Self Test Using Unequiprobable Random Patterns

Hans-Joachim Wunderlich
Universität Karlsruhe
Institut für Informatik IV
(Prof. Dr. D. Schmid)
7500 Karlsruhe
Postfach 6980
Phone: 0721/608-4257
FRG

Abstract

Self test modules based on linear feedback shift registers (LFSR) like BILBOs perform signature analysis and generate equiprobable pseudo random patterns. The self test is carried out after the production process and also during system operation while the circuit is idle. But there exist many combinational circuits, which cannot be tested by equiprobable random patterns due to the unsufficient fault coverage. Recently it has been shown that this problem can be solved if each primary input is set to logical "1" with its special optimal probability.

In this paper we present a module generating unequiprobable random patterns, which can also perform signature analysis and work like a normal register similar to the well known BILBO. The hardware overhead of this module has the same magnitude as a conventional BILBO. Thus the class of self testable circuits is enlarged without additional costs.

Keywords: Self test, design for testability, reliable hardware

1. Introduction

Reconfigurable and fail safe architectures of reliable computing systems provide the detection and identification of faulty components. Efficiently this is done by circuits which are testing themselves while their function is not needed by the system. The self test feature is also used to support the production test now reaching more than 60 % /Benn84/ or even 70 % /Will86/ of the overall chip costs.

Most self test strategies are based on linear feedback shift registers (LFSR) generating pseudo-random patterns which set each flip-flop to logical "1" with probability 0.5. During self testing the system registers are configurated to LFSRs, generate peudo random patterns and perform signature analysis /McCl85/, thus testing the combinational part of the circuit. Well known is the BILBO approach /KOEN79/.

Here we can dispense with the time consuming automatic test pattern generation, and no expensive test equipment is needed. The test is carried out in high speed, and therefore many technology dependent dynamic faults can be detected in addition (/Tsai83/, /WuRo86/). Since a randomly generated test set is larger than a deterministic one, the detection rate of logical faults not in the fault model, multiple faults for instance, will be higher.

Let $b \in [0,1]$ be the probability to detect all faults $f \in F$ of a non-redundant circuit by N random patterns. The length N can be estimated for each required confidence b, if for all faults the detection probabilities are known (/WCM86/, /Wu85/, /BaSa84/). In recent papers several algorithms estimating fault detection probabilities have been presented (/BDS84/, /Wu85/, /AgJa84/, /SETH86/), the approach of this paper is based on the tool PROTEST.

PROTEST determines fault detection and signal probabilities at gate level by some analytical procedures described in /Wu85/.

But now it turns out that there are many circuits which cannot be tested randomly due to faults with very low detection probabilities. Based on the estimations of PROTEST table 1 shows for some circuits the test lengths which are necessary in order to detect all detectable faults..

	Circuit	Required test length
*	S1	$5.6*10^8$
*	S2	$2.0*10^{11}$
	C432	$2.5*10^3$
	C499	$1.9*10^3$
	C880	$3.7*10^4$
	C1355	$2.2*10^6$
	C1908	$6.2*10^4$
*	C2670	$1.1*10^7$
	C3540	$2.3*10^6$
	C5315	$5.3*10^4$
	C6288	$1.9*10^3$
*	C7552	$4.9*10^{11}$

Table 1: Necessary test lengths for a conventional random test (by PROTEST)

The circuits Cn are the well known benchmarks of the ISCAS 1985 test session /BRGL85/, the circuit S1 is a 24-bit comparator constructed by six Texas Instruments comparators SN7485 /TI80/, where some redundancies are removed, and S2 is the combinational part of a 32 bit divider /KuWu85/.

In table 1 the marked (*) circuits need an exorbitant size of the random test set. If we assume a system working at 20 MHz, then a self test technique applying one pattern within 3 cycles would need the very large test times listed in table 2.

Circuit	Test application time (sec)
S1	84
S2	30 000
C2670	1.7
C7552	73 500

Table 2: Time needed for self test

0731-3071/87/0000/0258$01.00 © 1987 IEEE

One cannot assume that components of a running system are idle during those long times, and therefore in many cases this conventional random pattern approach cannot be used to improve reliability. Furthermore the production test becomes uneconomical.

But recently it has been shown that the necessary test length can decrease by several orders of magnitude if each primary input i of the combinational circuit is set logical "1" with a specific optimal probability $x_i \in \{0,1\}$ /Wu85/, and an efficient algorithm has been presented to compute those optimized probabilities based on the circuit structure (/Wu86/, /Wu87/). Another approach tries to compute those optimized input probabilities during simulation /LBGG86/. In table 3 the necessary test lengths using optimized random tests are shown.

Circuit	Required test length
S1	$1.5*10^4$
S2	$4.0*10^4$
C2670	$6.9*10^4$
C7552	$1.2*10^4$

Table 3: Necessary test lengths for an optimized random test estimated by PROTEST

As already mentioned the predictions of PROTEST are correlated to the number of test patterns necessary to get complete fault coverage. This is validated for conventional and for optimized random tests by fault simulation in table 4:

Circuit	test length	fault coverage (conventional)	fault coverage (optimized)
S1	12 000	80.7 %	99.7 %
S2	10 000	81.2 %	99.2 %
C2670	4 000	88.0 %	99.7 %
C7552	4 096	93.9 %	98.9 %

Table 4: Fault coverage achieved by simulation using optimized and conventional random patterns

Assuming an appropriate self test strategy now all circuits are tested within a few milliseconds. Thus the production test can be carried out this way economically, and in addition the self test during system operation is possible, since the test time has the same magnitude as a disk access for instance.

For the rest of this paper we present a self test architecture applying optimized random patterns. In section 2 we resume the essential properties of LFSRs, and we present the basic structure of our approach. In section 3 we demonstrate, how a generator of unequiprobable random tests (GURT) is composed by cascading two basic types of cells.

In section 4 the four operating modes of a GURT are discussed, and in section 5 an example is presented.

2. Self test by random patterns

The most widely used self test techniques are based on LFSRs, where the system registers are augmented by some additional hardware. Then those registers can be controlled to perform the normal operating mode, the shifting mode or the LFSR mode. Fig. 1 shows the typical test configuration.

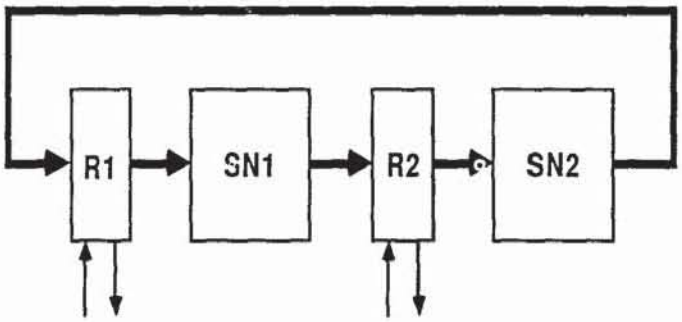

Fig. 1: A self test configuration

Here the test is carried out within five phases. First the registers R1 and R2 are reset. Then both registers work in the LFSR mode, where R1 produces random patterns for the combinational network SN1, and R2 compresses its responses by signature analysis. Third the signature of R2 is shifted out, and then both registers work as LFSR again, but R2 generates the patterns for SN2 and R1 performs signature analysis. At last the signature of R1 is shifted out.

The LFSRs produce random patterns by polynomial division over GF[2]. This is possible by two different architectures which are usually denoted as LFSR of type I and LFSR of type II (see /HeLe83/). Both automata are equivalent, and implement a polynomial division. For a discussion in some deeper detail see /Golo67/ or /HeLe83/. The LFSR of type I feeds back the linear sum

$$\sum_{i=0}^{r-1} g_{r-1-i} t_i$$

into t_0 (fig. 2a), and the LFSR of type II feeds $s_{r-1}+s_{i-1}$ into those s_i, where $g_i = 1$ (fig. 2b).

Both automata of fig. 2 perform a division by the polynomial $x^6+x^5+x^3+x^2+1$.

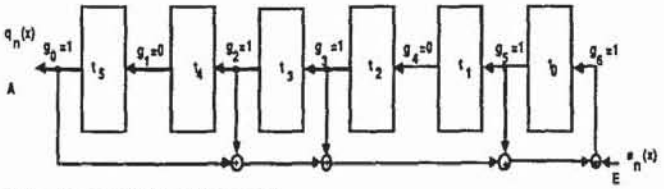

Fig. 2a: LFSR of type I

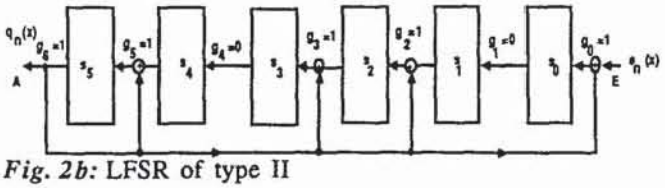

Fig. 2b: LFSR of type II

If they implement a division by a primitive polynomial, their period is maximum and a produced bit sequence A of length m satisfies some basic random properties:

1. The "1" appears with probability p = 0.5 approximately.
2. A run of length n has probability p^n approximately.

3. The autocorrelation function

$$c_m(t) := \frac{1}{m}\sum_{i=1}^{m} x_i x_{i+t}$$

is two-valued if m approaches infinity: $c(t) = p^2$ for $t \neq 0$ and $c(0) = p$.

The properties 2) and 3) should also hold for bit sequences realizing other probabilities than $p = 0.5$.

Conventional self test techniques use LFSRs of type I, since they can easily be composed by cascading identical cells. But pattern generation by this type has some significant disadvantages. One disadvantage results from the fact that between stage t_{r-1} and stage t_0 a rather complex boolean function has to be implemented, whereas in a shift register of type II this function is distributed to XOR gates between different stages making higher speed possible. Furthermore in a LFSR of type I two subsequent patterns differ only in one bit whereas the other bit positions are results of a simple shift operation, causing two other problems:

Already Bardell and McAnney /BaMc84/ noticed that one cannot produce parallel pseudo random sequences by one LFSR of type I feeding different scan paths, since the patterns in those paths would be highly correlated (fig. 3).

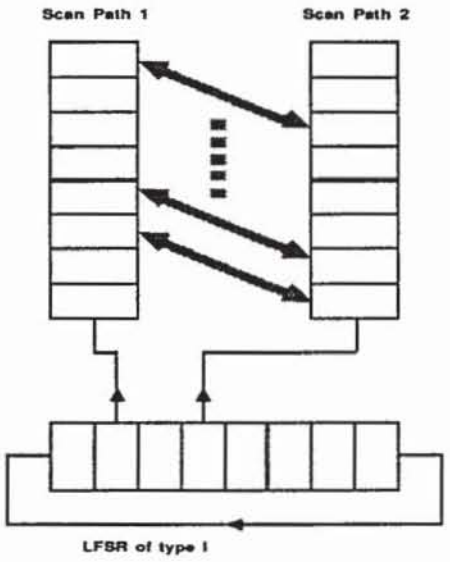

Fig. 3: Autocorrelation by using LFSRs of type I

If one wants to generate a bit sequence realizing another probability than 0.5, one cannot combine two storage elements by a boolean function, since the autocorrelation function wouldn't be two-valued any more. If for instance we want to generate a bit sequence of probability 1/4, and therefore use an AND combining the flip-flops at position i and at position i+k, then the resulting sequence violates property 3): If at time t the AND output is "1", then both positions x_i and x_{i+k} are "1", and in this case at time t+k the position x_{i+k} is "1" too. Thus if the AND output is "1" at time t, it is "1" with probability 0.5 at time t+k. Therefore $c(0) = 1/4$, but $c(k) = 1/8 \neq 1/16$.

Consequently we cannot generate biased patterns by LFSRs of type I, and we have to use a new self test architecture instead.

3. A Generator of unequiprobable random patterns (GURT)

A GURT has four operation modes:

1) Normal system operation as register
2) Unequiprobable random pattern generator
3) Signature Analysis
4) Shift register

Each GURT consists of two basic types of cascadable cells containing one master-slave D-flip-flop each and some additional circuitry T1 and T2 respectively. The functions of T1 and T2 are described in table 5 below:

Subcircuit T1						Subcircuit T2				
A	B	C	B_1	B_0	D	A	B	B_1	B_0	D
X	X	X	0	0	B≠C	X	X	0	0	B
X	X	X	0	1	B	X	X	0	1	B
X	X	X	1	0	B≠C≠A	X	X	1	0	A≠B
X	X	X	1	1	A	X	X	1	1	A

Table 5: Elementary functions T1 and T2

Fig. 4a and fig. 4b show implementations on gate level of those functions. The inputs B_0 and B_1 are control lines selecting one of the four modes.

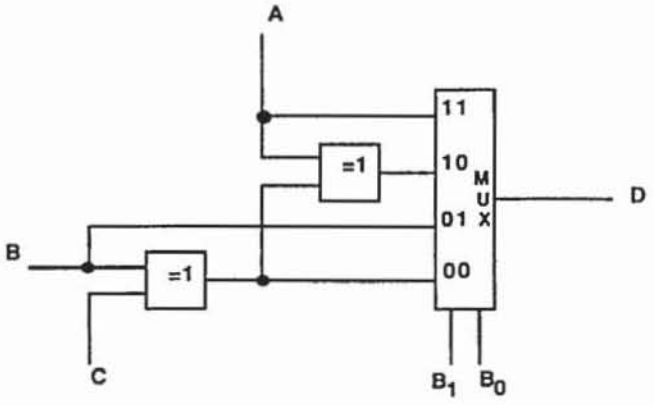

Fig. 4a: T1

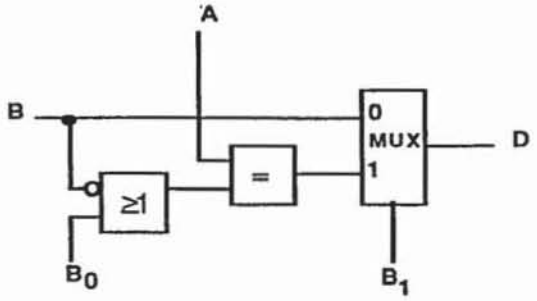

Fig. 4b: T2

The output D of T1 or T2 is connected to the data input of the flip-flop. In fig. 5a and fig. 5b the complete basic cells G1 and G2 can be seen.

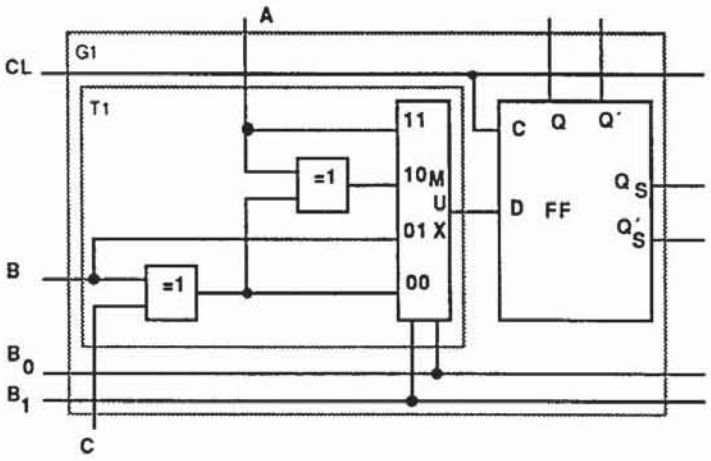

Fig. 5a: Basic cell G1

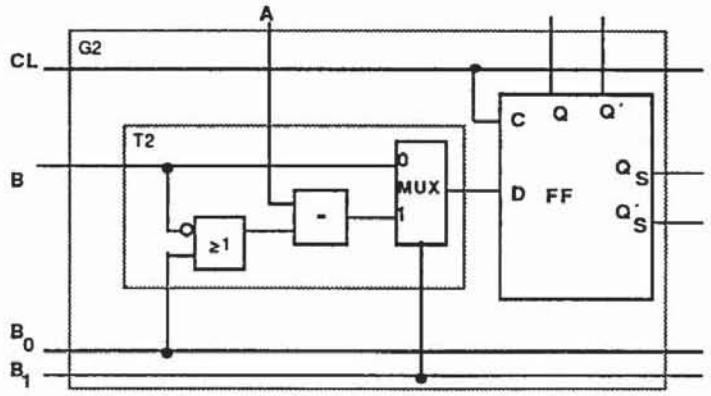

Fig. 5b : Basic cell G2

Cascading these cells two modules LR and SR are constructed.

Module LR consists of a chain of k+1 basic cells G1 and G2 in arbitrary order, but starting with G1 (fig. 6).

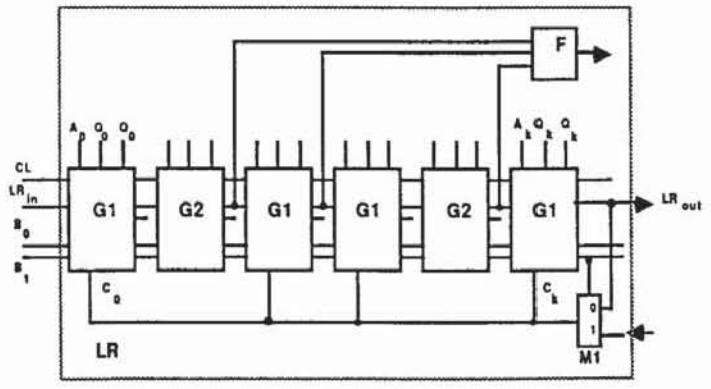

Fig. 6: Module LR

The output Q_S of a basic cell is connected to the input B of the following cell, the first B is the shifting input LR_{in}, and the last Q_S is the shifting output LR_{out}. Additionally LR_{out} is fed back to all C inputs of the G1-cells, controlled by a multiplexer M1 if B_1 is "0". If B_1 = "1" the multiplexer M1 selects an input explained later. Furthermore there is a boolean function F, getting its arguments by the Q_S outputs of some basic cells.

The module SR is just a cascadation of basic cells G2 (fig. 7), and the inputs are connected to a preceeding module LR. The multiplexer M2 selects F for B_0=B_1, and LR_{out} otherwise.

The shifting output SR_{out} is connected to the already mentioned input of the multiplexer M1 of the preceeding LR.

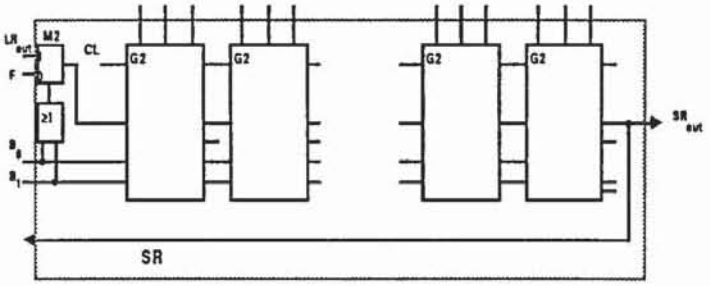

Fig. 7: Modul SR

The construction of a complete GURT by the modules LR and SR can be seen in the next section, where the operation modes of a GURT are discussed.

4. The operation modes of a GURT

The inputs (B_1,B_0) control the operation modes of a GURT:

(1,1) Normal system operation as register:
According to table 5 the modules T1 and T2 sensitize a path from the input A to the output D. Therefore the modules G1 and G2 respectively are working like a D-flip-flop with data input A.

(0,0) Unequiprobable random pattern generation:
Here the module LR configurates to a linear feed back shift register of type II, where the positions of the G1 cells determine the feed back function (fig. 8). The boolean function F gets its arguments by three flip-flops, which have to be selected carefully in order to diminish the autocorrelation of the resulting random sequence. Easily it can be shown, that each possible state of LR has the same probability, if LR_{in} is stimulated by a randomly generated (perhaps biased) bit sequence /Wu86/. In this case each position of LR is "1" with probability 0.5, and the random variables at all positions of LR are completely independent. Therefore each probability p ∈ {1/8,...,7/8}.can be generated by an appropriate function F, and the random sequence generated by F satisfies property 1). The fulfilment of properties 2) and 3) depends on the bias of the input sequence, the length of LR and the tabs for F. In a straightforward way this can be proven by describing the values of the input variables of F at different times in terms of the values of the flip-flops of LR at the starting time and in terms of the incoming random sequence at LR_{in}. Using this tabulating method the appropriate tabs can be found automatically.

The module SR becames a normal shift register which has the random sequence generated by F as input. Some flip-flops of SR may get only the inverted logical value of the preceeding one, therefore each flip-flop

of SR is logical "1" with probability p or with probability 1-p.

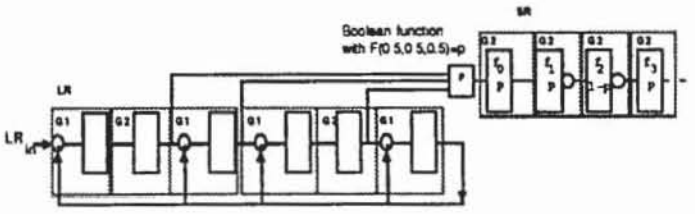

Fig. 8: Random pattern generation using LR and SR

Usually LR consists of 6 basic cells, and the incoming sequence at LR_{in} can be produced by a preceeding GURT too.

(1,0) Signature Analysis:
Here LR and SR together form a large LFSR with the parallel inputs A_i (fig. 10). Since fault detection by a LFSR is only determined by its length /Davi80/, we need not modify the resulting linear function by composing LR and SR.

(0,1) Shift register:
LR and SR form together a large shift register. This mode is used in order to initialize random pattern generation or signature analysis or in order to read the signature.

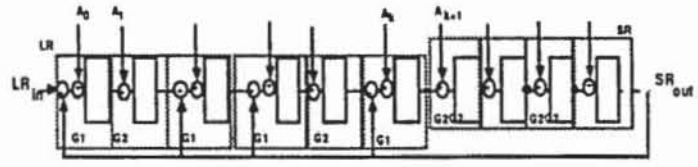

Fig. 9: A GURT in signature register mode

Using those four modes the system operation and a complete self test can be carried out.

5. The complete self test configuration

A GURT needs a sufficiently long pattern sequence as input in order to guarantee properties 1) and 2) and to avoid periodicity. This input sequence can be generated by a long LFSR with self-test features like a BILBO or a LR described above, or by a preceeding GURT.

Furthermore one GURT is only able to generate the three probabilities p, 1-p, and 0.5 at its storage elements. Therefore the inputs of the combinational circuit under test have to be grouped into four sets: one set contains the inputs which must be stimulated with probabilities 1/8 and 7/8, in the next set there are the inputs with probabilities 1/4 and 3/4, a further set needs probabilities 3/8 and 5/8, and at least there is the set with probability 1/2. If a circuit has inputs in each of those sets at least 3 GURTs must be connected in series (see fig. 10).

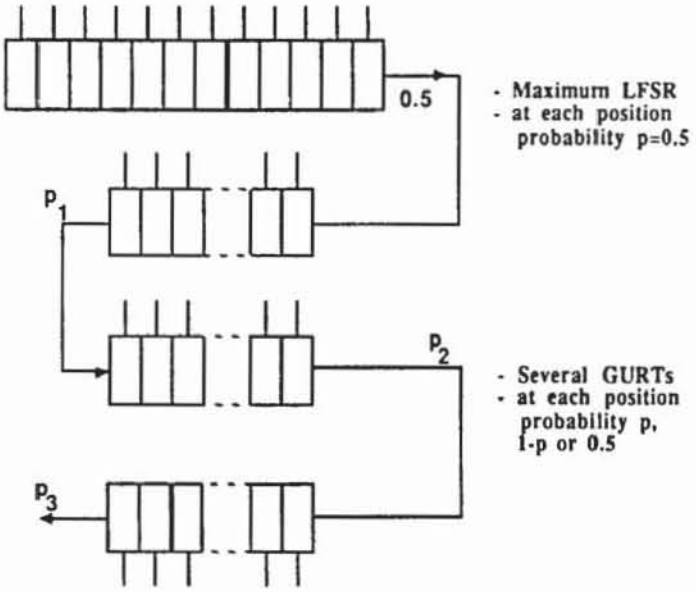

Fig. 10: Self test configuration using GURTs

Now we want to use GURTs in order to generate random patterns for the circuit example C2670. PROTEST proposes the optimized input probabilities listed in table 6. Furthermore we will use three GURTs I, II and III, and a module LR. GURT I realizes the probabilities 1/8, 7/8 and 1/2, GURT II the probabilities 1/4, 3/4 and 1/2, GURT III the probabilities 3/8, 5/8 and 1/2, and LR generates equiprobable patterns of 1/2. Then table 6 describes the correspondence between the primary inputs, the GURTs and LR:

primary inputs	optimized probability	GURT	realized probability
1-3, 7, 10-11, 28,32, 41, 43, 51, 53, 62, 64, 73-74, 76, 86, 96, 106, 115-233	0.5	LR	0.5
4, 13-20, 24 37,39,55, 88, 99-100	0.1	I	0.125
5-6, 21, 40, 25-26, 47, 54, 63,93	0.15	I	0.125
8, 103	0.7	II	0.75
9, 22, 114	0.9	I	0.875
12, 27, 92	0.55 0.2	LR II	0.5 0.25
29, 31, 33-36, 38, 42, 44, 48, 52, 56-61, 67, 75, 78-83, 95, 98, 105, 108-112	0.05	I	0.125

77, 84, 87, 94	0.95	I	0.875
30, 97	0.85	I	0.875
45-46, 70, 85, 89-91, 101	0.25	II	0.25
49-50, 71, 113	0.3	II	0.25
65, 69	0.35	III	0.375
66, 72	0.4	III	0.375
68,102	0.45	LR	0.5
104	0.75	II	0.75
107	0.65	III	0.625

Table 6: Optimized input probabilities and their implementation for the circuit C2670

As it is shown in /Wu87/ small differences between realized and optimized input probabilities have no significant effect on the test length, since the detection probability of each fault depends on the signal probability of each primary input linearly.

At least 18 of the flip-flops denoted by LR are used for the LR modules of the GURTs, whereas the other flip-flops are forming a LR module implementing a LFSR with maximum period.

In order to minimize the routing overhead the GURTS can be divided. Currently some research is done to minimize the hardware overhead by algorithms searching the best order and the optimal size of the GURTs.

6. Conclusion

A self test architecture was presented generating unequiprobable random patterns based on linear feed back shift registers of type II. Using these modules the class of self testable circuits is enlarged without significant additional hardware costs compared with the conventional BILBO approach.

The self test features can be used during the production test and during system operation while the circuit is idle. The later can be used to support the design of reliable and fail safe system architectures.

Literature:

AgJa84 Jain, S. K.; Agrawal, V.D.: STAFAN: An alternative to fault simulation; in: Proc. 21st Design Automation Conference, 1984

BaSa84 Savir, J.; Bardell, P.H.: On Random Pattern Test Length; in: IEEE, Trans. on Comp., Vol. C-33, No. 6, June 1984

Benn 84 Bennetts, R. G.: Design of Testable Logic Circuits, Addison-Wesley, 1984

BDS84 Savir, J. et al.: Random Pattern Testability; in: IEEE, Trans. on Comp., Vol. C-33, No. 1, Jan. 1984

BRGL85 Brglez, F. et al.: Accelerated ATPG and fault grading via testability analysis; in: Proc. ISCAS'85, Kyoto 1985

Davi80 David, R.: Testing by Feedback Shift Registers; in: IEEE, Trans. on Comp., Vol. C-29, No. 7, July 1980

Golo67 Golomb, S. W.: Shift Register Sequences; Holden-Day, Inc., 1967

Hele83 Heckmaier, J. H.; Leisengang, D.: Fehlererkennung mit Signaturanalyse; in: Elektronische Rechenanlagen, Heft 3, 1983

KuWu85 Kunzmann, A.; Wunderlich, H.-J.: Design automation of random testable circuits; in: Proc. ESSCIRC 1985, Toulouse

KOEN79 Koenemann, B. et al.: Built-In Logic Block Observation Techniques, Proc. Test Conference, Cherry Hill 1979, New Jersey

LBGG86 Lisanke, R. et al.: Testability-Driven Random Pattern Generation; in: Proc. ICCAD, November 1986

McCl85 McCluskey, E.J.: Built-In Self-Test Techniques & Built-In Self-Test Structures; in: IEEE Design & Test, April 1985

SETH86 Seth, S. C. et al.: An Exact Analysis for Efficient Computation on Random-Pattern Testability in Combinational Circuits; in: FTCS 16, 1986

Tsai83 Tsai, M. Y.: Pass Transistor Networks in MOS Technology: Synthesis, Performance, and Testing; in: Proc. IEEE, Symp. of Circuits and Systems, 1983

TI80 The TTL Data Book; Texas Instruments, 1980

WCM86 Wagner, K.; Chin, C., McCluskey, E.: Fault Coverage of Pseudorandom Testing; in Proc. ICCAD-86, IEEE International Conference on Computer-Aided Design, 1986

Will86 Williams, R.M.: IBM Perspectives on the Electrical Design Automation Industry. Keynote Address to the 23rd Design Automation Conference, Las Vegas, 1986

Wu85 Wunderlich, H.-J.: PROTEST: A Tool for Probabilistic Testability Analysis; in: Proc. 22nd Design Automation Conference, Las Vegas, 1985

Wu86 Wunderlich, H.-J.: Probabilistische Verfahren zur Verbesserung der Testbarkeit synthetisierter digitaler Schaltungen; Dissertation an der Fakultät für Informatik der Universität Karlsruhe, 1986

Wu87 Wunderlich, H.-J.: On Computing Optimized Input Probabilities for Random Tests, Proc. 24rd Design Automation Conference, Miami Beach, 1987

WuRo86 Wunderlich, H.-J.; Rosenstiel, W.: On Fault Modeling for Dynamic MOS Circuits; in: Proc. 23rd Design Automation Conference, Las Vegas 1986