strategy is based on pseudo-exhaustive testing for the cell-based parts of the fast multiplier core (i.e., the Booth encoder and the Wallace tree block). This means that every cell (r-cell, pp-cell, full adder cell and half adder cell) receives, during BIST mode, all input combinations that can appear during normal mode.

We assume that only one cell can be faulty at a time and that only combinational faults can happen. Adopting this fault model we cover all detectable combinational faults. The set of faults included in the adopted fault model is a superset of all single stuck-at faults. It also includes any other type of single or multiple fault, that may happen in the single faulty cell and change its function to a different combinational one.

The set of input combinations that each type of cells receives during BIST is given in the following :

r-cells: They receive all possible 8 input combinations of their 3 inputs. The first r-cell has one input stable to "0". This r-cell is reduced to a simpler irredundant 2-input cell and receives all possible 4 input combinations.

pp-cells: They receive all 24 different input combinations of their 5 inputs that can appear during normal operation. In details, the 3 inputs from the corresponding r-cell receive 6 input combinations, which represent a recoded digit (+0, -0, +1, -1, +2, -2), and the 2 inputs from the pair of bits of the non-recoded operand receive all possible 4 input combinations. There are some pp-cells which receive less input combinations during normal operation. These pp-cells are reduced to simpler irredundant pp-cells and receive during BIST mode all input combinations appearing during normal operation.

adders: The full adder cells receive all possible 8 input combinations of their 3 inputs. The half adder cells receive all possible 4 input combinations of their 2 inputs. There are some full adders with one input stable to "1". These full adder cells are reduced to simpler irredundant 2-input cells and receives all possible 4 input combinations.

Apart from this, for the non cell-based part of the fast multiplier core (i.e., the CLA adder) we adopt the classical single stuck-at fault model.

4. Proposed BIST Architecture

Usually fast multiplier core designs have optimized layouts since they are critical modules of a circuit in terms of both circuit area and speed. Since Design-for-Testability (DFT) modifications inside a multiplier structure may add extra hardware overhead and lead to performance degradation, it is preferable to avoid such modifications whenever possible. The proposed BIST architecture does not require modification of the fast multiplier structure since it uses a test pattern generator and an output data compactor to be augmented on the periphery of the multiplier, as shown in Fig. 3. The proposed BIST structure makes a complex BIST controller unnecessary.

Fig. 3 also shows the input data registers for the two operands X and Y. The architecture uses two sets of multiplexers at the top and left sides of the fast multiplier core to select between normal and BIST multiplier inputs. The propagation delay of the fast multiplier core affects the performance of the system. To avoid performance degradation we can place the multiplexers before the input registers, provided that the multiplier inputs come from faster modules. As a result, the BIST architecture imposes virtually *no delay overhead.*

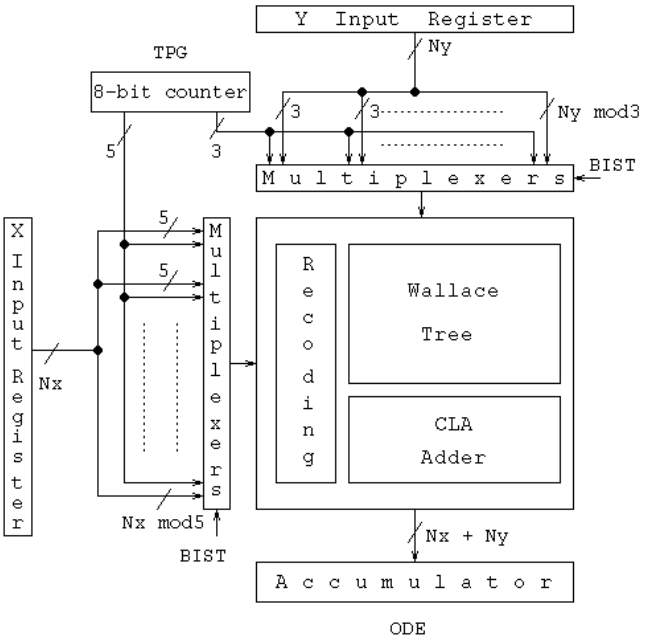

Figure 3: BIST architecture

TEST PATTERN GENERATION

The proposed Test Pattern Generator for an N_x x N_y bit fast multiplier core is an 8-bit counter which goes through all its 256 cycles. During BIST, through the first set of multiplexers, the multiplier repeatedly receives five counter outputs as X inputs (which are the inputs of the Booth encoder) while, through the second set of multiplexers, it repeatedly receives the remaining three counter outputs as Y inputs, as shown in Fig. 3.

During the application of the 256 test vectors produced by the counter, all the cells of the Booth encoder and the Wallace tree block (i.e., r-cells, pp-cells, full adder cells and half adder cells) receive all their input combinations that can appear during normal operation. Besides, during the application of these test vectors the CLA adder has very high fault coverage with respect to single stuck-at fault model.

Apart from this, any faulty cell output of the Booth encoder and the Wallace tree block is propagated towards primary outputs.

OUTPUT DATA COMPACTION

Accumulator-based output data compaction (ABC) with single rotate carry [11] and multiple rotate carry [5] has been adopted in our BIST architecture considering that fast multiplier are usually accompanied by accumulators in the majority of datapath architectures.

If the multiplier is not accompanied by an accumulator we add either an accumulator with area optimized adder cells and exact pitch matching, or a classical multiple input signature register (MISR).

The effectiveness of our BIST architecture has been measured by fault simulation. We extensively simulated the proposed BIST architecture for a 16x16 bit fast multiplier core using the `Verifault` stuck-at fault simulator of the `Cadence Design Framework`. We explored all different compaction schemes mentioned below. Table 1 summarizes the fault simulation results.

Table 1: Fault Simulation Results

compaction scheme	fault coverage
before compaction	99.8%
ABC with single rotate carry	99.1%
ABC with multiple rotate carry	99.7%
MISR	99.8%

We analyze the results of Table 1 as follows:

- The fault coverage of the proposed BIST architecture is 99.8% before compaction. Undetected faults appear only in the CLA adder.
- ABC with single rotate carry [11] satisfies the fault coverage target of 99% with aliasing of 0.7%.
- ABC with multiple rotate carry [5] over-satisfies the fault coverage target of 99% with negligible aliasing of 0.1%.
- MISR also over-satisfies the fault coverage target of 99% with zero aliasing.

Therefore, all compaction schemes satisfy the target of 99% and they can be used in our BIST architecture for fast multiplier cores.

HARDWARE AND DELAY OVERHEAD

A rough estimation of the hardware overhead imposed by our BIST architecture is given below in gate equivalents for the case of the 16x16 bit fast multiplier core.

Table 2: Hardware and Delay Overhead

Acc. Exists ?	ABC single rot. carry	ABC mult. rot. carry	MISR
yes	3.8%	4.9%	-
no	12.7%	13.9%	8.8%

From Table 2 we conclude that:

- when the accumulator exists, the hardware overhead is very little (less than 5%) for both cases of ABC with single and multiple rotate carry.
- when the accumulator does not exist, the hardware overhead imposed for the cases of ABC with single and multiple rotate carry is 12.7% and 13.9%, respectively. This hardware overhead can be reduced using an accumulator with area optimized adder cells and exact pitch matching. Alternatively, a classical MISR should be used that imposes little hardware overhead (8.8 %) and achieves zero aliasing.

In all cases the hardware overhead becomes negligible for fast multiplier cores of larger size.

The delay hardware overhead imposed by our BIST architecture is no more than the smallest possible delay overhead of any off-line BIST architecture that has a single multiplexing stage. Moreover, when the multiplier inputs come from faster modules, we can place the multiplexers (Fig. 3) before the input registers that imposes virtually *no delay overhead*.

The small delay overhead introduced in the accumulator [5] when the multiple rotate carry compaction scheme is utilized does not affect the system performance. This is because the multiplier's speed, which in any case is less than the adder's speed, determines the clock period of the data path in which both the multiplier and adder participate. Since we introduce no DFT modification to the multiplier structure, the proposed architecture does not significantly affect the data path's performance.

5. Conclusions

We have introduced an effective BIST architecture for fast multiplier cores. An 8-bit counter test pattern generator is used along with either accumulator-based output data compaction or signature analysis. No Design-for-Testability modifications to the multiplier are required and the architecture is totally applied on the periphery of the multiplier structure therefore causing no internal performance degradation. The BIST architecture is generic and can be applied to any tree-like (Wallace) multiplier, with or without Booth encoder to form the partial products and with other adder structure instead of CLA adder to obtain the final product.

References

1. A. D. Booth, *"A Signed Binary Multiplication Technique"*, A. J. Mech. Appl. Math. 4, pp. 260-264, April 1951.
2. L. P. Rubinfield, *"A Proof of the Modified Booth Algorithm for Multiplication"*, IEEE Trans. on Computers, vol. C-24, pp. 1014-1015, October 1975.
3. C. S. Wallace, *"A Suggestion for a Fast Multiplier"*, IEEE Transactions on Computers, vol. EC-13, pp. 14-17, February 1964.
4. D. Gizopoulos, A. Paschalis and Y. Zorian, *"An Effective BIST Scheme for Carry-Save and Carry-Propagate Array Multipliers"*, in Proc. 4th IEEE Asian Test Symposium, pp 286-292, 1995.
5. D. Gizopoulos, A. Paschalis and Y. Zorian, *"Effective Build-In Self-Test for Booth Multipliers"*, IEEE Design & Test of Computers, vol. 15, no. 3, pp 105-111, July 1998.
6. B. Nadeau-Dostie, A. Silburt and V.K. Agarwal, *"Serial Interfacing for Embedded Memory Testing"*, IEEE Design & Test of Computers, vol. 7, no. 2, pp. 52-63, April 1990.
7. Y. Zorian and A. Ivanov, *"An Effective BIST Scheme for ROMs"*, IEEE Trans. on Computers, vol. 41, no. 5, pp. 646-653, May 1992.
8. Y. Zorian, A.J. Van de Goor and I. Schanstra, *"An Effective BIST Scheme for Ring-Address Type FIFOs"*, in Proc. IEEE International Test Conference, 1994.
9. D. Gizopoulos, A. Paschalis, Y. Zorian and M. Psarakis, *"An Effective BIST Scheme for Arithmetic Logic Units"*, in Proc. IEEE International Test Conference, pp 868-877, 1997.
10. A. Ivanov and Y. Zorian, *"Count-Based BIST Compaction Schemes and Aliasing Probability Computation"*, IEEE Trans. on Computer-Aided Design, vol. 11, no. 6, pp.768-777, June 1992.
11. J. Rajski and J. Tyszer, *"Test Responses Compaction in Accumulators with Rotate Carry Adders"*, IEEE Trans. on Computer-Aided Design, vol. 12, no. 4, pp. 531-539, April 1993.
12. R. A. Frohwerk, *"Signature Analysis: A New Digital Field Service Method"*, Hewlett-Packard J., pp. 2-8, May 1977.
13. H. Yamauchi et al. "10ns 8x8 Multiplier LSI Using Super Self-Aligned Process Technology", IEEE Journal of Solid-State Circuits, , vol. SC-18, pp. 204-210, April 1983.