A Comparative Study of BIST PRBS Pattern Generation for Signed Parallel Multipliers

Canturk Isci
Department of Electrical Engineering
Princeton University
Princeton, NJ
609 986 7446
canturk@princeton.edu

Richard C. S. Morling
University of Westminster,
Department of Electronic Systems,
Applied DSP & VLSI Research Group
London W1W 6UW, UK
+44 20 7911 5084
morling@cmsa.wmin.ac.uk

Izzet Kale
University of Westminster,
Department of Electronic Systems,
Applied DSP & VLSI Research Group
London W1W 6UW, UK
+44 20 7911 5157
kalei@wmin.ac.uk

ABSTRACT
Multipliers are often the critical functional blocks of datapath architectures. Due to their deeply embedded configurations in datapath architectures and two dimensional iterative array structure, they attain very low controllability and observability, which entails Built in Self Test (BIST) for multiplier testing. In this paper BIST techniques for a signed parallel multiplier are investigated. Deterministic fault simulation for single stuck at model is used to determine the fault coverage characteristics of the investigated methods, which are also compared to cell fault model (CFM). Various well-known and original input pattern generation techniques are investigated, with emphasis on PRBS generation using Linear Feedback Shift Registers (LFSRs) and Cellular Automata (CA), and use of repeated patterns.

Categories and Subject Descriptors
M.1.6 [**Design Methods**]: Testing and test generation.

General Terms
Design, Verification.

Keywords
BIST, multiplier, testing, LFSR, Cellular Automata

1. INTRODUCTION
With the immense present and ongoing technological developments in communications and computerization, digital signal processors (DSPs) and general purpose processors are two of the most active fields in VLSI and Application Specific Integrated Circuit (ASIC) design. Datapath architectures, which constitute the operational backbone of these two systems are of particular importance, as even though the significant advances in computation speed, electronic design automation (EDA) tools and IC technology instigate faster, smaller and less power dissipating circuits with less production time overhead, the requirements for efficient testing methodologies get stubbornly more demanding as such architectures are deeply embedded in overall system structure, with low controllability and observability, and increased gate to pin ratios.

Multipliers are critical functional blocks of datapath architectures in terms of speed and area and due to their deeply embedded configuration, the low observability and controllability, which are even deteriorated by the general regular 2 dimensional iterative array structure[3], make multiplier testing a significant bottleneck in the design process. Therefore, BIST architectures are the preferred solution in multiplier testing for most applications ([3],[4]), as efficient BIST methods provide testing at the operation speed of the overall system – *at speed testing* –, very high fault coverage with moderate amount of test vectors – constant or linearly dependent on multiplier size – and reduced test time, which in turn reduce the test cost.

In this paper, we investigate several BIST techniques for a signed parallel multiplier. The second section describes the design/simulation flow. The third section describes the design and design units. The fourth section discusses alternatives for BIST and fault simulation and the fifth section summarizes the significant outcomes.

2. DESIGN DATA AND TOOL FLOW
For the design of multiplier and BIST circuit, we used HDL design entry with VHDL, using Mentor Graphics' Renoir. This allowed us faster, flexible design phase yet required an intermediate synthesis step for detailed fault simulation with Mentor's QuickFault as shown in the overall design and data flow in figure 1.

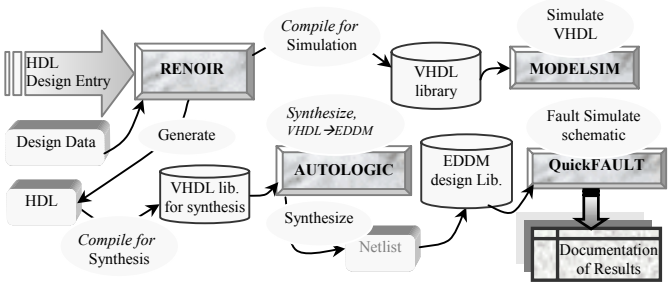

Figure 1, Design and Tool flow

As seen in the data and tool flow, the VHDL design is converted to Electronic Design Data Model (EDDM), which is the general data format for most Mentor tools. QuickFault is the preferred fault simulator for deterministic fault simulation. We relied on deterministic fault simulation, for accurate results trading for computation time.

3. MULTIPLIER AND BIST CIRCUIT DESIGN

The multiplier is designed as a signed, parameterized, parallel carry propagate array (CPA) multiplier. The designed BIST circuitry comprises of a parameterized LFSR for input pattern generation and a parameterized signature analyzer for output compression. Although the BIST circuit design follows after the investigation of various techniques, we demonstrate only the representative final circuits as the pattern generation is done in software for different techniques during fault simulation. Although only CPA multiplier is implemented in design, [3] suggests the fault coverage is very slightly dependent on multiplier architecture.

3.1 Multiplier Architecture

In general, the multiplier function is divided into two major parts: 1) Bit product generation 2) Addition of bit products to form the final product. Different multiplier architectures emerge from how these two steps are performed. In unsigned multiplication, the second step is mere addition, while in signed multiplication, the second step includes an addition or subtraction in the final level of partial product accumulation depending on the most significant bit (MSB) of multiplier. In unsigned multiplication, final product for an NxN multiplication is 2N bits. However, for signed multiplication, 2N-1 bits are sufficient as long as both multiplier and multiplicand do not take their most negative value simultaneously. Two distinct properties of signed multiplication are: 1) Multiplier and multiplicand are not completely symmetric in hardware architecture 2) In order to keep track of sign, sign extension must be handled during partial product additions.

In twos complement addition. considering multiplication as the accumulation of partial products of multiplicand by the individual bits of multiplier, the sign extension is performed as one bit per partial product, except for the first row, which requires 2 bit sign extension as described in figure 2.

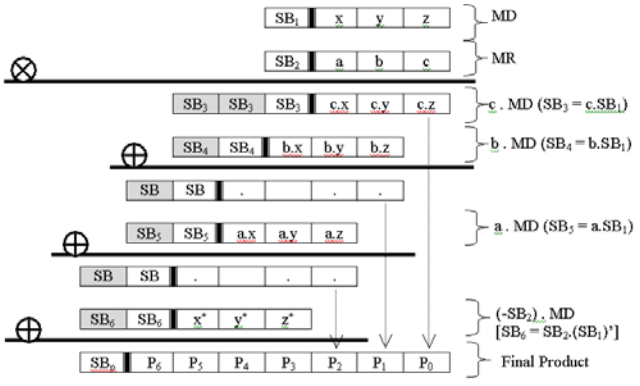

Figure 2, Twos Complement Multiplication

The shaded bits are extended signs for correct signed addition. (.)' denotes the complement of the binary value. As can be deduced, the last set of partial products is either all 0s if SB_2 is 0, or twos complement negation of multiplicand if SB2 is 1; which is stated as "(-SB2).MD". As observed in figure 2, unlike signed multiplication, the partial summations require N+1 bitwise additions for each accumulation step. In unsigned addition, the MSB of sum is gathered from the carry out output of the MSB adder however, in signed multiplication, the adder terms are sign extended and the MSB is the final adder's sum output, where the carry out is discarded as shown in figure 3.

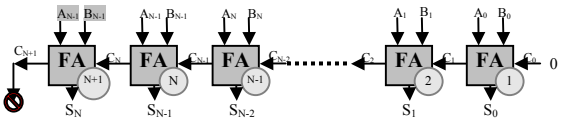

Figure 3, Sign extended 2s Complement Addition

However, as described in [6], a slight modification in the MSB adder can alleviate this redundancy. In the sum function for S_N, $S_N = A_{N-1} \oplus B_{N-1} \oplus C_N$, expanding C_N as $maj(A_{N-1},B_{N-1},C_N)$[1] reveals:

$$S_N = A_{N-1}B_{N-1} + A_{N-1}(C_{N-1})' + B_{N-1}(C_{N-1})' \qquad (1)$$

This expression is almost the same as the carry out function for N^{th} adder, with the carry in inverted. Therefore we modify the carry out of the N^{th} adder into the above equation to produce S_N bit without the expenditure of the $N+1^{th}$ adder. With this modified MSB Full Adder (FA), the adder circuit demonstrated in figure 4 is almost at equivalent cost to an unsigned adder circuit.

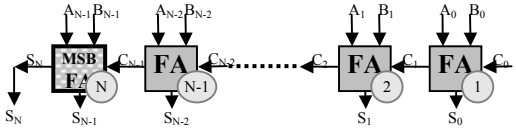

Figure 4, Modified signed adder

Using this described modification, the parameterized multiplier is designed in VHDL. The generic structure of multiplier is shown in figure 5.

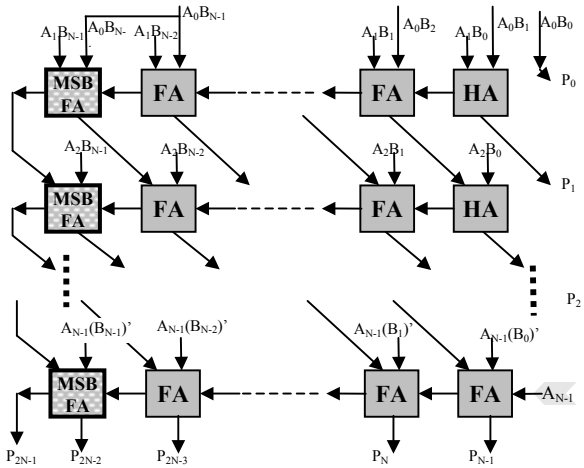

Figure 5, NxN CPA multiplier structure

[1] maj represents the majority function:
$A_{N-1}B_{N-1} + A_{N-1}C_{N-1} + B_{N-1}C_{N-1}$

3.2 LFSR Architecture

After investigation of BIST input pattern generation techniques in section 4, the technique decided upon was repetitive pattern PRBS generation using an 8 bit LFSR with seed x7B. Nevertheless, to provide a more flexible top level system, a parameterized LFSR is designed in VHDL.

LFSR is a shift register configuration with the XOR feedback – modulo 2 sum – of the selected flip-flop outputs, named *taps*. When the taps are chosen properly, the LFSR will traverse through all possible 2^n-1 states except for one forbidden state, where n is the length of LFSR, and will produce a maximum length PRBS sequence named *M-sequence*. For PRBS generation, the LFSR is first initialized to a well-known stage, named *seed*. In a type-A LFSR – as in figure 6 – the serial feedback at time i, y_i, of LFSR can be represented as the summation, in Galois Field of 2 (GF(2)), of all previous serial feedbacks in time as:

$$y_i = \sum_{j=1}^{\infty} c_j y_{i-j} \tag{2}$$

where c_j=1 represents an existing tap for flip-flop j and c_j=0 represents no tap. Then, representing the y_i values in time as a function of x in GF(2) as shown in equation 3:

$$G_y(x) = \sum_{m=0}^{\infty} y_m x^m \tag{3}$$

where x^m represents the m^{th} timestamp; and substituting equation 2 as y_m in equation 3, we arrive at the division equation for y_m:

$$G_y(x) = \frac{\sum_{j=1}^{n} c_j x^j \left(y_{-j} x^{-j} + y_{-(j-1)} x^{-(j-1)} + \cdots + y_{-1} x^{-1} \right)}{1 + \sum_{j=1}^{n} c_j x^j} \tag{4}$$

This equation describes the initial condition and tap locations of the LFSR and is a GF(2) division of seed dependent numerator polynomial to the only tap location dependent denominator polynomial, referred as the *characteristic polynomial* ([1],[8])

Regarding this division process, it is proven ([1], [9]) that for $G_y(x)$ to have maximum period, the characteristic polynomial must be not factorizable. Moreover, as $G_y(x)$ will still be periodic with 2^n-1, the characteristic polynomial must be a factor of $1+x^{2^n-1}$. The polynomials that satisfy above conditions are *primitive polynomials*, which are a special case of *irreducible polynomials,* and are used as the characteristic polynomial for maximum length LFSRs.

A generic representation of the design that uses the seed x7B ("01111011") with characteristic polynomial $1 + x^1 + x^5 + x^6 + x^8$ is demonstrated in figure 6.

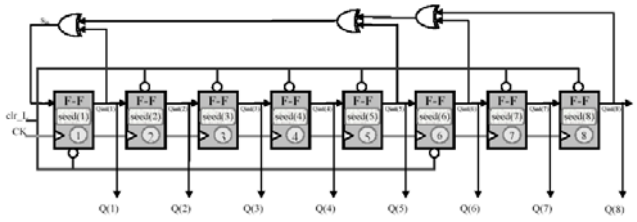

Figure 6, Structure of LFSR

3.3 Multiple Input Signature Analyzer

For BIST of multiple output circuits, multiple input signature registers (MISRs) are extensively utilized in practice due to their easy and low cost implementation and efficient fault coverage. As described in section 4, signature analysis is observed to be have very low fault masking and is the implemented BIST technique for output compression. MISR, in principle is not different from the single input signature analyzer, but instead of taking one serial input from the first flip-flop, every flip-flop in the MISR has one input coming from the primary outputs of the to be tested circuit as shown in figure 7. Therefore, for an n output circuit, at least an n stage MISR is used. In figure 7, the c_n switches define the

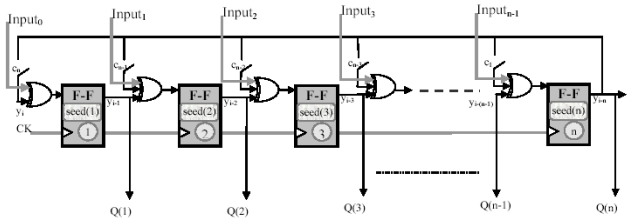

Figure 7, Multiple Input Signature Analyzer

divisor polynomial D(x) and the Q outputs show the internal stages. The residue is usually clocked out serially through one of the flip-flop outputs.

If the initial state of the MISR is known to be $S_0(x)$, then, the k^{th} state can be computed, using the set of all primary outputs to be compressed, defined as $I_i(x)$ for time i, and D(x), as:

$$S_k(x)=[x^{k-1}I_0(x) + x^{k-2}I_1(x) +\ldots+ x^1I_{k-2}(x) + x^0I_{k-1}(x)] \bmod D(x) \tag{5}$$

where, the $I_i(x)$ polynomials represent the i^{th} set of inputs to the signature analyzer. Consequently, the expected signature can be analytically derived from the expected outputs of the circuit, prior to application of signature analysis.

In terms of fault masking, assuming uniform distribution of error bits, for a length 'n' signature analyzer, output width of 'm' ($m \le n$) and length 'L' input test set, the fault masking probability is shown to be ([1]):

$$P_{fm} = \frac{2^{mL-n} - 1}{2^{mL-1} - 1} \tag{6}$$

which reduces to $P_{fm} = \frac{1}{2^n}$ for large L.

For the top level design, a parameterized signature analyzer is implemented in VHDL as in figure 7 as a generic example.

4. BIST TECHNIQUES

As described in design flow, the VHDL design for multiplier is synthesized into EDDM for fault simulation and the schematic representation of the multiplier is used as the design input for QuickFault, which is a deterministic fault simulator, based on single stuck at model. During the investigation of different BIST schemes, we used automatically generated stimulus files, either created by QuickFault or by Matlab Scripts. With the quick turnover of this software approach, various techniques are investigated without the necessity of building the BIST hardware. After we concluded the investigation, the preferred BIST structure is implemented as described in section 3.

As discussed in [1] and [8], single stuck at model, which is the most commonly used fault model is proven to also detect all multiple stuck at faults in a two level combinational circuit and a set of patterns, which detect all single stuck at faults will also

detect bridging faults([1]). Although until recently this model was observed to be fairly adequate in fault representation in real circuits, particularly in bipolar technologies, the advanced CMOS technologies begin to produce faults, which cannot be modeled with the single stuck at model. A more comprehensive technique named “Cell Fault Model”, which is proposed in [11], is defended in [3] as a more realistic technique, but it lacks the required simulation tools for general acceptance. Nevertheless, [12] provides indirect techniques to utilize this model with the current single stuck at fault simulators. However, as demonstrated in 4.2.1, although it can be verified that CFM is more comprehensive than single stuck at model, the fault coverage measurement method, "Cell Fault Coverage" (CFC), is seen not to always reveal more pessimistic results than single stuck at model, contrary to the outcomes of [3]. One drawback of the single stuck at fault model is the nonstandard determination of primitive levels for hierarchical faulting. As each vendor supplies different set of leaf cells, the results of fault simulation are dependent on the used technology. To prevent possible criticism about the level of abstraction in fault simulation, we used the most detailed possible representation of the circuit. Consequently, the used primitives were AND, OR, NOT gates and D flip-flops. Although this description is not very realistic, any vendor's cells will be a superset of these, and the specified fault coverage for the circuit will always represent a safe lower bound in any technology.

4.1 16 Bit Downcounter

As the first BIST technique, we applied a 16 bit fully exhaustive downcounter test, which will serve us as a benchmark when evaluating other strategies. One of the expectations of this fault simulation is not being able to achieve 100% fault coverage with even full-exhaustive testing, as the hierarchical design procedure might inherently involve some untestable nodes at this detailed level. Therefore, the fault coverage result of this test serves as the upper bound for the following fault simulations. With the injected 6008 hierarchical faults, we achieved a 98.49 % fault coverage after all the possible patterns are applied. Therefore, no test can exceed this value of fault coverage. This also raises one counterargument against Cell Fault Model and Cell Fault Coverage (CFC) described in [3]. It is suggested on [3,p. 947] that single stuck at fault simulation fault coverage values are always larger than the CFC values. However, the CFC value of 99.40% for the 8x8 CPA multiplier given on p. 945 is not even achievable with single stuck at model with the primitives we use. As a conclusion, we disagree with [3] that CFC is always more pessimistic than single stuck at model and we assert that this rather depends on the primitive levels used in hierarchical faulting.

The histogram plot of detected faults per each applied input test vector is shown in figure 8, which reveals vast redundancies in the fault detection with the upcounter test. There exists a very undesirable spread of detections in the first 20K cycles, and there is still a very significant peak of detection around 32K – specifically 32897th vector.

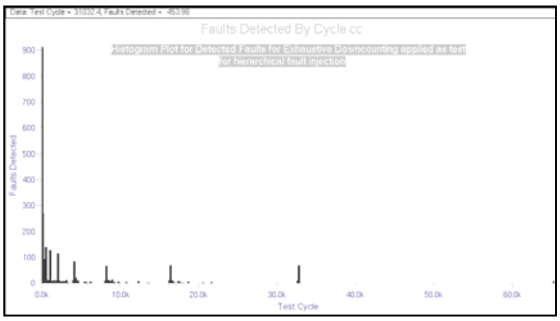

Figure 8, Histogram for exhaustive downcounter test

4.2 8 Bit Upcounter with Repetitive Patterns

Repetitive patterns have been widely used in testing of iterative logic arrays and have been shown to provide excellent fault coverage with a small test set ([3],[4],[11]). For the multiplier, the a and b inputs of multiplier are applied the same patterns for every k inputs where k is the defined *"repetition length"*. If we apply the same pattern to both lower and higher 4 bits of one of the 8 bit inputs in our 8x8 multiplier, the repetition length of this pattern is 4. To verify the effectiveness of repetitive patterns we applied a BIST scheme with a repetition length of 4 and an 8 bit upcounter, which can produce only 256 of the 2^{16}=65536 possible input patterns, chosen as input pattern generator. The corresponding input test vectors are then as shown in table 1.

Table 1, Repetitive pattern with repetition length(k)=4, pattern generator → 8 bit Upcounter

	a:		b:	
	a(7:4)	a(3:0)	b(7:4)	b(3:0)
4x4 bits per pattern	0000	0000	0000	0000
	0001	0001	"	"
	•	•	"	"
	•	•	"	"
	1111	1111	"	"
	0000	0000	0001	0001
	0001	0001	"	"
	•	•	"	"
	•	•	"	"
	1111	1111	"	"
	•		•	
	•		•	
	0000	0000	1111	1111
	0001	0001	"	"
	•	•	"	"
	•	•	"	"
	1111	1111	"	"

16 patterns of 'a' for each 'b'

Total 256 patterns

With this pattern generation scheme, an excellent fault coverage of 97.02% is achieved with the mere 228 of the 256 described vectors. Both the histogram and the fault coverage plot in figure 9 show the spread is within the first 228 cycles and the detection activity includes very little redundancy, with most vectors contributing to fault detection.

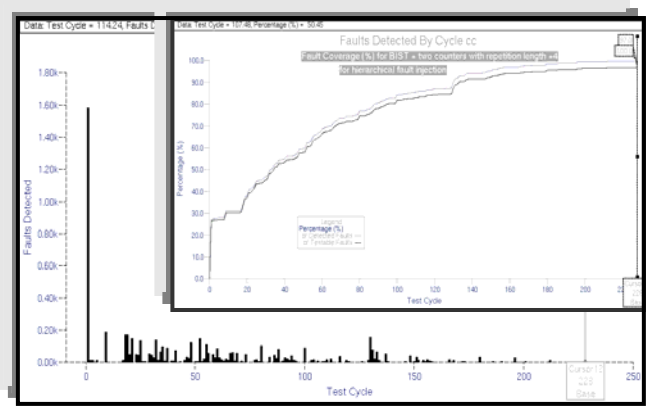

Figure 9, Histogram and fault coverage (%) for Repetitive upcounter with k=4

Figure 9 also leads to two other observations. There are 2 very ineffective regions right at the start of test, within the first 20 input test vectors – indicated by the inactive regions in histogram and flat regions in plot – and a very effective input sequence starting around 127th input test vector – indicated by the peaks in histogram and steep jump in plot. These redundancy and peak observations are later used to improve test by seed determination.

4.2.1 CFC vs. Single Stuck at Fault Coverage

This BIST technique also provides the opportunity of comparing CFC with Single Stuck at Fault Coverage. As stated in [3], the CFC for the 8x8 CPA multiplier with repetition length 4 counter/PRBS test is 99.40%. However, the fault coverage with the single stuck at model, with the described primitive levels is only 97.02%. Therefore, it should be noted that, although CFC provides more pessimistic results for the fault coverage when the same cells are taken as primitives and applied single stuck at fault simulation, it cannot be securely suggested that CFC is independent of gate level implementation as a more detailed hierarchical faulting can produce worse results with single stuck at model, as observed herein.

4.3 PRBS Techniques with LFSRs and CA

The two previous techniques discussed can be classified as deterministic techniques, and pattern generation with LFSRs and CA is referred as pseudorandom pattern generation, which is the main interest of this paper.

For the investigation of PRBS techniques we started with 16 bit LFSRs, and determined a simple yet efficient methodology for seed determination. We then, extended the investigation of LFSRs to repetitive patterns and with the applied seed determination technique, achieved very promising results. For the BIST circuits, the outputs are generated with Matlab, regarding the LFSR structure described in section 3.2. The fault simulation details are listed in Table 2 and are not repeated here for brevity.

4.3.1 Seed Determination Methodology

For both 16 and 8 bit LFSR structures, we initially started with a seed of x"0001" or x"01", and as described in section 4.2, identified the redundant regions and "late peaks", which refer to the peaks observed in the histogram after the initial "hyperactive" region. Then, we determined patterns around the late peaks, which, when chosen as the initial state, eliminate the redundant regions and include the late peaks. Then, this algorithm is planned to be applied recursively if new late peaks are observed with the terminating condition of completing one whole period of the input sequence, i.e. starting from x"01", increasing the seed up to x"FE" and then wrapping back to seed x"10" as the next seed candidate, in which case we restart to observe the initial late peaks. However, after at most two iterations, no more late peaks are observed in the new histograms and the algorithm converged efficiently. An example to the application of algorithm is shown in figure 10, where the finally implemented seed, x"7B" is determined from seed x"01".

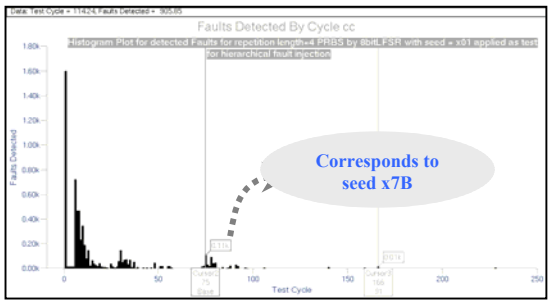

Figure 10, Histogram for 8 bit LFSR with seed x01 (k=4)

In the next iteration, using seed x7B reveals an excellent histogram and fault coverage curve, as shown in figure 11, and no new late peaks are detected.

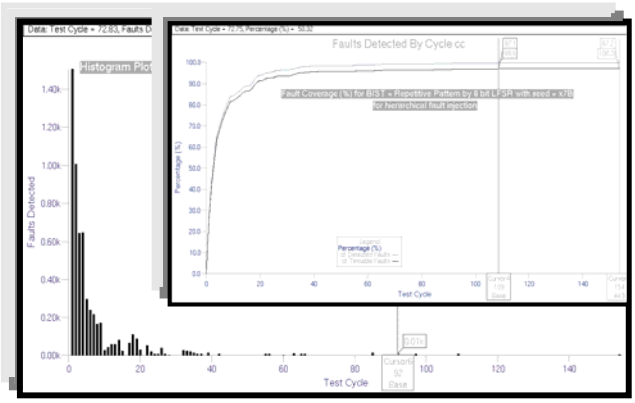

Figure 11, Histogram and fault coverage (%) for 8 bit LFSR with seed x07 (k=4)

4.3.2 Cellular Automata

A second less studied alternative for PRBS generation is cellular automata. CA have a similar structure to LFSRs with the ultimate difference that all the cell interconnections have some XOR operation and that no global feedback is required. Therefore, the regular shifting of data within the shift register is not existent in CA ([8],[10]). The input to a CA cell depends only on its adjacent neighbors and maybe itself depending on whether the cell is a 90 or 150 cell, as shown in figure 12.

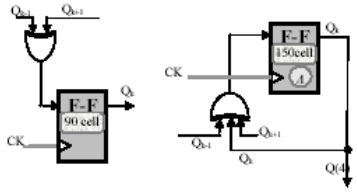

Figure 12, 90 and 150 CA cells

Other functions of the three outputs are also investigated and it is formally proven that, only 90 and 150 cells produce m-sequences for PRBS generation. Moreover, not all combinations of 90 and 150 cells can produce m-sequences. It is theoretically proven that, for n≤150, at most 2 150 cells are sufficient to produce a configuration of 90 and 150 cells that produce m-sequences. An exemplary 4-stage max-length CA is shown in figure 13.

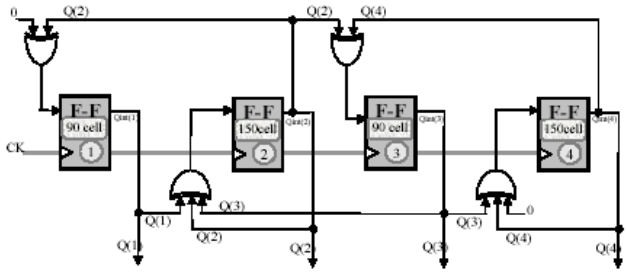

Figure 13, 4 Stage max-length CA

The generated sequence for a CA can be analytically determined using a tridiagonal transition matrix T, whose 1st diagonals are all 1s and main diagonal only 1 for 150 cells. Then, the next states of the CA can be determined from this matrix mapping. As an example, the next state of the above CA for current state "1111" is found as[2]:

$$[1\ \ 1\ \ 0\ \ 0] = [1\ \ 1\ \ 1\ \ 1] \times \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad (7)$$

Despite the increased circuit size, CA are expected to perform better than LFSRs as they don't possess the inherent shifting behavior of LFSRs, thus intuitively imitate randomness better. Therefore, similar to LFSR case, CA are investigated as BIST pattern generators for both 16 bit exhaustive and 8 bit repetitive patterns, again using the same seed determination methodology. The acquired results revealed a very similar profile as LFSRs, with both the application repetitive patterns and seed determination method improving fault detection characteristics. However, compared to LFSRs, CA revealed only very slight improvements, which are also demonstrated in table 2.

[2] Hence all the operations are in GF(2)

Table 2, Summary of BIST Results

BIST method	*Fault Coverage vs. # of patterns*
8 bit Upcounter Using Repetitive patterns with k=4	97.02% with 228 patterns
16 bit LFSR with seed=x0001	97.2% with 157 patterns
16 bit LFSR with seed=xEA58	97.2% with 134 patterns
16 bit LFSR with seed = x8080	97.2% with 153 patterns
8 bit LFSR using repetitive patterns with k =4 and seed=x01	97.12% with 228 patterns
8 bit LFSR using repetitive patterns with k =4 and seed=x7B	97.2% with 154 patterns 97.1% with 109 patterns
8 bit LFSR using repetitive patterns with k =4 and seed=xB7	97.0% with 82 patterns 97.24% with 230 patterns
16 bit CA with seed=x0001	97.2% with 159 patterns
16 bit CA with seed=x8080	97.2% with 138 patterns
16 bit CA with seed=x534F	97.2% with 122 patterns
16 bit CA with seed=xD5D5	97.2% with 127 patterns
8 bit CA using repetitive patterns with k =4 and seed = x01	97.1% with 139 patterns 97.2% with 179 patterns
8 bit CA using repetitive patterns with k =4 and seed = x6D	97.2% with 124 patterns 97.0% with 123 patterns
8 bit CA using repetitive patterns with k =4 and seed = xAB	97.19% with 125 patterns 97.1% with 110 patterns

As observed in the table, different implementations are favorable depending on the target fault coverage. Nevertheless, two important deductions of BIST investigation are, the repetitive patterns and the seed determination algorithm work extremely well and CA have no significant advantage over LFSRs. Consequently, the implementation decision for input pattern generation has been for 8 bit LFSR with seed x7B.

4.4 Signature Analysis

After the conclusion of input pattern generation, output compression is performed with signature analysis. The regular multiplier structure suggests that it is not probable to have a single stuck at fault that causes two distant separate multiplier outputs change at the same time and therefore we expect a very low fault masking probability with signature analyzer. In order to reduce fault masking before the faulty bit is fed back to signature analyzer, a high weight divisor polynomial is used and application of 8 bit LFSR with seed x7B revealed 96.80% fault coverage, which is almost as high as original fault coverage with direct measurement of 16 multiplier outputs.

Inclusion of the BIST circuit in fault simulation is observed to produce almost same fault coverage as faulting only the multiplier.

4.5 Larger Multipliers

In order to verify the effectiveness of repetitive patterns, we synthesized larger circuits with the same 8 bit LFSR with seed x7B, using repetitive patterns with k=4. 16x16, 32x32 and 24x24 multipliers are synthesized, but only 16x16 circuit could be fault simulated, with 26456 injected faults, due to computation memory problems of deterministic fault simulation. Using the repetitive pattern scheme, the 16 bit 'a' and 'b' inputs of multiplier are connected to the 8 bit LFSR outputs as:

b(15:12), b(11:8), b(7:4) and b(3:0) ←LFSR(1:4)

a(15:12), a(11:8), a(7:4) and a(3:0) ←LFSR(5:8)

Producing a constant test of 256 vectors as in 8x8 multiplier.

Counterintuitively, as shown in figure 14, we achieve an unexpectedly good result. The fault coverage reached 97% in just 57 cycles and it climbed up to 98.83% in the whole test, revealing an even better fault coverage than the 8x8 case.

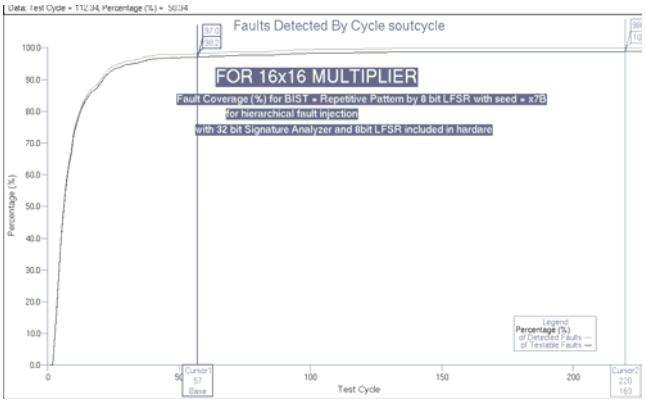

Figure 14, Fault coverage plot (%) 16x16 multiplier with 8 bit LFSR with seed x7B (k=4)

This interesting result reveals a very promising conclusion. Repetitive patterns provide very high fault coverage with a fixed number of patterns independent of the size of the multiplier.

5. CONCLUSIONS

In this paper we presented a comparative investigation of BIST techniques for a signed parallel CPA multiplier, with the modified MSB full adder. Various input pattern generation techniques are compared with emphasis on PRBS techniques using CA as well as LFSRs, and repetitive patterns. A simple, yet efficient seed determination heuristic for both LFSRs and CA is described. A quantitative comparison of CFM and single stuck at model is also presented. With the over-detailed hierarchical fault model, guaranteed worst case results for fault coverage are acquired.

Compared to the deterministic exhaustive downcounter test, exhaustive pseudorandom test with LFSRs and CA revealed much better fault coverage characteristics. For non-exhaustive test, repetitive patterns are seen to provide excellent results with constant size test sets. For both exhaustive and non-exhaustive tests, CA are seen to provide no significant improvement over LFSRs despite their more random natured output sequence. Described seed determination technique is observed to work very efficiently in seed determination for both LFSRs and CA. A high weight signature analyzer is seen to be very effective in output data compression, with insignificant downgrade from direct output measurement.

The comparison of fault modeling techniques CFM and single stuck at model revealed, although CFM is more comprehensive than single stuck at model for the same level of faulting hierarchy, a significantly more detailed hierarchy produces worse fault coverage results with single stuck at model than CFM.

Interestingly, use of repetitive patterns with a constant repetition length – thus constant size test set – revealed even better fault coverage with larger multipliers.

6. ACKNOWLEDGEMENTS

Our thanks to Mr. Alan Wood for his system support throughout the course of this research.

7. REFERENCES

[1] Russell, G. and Ian L. Sayers, "*Advanced Simulation and Test Methodologies for VLSI Design*", London: Van Nostrand Reinhold (International), 1989, ISBN 0-7476-0001-5

[2] Psarakis, M., D. Gizopoulos, A. Paschalis, N. Kranitis and Y. Zorian, "*Robust and Low-Cost BIST Architectures for Sequential Fault Testing in Datapath Multipliers*", VLSI Test Symposium, 19th IEEE Proceedings on. VTS 2001, pp. 15-20, 2001

[3] Gizopoulos, D., A. Paschalis and Y. Zorian, "*An Effective Built-In Self-Test Scheme for Parallel Multipliers*", IEEE Trans. on Computers, vol. 48, no. 9, pp. 936-950, Sep. 1999

[4] Gizopoulos, D., A. Paschalis and Y. Zorian, "*Effective built-in self test for Booth multipliers***"**, IEEE Design & Test of Computers, vol. 15, no. 3, pp. 105 -111, July-Sep. 1998

[5] Weste N. H. E. and K. Eshragian, "*Principles of CMOS VLSI Design, A Systems Perspective*", Massachusetts: Addison-Wesley, 1993, ISBN 0-201-08222-5

[6] Morling R. C. S. And I, Kale, lecture notes, "*DSP and Communication Processor Design*", MSc VLSI System Design, Univ. of Westminster, Feb. 2001

[7] Roth, C. H., "*Digital Systems Design Using VHDL*", Boston: PWS, 1998, ISBN 0-534-95099-X

[8] Hurst, S. L., "*VLSI Testing, Digital and Mixed Analogue/Digital Techniques*", London: IEE Circuits, Devices and Systems Series 9, 1998, ISBN 0-85296-901-5

[9] Bardell P. H., McAnney W. H. and Savir J., "*Built-In Test for VLSI: Pseudorandom Techniques*", New York: John Wiley & Sons, 1987, ISBN 0-471-62463-2

[10] Lala, P. G., "*Digital Circuit Testing and Testability*", San Diego: Academic Press, 1997, ISBN 0-12-434330-9

[11] Kautz, W. H., "*Testing for Faults in Cellular Logic Arrays*", proceedings of the 8th Annual Symposium on Switching and Automata Theory, pp. 161-174, 1967

[12] Psarakis, M., D. Gizopoulos and A. Paschalis, "*Test Generation and Fault Simulation for Cell Fault Model Using Stuck at Fault Model Based Test Tools*", Electronic Testing Journal: Theory and Applications, vol. 13, no:3, Dec. 1998