Proceedings 18th International Symposium on Fault-Tolerant Computing (FTCS-18), Tokyo, 1988

Generating Pattern Sequences for the Pseudo-Exhaustive Test of MOS-Circuits

Hans-Joachim Wunderlich and Sybille Hellebrand

Institute of Computer Design and Fault-Tolerance
(Prof. Dr. D. Schmid)
Universität Karlsruhe
D-7500 Karlsruhe 1
Federal Republic of Germany

Abstract

In order to ensure a high product quality some authors propose pseudo-exhaustive or verification testing. This is applicable if each primary output of the combinational circuit only depends on a small set of primary inputs, where all possible patterns can be enumerated completely. But in CMOS-circuits even a single stuck-open fault may fail to be detected this way, and the already proposed additional test of each input transition is not sufficient either.

In this paper a method based on linear feedback shift registers over finite fields is presented to generate for a natural number n a pattern sequence with minimal length detecting each m-multiple stuck-open fault for $m \leq n$. A hardware architecture is discussed generating this sequence, and for $n = 1$ a built-in self-test approach is presented detecting all combinations of multiple combinational and single stuck-open faults.

Keywords: Pseudo-exhaustive test, built-in test, stuck-open faults, LFSR, finite fields.

1) Introduction

In recent years much attention has been paid to the so called pseudo-exhaustive or verification test. This test strategy is applicable, if each primary output of the combinational circuits under test only depends on a limited number of primary inputs. In this case for each primary output all possible patterns at the relevant inputs can be enumerated completely, and thus its function is tested exhaustively (fig. 1):

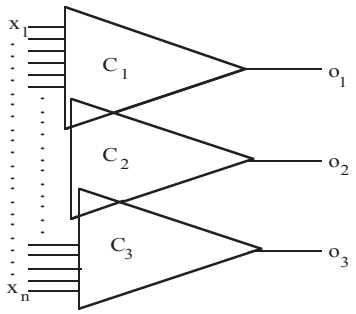

Fig. 1: Pseudo-exhaustively testable circuit with its cones

Not all circuits are testable this way, and they have to be segmented. This is done either by path sensitizing, or directly by hardware. Doing path sensitizing some inputs are set to fixed values, such that only suitable segments are activated. But this approach has the disadvantages of high computing time, since the segmentation problem is np-complete [Arch85], [Pata83], and of an inclomplete fault coverage, since multiple faults and shorts between the segments may not be detected [ArMc84].

On the other hand the hardware segmentation costs some chip area, if for instance multiplexer partitioning is used [McCl85],[BoMc81]. Furthermore some faults within the partitioning circuitry may be undetectable. The later problem is solved using the partitioning technique applied in [Behr87]. The pseudo-exhaustive test of circuits segmented by hardware is often called verification testing [McCl84]. This technique has several advantages:

1) All combinational faults can be detected this way with the exception of some shorts between different segments.

2) There are some techniques known implementing this test strategy as a built-in self-test [WaMc86], [Aker85] by costs comparable to self-test by pseudo-random patterns (for example the BILBO [KOEN79] or the GURT [Wu87]).

But some nMOS-pass-transistor and static CMOS networks are causing further problems, since faults may cause a sequential behavior [Wads78]. For this reason a modified linear feedback shift register (LFSR) approach has already been proposed, generating pattern sequences of length 2 supporting the deterministic self-test [Star84]. In order to test these circuits exhaustively an architecture was presented generating all possible single transitions at the primary inputs [CrKi85]. But here the authors already pointed out that complete fault coverage may only be obtained for irredundant two level networks designed for a robust test.

In section 2 we will discuss the fault coverage and the hardware overhead of both approaches. Furthermore we will establish the requirements for a complete test sequence for MOS circuits, and we will give a lower bound of the length of such a sequence. In section 3 we will show that this lower bound is obtained using a technique based on finite fields. In section 4 we present a hardware architecture generating complete test sequences, which can also be used as a built-in self-test method. Finally section 5 gives an example.

2) Self-test of CMOS circuits

The problems of fault modeling for CMOS gates have widely been discussed ([Wads78], [GALI80], [Chan83]). In presence of a stuck-open fault CMOS gates may show sequential behavior. Without such a fault the circuit of fig. 2 behaves as a NOR-gate.

In presence of the marked fault the output of the gate is prevented from being decharged when the inputs are A=1 and B=0. Because of the capacitive load effect the output retains the previous value, and the gate behaves like a sequential circuit (table 1).

A	B	f	f'_t
1	1	0	0
1	0	0	f'_{t-1}
0	1	0	0
0	0	1	1

Table 1: Fault free and faulty function of the circuit in fig. 2

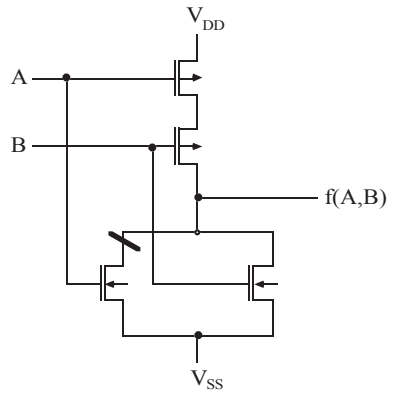

Fig. 2: CMOS-NOR with stuck-open fault

Obviously the fault is not detected if we apply the patterns in the shown order. Such faults are only guaranteed to be tested, if certain pattern sequences are applied, in this case the sequence of (A,B) = (0,0),(1,0). Sequential behavior may also be caused by faults in nMOS-pass-transistor networks, but using dynamic MOS-techniques this can be avoided ([OkKo84], [WuRo86]). For static MOS-techniques Starke proposed a self-test architecture generating pairs of test patterns. For an n-input circuit he uses a feedback shift register of length 2n (fig. 3).

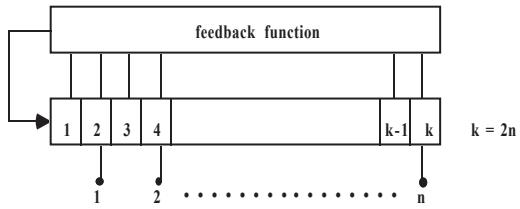

Fig. 3: Generator circuit by [Star84]

Only every second flip-flop is connected to an input of the combinational circuit. Within two clock phases the contents of the flip-flops 2i, i:=1,…,n, form the first pattern, and the flip-flops 2i-1 form the second one. The feedback function is computed by methods described in [DaMu81]. Besides the high computing effort to determine the test patterns and the feedback function, this approach also suffers from rather a high hardware overhead, since the flip-flops are doubled, and the feedback function may become complex.

The fault coverage depends on the quality of the deterministic test set. Furthermore due to delays and hazards the nodes under test may be charged or decharged at times preventing fault detection ([Chan83], [REDD83], [REDD84]). The fault coverage is enhanced if the feedback function represents a primitive polynomial generating 2^{2n}-1 patterns, resulting in a complete transition test. But in this case we have to deal with a long test length besides the hardware overhead by the doubled flip-flops.

In order to enhance the fault coverage and to reduce the test length pseudo-exhaustive adjacency testing was proposed by Craig and Kime ([CrKi85], see fig. 4).

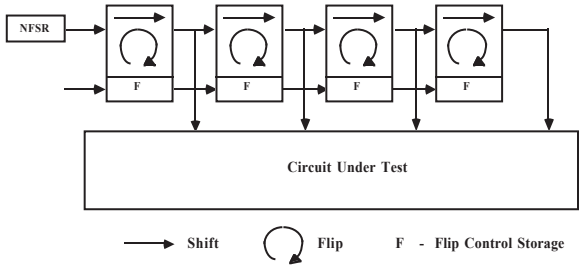

Fig. 4: Pseudo-exhaustive adjacency testing by [CrKi85]

An adjacency test is a pair of patterns differing at a single bit position. It is implemented by a modified LFSR (NFSR) generating all 2^n patterns, and for every pattern sequentially each register bit is flipped. This is controlled by an additional flip control register. This way within $(n+1)2^n$ clocks all single transition tests are generated.

But this approach has a large trade-off between the hardware overhead and the test length. If for instance the circuit of fig. 1 is tested this way, we cannot take advantage of its pseudo-exhaustive testability property, and we have to apply $(n+1)2^n$ patterns. If we tried to test the cones C_1, C_2, and C_3 separately, we would have to implement multiple hardware for some inputs, since the cones are not disjoint in general. Furthermore the authors already pointed out that their approach is not pseudo-exhaustive in its real sense, because there are stuck-open faults which may not be detected by a single transition.

For the general case more patterns have to be applied:

Observation 1: Let $F:\{0,1\}^n \rightarrow \{0,1\}$ be a boolean function, $n \geq 2$. For each pair of patterns $t_1 \neq t_2 \in \{0,1\}^n$ the function F can be implemented by a CMOS circuit F' with the following property: there is a fault possible in F', which is detectable if and only if the pattern sequence t_1, t_2 is applied.

Proof: (Sketch) Start with an arbitrary implementation of F, and add some obvious hardware fulfilling the property that the initializing function of a fault is true exactly at t_1, and the test function at t_2. Control the output of F by this hardware.

Observation 2: An exhaustive test including all single stuck-open faults must include all pattern sequences

$<t_1,t_2> \in \{0,1\}^n x \{0,1\}^n \setminus \{<t,t> \mid t \in \{0,1\}^n\}$.

Notations: Let C be a combinational circuit with primary outputs $O = \{o_1,\ldots,o_m\}$, primary inputs $I := \{i_1,\ldots,i_n\}$, and cones $K = \{k_1,\ldots,k_m\}$. For each $j := 1,\ldots,m$ let I_j be the set of primary inputs of the cone k_j.

Now we can sequentially apply observation 2 to each output $o \in O$. But there is a more general approach possible.

Definition 1: For each $j := 1,\ldots,m$ the projection

$$pr_j: GF(2)^n \rightarrow U,\ GF(2)^n \supset U$$

is defined by $(x_1,\ldots,x_n) \rightarrow (y_1,\ldots,y_n)$ where $y_i := x_i$, if $i \in I_j$, and $y_i := 0$ otherwise.

Definition 2: A pattern set T, $GF(2)^n \supset T$, is pseudo-exhaustive, if for all $j := 1,\ldots,m$, the restricted projections $pr_j|T$ are surjective.

This definition describes, that a test set has to enumerate all possible input patterns for each cone.

Observation 3: If T is a pseudo-exhaustive test set for a multiple-output circuit, then the set of all sequences TxT of length 2 detects all stuck-open faults. Furthermore this set of pairs of patterns guarantees to detect multiple stuck-open faults under the restriction that there is at least one cone with only a single fault.

If there is a cone containing a single stuck-open fault at most one additional state is introduced into the output function. The probability that cones contain multiple faults is reduced using design techniques described in [Koep87]. If we want to consider multiple stuck-open faults within cones or sequential circuits, the length of the sequences must not be restricted to 2.

Let $t := |T|$ the cardinality of the test set. We now want to generate a long pattern sequence of length L containing all sequences of length (fig. 4).

(11)(10)(11)(11)(01).....
|_____|
|_____| Sequences of length 2
|____|
|____|
|_____Sequence of length L__|

Fig. 4: Global sequence containing 2-sequences

Observation 4: The length L of the global sequence is L ≥ +t -1.

Proof: There are t different sequences which have to be generated. The first patterns form the first sequence, and t -1 more sequences have to be generated. In the best case each additional pattern adds one new sequence. Thus at least +t -1 patterns are necessary.

In the next section we summarize the known procedures to determine pseudo-exhaustive test sets T, and we present an algorithm to compute a global sequence containing all sequences of a given length. Furthermore the global length will be minimal, if t := |T| is a power of a prime number.

3) Computing pseudo-exhaustive test sequences for CMOS circuits

Obviously creating short test sequences can be done within two steps. Firstly a minimum sized pseudo-exhaustive test set T has to be determined, and secondly a global sequence of minimum length L must be generated, containing all sequences of length . Some variants of the first problem have already been proven to be np-complete[HiSi82], hence we will rely on some well known heuristics. These heuristics are:

- the straightforward complete enumeration;
- the sequential enumeration with respect to the input set of each cone;
- the extended shift register approach of [BARZ83];
- the dependency matrix approach of [HiSi82], [McCl84];
- the techniques based on linear and cyclic codes (e.g. [VaMa85]);
- the constant weight vector technique [TaWo83];
- the technique of linear sums [Aker85], which is mainly a hardware approach;
- condensed shift register approaches based on cyclic codes ([WaMc84], [WaMc86], [WaMc87]), which are also hardware solutions.

Each of these techniques generates a pseudo-exhaustive test set T of different size. Now we assume that such a test set T is given, and we are looking for a global sequence containing all -sequences.

Let $p^r \geq t := |T|$ be the power of a prime. For practical reasons explained later two cases are important:

1) $p = 2$, and r is the minimal natural number with $p^r \geq t$;
2) $r = 1$, and p is the minimal prime such that $p \geq t$.

If we set $q := p^r$ then it is known from elementary algebra that there is a finite field H := GF(q), and there is a injective function $f: T \rightarrow H$.

We will now construct a sequence of elements of H with length +q -1 containing all -sequences. We have already proven that this is the minimum length. Furthermore using f^{-1} we have a sequence of test patterns containing all -sequences, and in the case of t = q this global sequence is even minimal.

Definition 3:

$$GF(q)[x] := \{\sum_{i=0}^{u} g_i x^i \mid u \in \mathbf{N}, g_i \in GF(q)\}$$

is the polynomial ring over GF(q) in a single variable.

Definition 4: A polynomial $s(x) \in GF(q)[x]$ of degree is called primitive, if it is irreducible, and the zeros of s in GF(q) are primitive (q -1)-th roots of unitiy.

Definition 5: A feedback shift register Φ over GF(q) is a mapping

$$\Phi: GF(q) \rightarrow GF(q)$$
$$(y \quad,\ldots,y_0) \rightarrow (R(y \quad,\ldots,y_0), y \quad,\ldots,y_1)$$

R is called feedback function.

A special case is a linear feedback function

$$R(y_{-r},\ldots,y_0) := \sum_{i=0}^{-1} c_i y_i$$

Here we say that the polynomial

$$s(x) := x - \sum_{i=0}^{-1} c_i x^i$$

represents the linear feedback shift register (LFSR).

Definition 6: The period of a feedback shift register Φ defined on U is the smallest number $\pi \geq 1$ such that $\Phi^\pi|U = id|U$.

Theorem 1: A feedback shift register over GF(q) with a primitive polynomial $s(x) \in GF(q)[x]$ of degree representing the feedback function has maximum period $\pi = q - 1$.

Proof: See [LiNi86], [Lüne79] e.g.

Based on the following theorem we can compute global sequences of minimum length, and can implement them by hardware:

Theorem 2: Let p be a prime number, $r \in \mathbf{N}\backslash\{0\}$, and $q := p^r$. For each $\in \mathbf{N}\backslash\{0\}$ there is a primitive polynomial $s(x) \in GF(q)[x]$ of degree .

Proof: A proof is found in [LiNi86].

Remark: The π-th cyclotomic polynomial Q_π over GF(q) (π=q -1) is the product of all primitive polynomials of degree over GF(q). Thus primitive polynomials of degree can be determined by applying one of the well known factorization-algorithms to Q_π (confer [LiNi86]).

Up to now theorem 1 and theorem 2 provide a method to compute a sequence of length q -1, containing q -l different -sequences. Now we set:

$a_0,\ldots,a_{\quad 2} := 0 \in GF(q)$;
$a_{-1} \in GF(q), \neq 0$;
$a_{+i} := R(a_{+i-1},\ldots,a_i)$, i := 0,…,q -2.

Obviously in $<a_i>$ all -sequences are generated, furthermore it is a minimal sequence having this property, since it has length +q -1. This sequence should be decoded by f^{-1} into test patterns, and if one wants to generate 2-sequences for stuck-open tests, for each pair of identical patterns one has to be omitted.

Summarizing the complete test sequences are computed by the following procedure:

1) Determine a pseudo-exhaustive test set T.
2) Choose a finite field H := GF(q) with $q \geq t := |T|$.
3) Encode the test patterns by elements of H, i.e. find an elementary, injective mapping $f:T \rightarrow H$.
4) Construct a primitive polynomial $s(x) \in GF(q)[x]$ of degree .
5) Construct the modified shift register sequence $<a_i>$ using the polynomial s(x).
6) Decode the sequence $<a_i>$ into a sequence of test patterns.

4) Generating complete test sequences by hardware

It is a well known built-in self-test technique to generate pattern sequences by a LFSR over GF(2) (fig. 5).

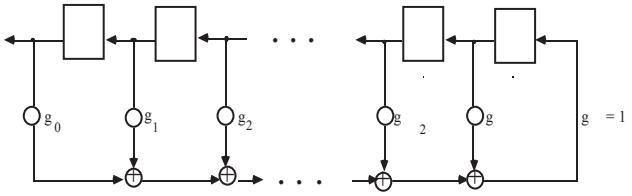

Fig. 5: Standard LFSR over GF(2)

The feedback function is determined by the polynomial

$$g(x) := g\ x + g_{-1}x^{-1} + \ldots + g_1x + g_0 \in GF(2)[x],$$

and since there are only two types of constants, 0 and 1, the multiplication in GF(2) is expressed by the presence or absence of feedback lines. In an arbitrary finite field GF(q) an LFSR is more complicated since we have to multiply by some constants (fig. 6).

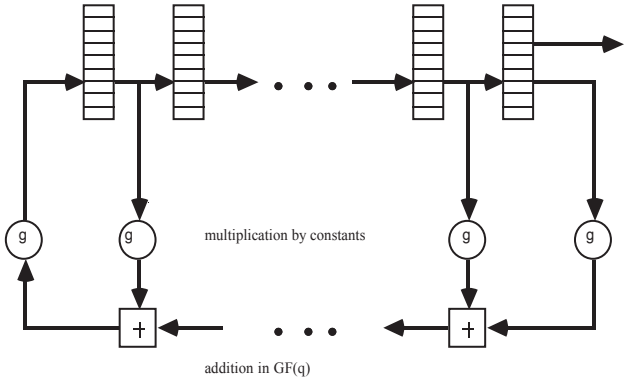

Fig. 6: An LFSR over GF(q)

The LFSR of fig. 6 represents the feedback polynomial

$$g(x) := g\ x + g_{-1}x^{-1} + \ldots + g_1x + g_0 \in GF(q)[x].$$

For the sake of simplicity we will only discuss the most important case = 2. Firstly we normalize the polynomial g(x), that is g := 1, by dividing. Then the LFSR of fig. 6 turns into an architecture as shown in fig. 7.

As a difference to other CMOS BIST-techniques discussed in section 2, there are no doubled flip-flops for test reasons. We only have to ensure that CUT1 and CUT2 are tested by the same set T. The circuit of fig. 7 is a general architecture for GF(q), but now we will explain further details only for the two cases $q = p \geq t$ and $q = 2^r \geq t$. Furthermore these are the cases which are feasable by the known algorithms of computer algebra.

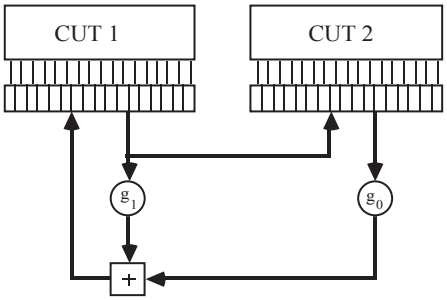

Fig. 7: LFSR of length 2 over GF(q)

Now we will discuss the implementation of multiplication by constants and of the addition in GF(q).

a) $q = 2^r$

We assume that the test patterns enumerate at r bit positions the complete field GF(q), otherwise we have to code and encode. GF(q) is a linear space over GF(2), and the additon in GF(q) can be done by addition of components. This is implemented by XOR-gates, and hence we need r XOR-gates for our 2-stage-register of fig. 7.

In order to explain the multiplication, a little more theoretical background is needed. In a finite field multiplication establishes a cyclic group, in our case of cardinality 2^r-1. Let $\Phi:GF(2)^r \rightarrow GF(2)^r$ be a shift register according to definition 5, with a linear feedback function

$$R(x_{r-1},\ldots,x_0) := \sum_{i=0}^{r-1} c_i x_i$$

By theorems 2 and 3 there are linear feedback functions R, such that Φ has the period 2^r-1.

Let c be a primitive element of the cyclic group of multiplication in H := GF(2^r). If 2^r-1 is a prime number, all elements except 0 and 1 will do. For x, $y \in H$ define the operation $*_c$ on $H\backslash\{0\}$ by

$x *_c y = z :\Leftrightarrow x = c^k$ and $z = \Phi^k(y)$ for a $k \leq 2^r-1$.

Theorem 3: $(H,+,*_c)$ is a finite field.

Proof: Left to the reader.

Now multiplication by a constant, that is a primitive element, can be expressed as a single shift by Φ. If 2^r-1 is prime, all elements of $GF(2^r)\backslash\{0,1\}$ are primitive too. Hence g_0 is primitive, and we set $g_0 *y := \Phi(y)$. We have $g_1 = g_0^k$ for some k and $g_1 *y = \Phi^k(y)$. It is an open problem, whether it is possible that neither g_0 nor g_1 can be primitive. But in this case there is a primitive c with $c^{k1} = g_1$, and $c^{k0} = g_0$, and we have $g_1 *y = \Phi^{k1}(y)$ and $g_0 *y = \Phi^{k0}(y)$.

Example: Let be r := 6, let g_0 be primitive, and $g_1 := g_0^2$. x^6+x+1 is a primitive polynomial over GF(2), and we have $\Phi(x_5,\ldots,x_0) = (x_0+x_1,x_5\ldots,x_1)$. This is implemented by

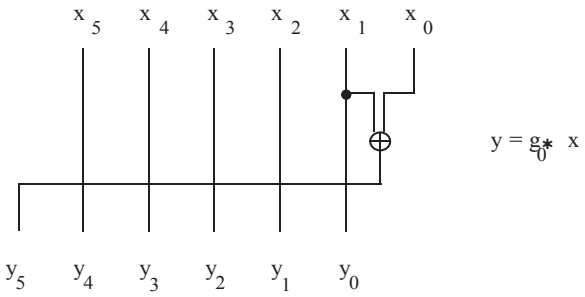

Fig. 8: Implementation of the multiplication $y = g_0 * x$

Furthermore we have $g_1*(x_5,\ldots,x_0) = (x_1+x_2,x_0+x_1,x_5,\ldots,x_2)$. The overall feedback function is as shown in fig. 9 using $g_0*(x_5,\ldots,x_0)+g1*(x_{11},\ldots,x_6) = g_0*(((x_5,\ldots,x_0)+g_0*(x_{11},\ldots,x_6))$

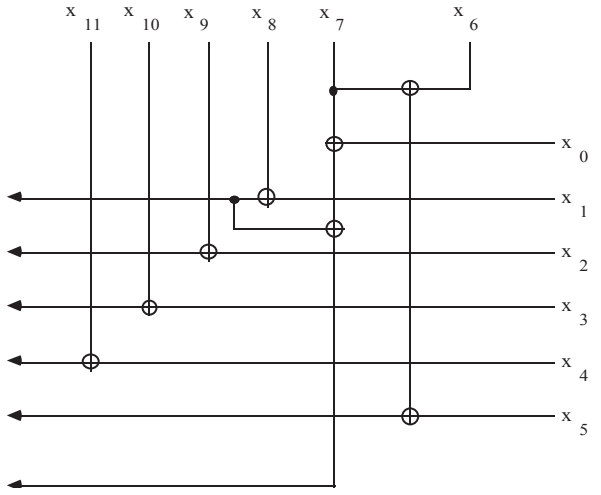

Fig. 9: Hardware implementation of $g_0*(x_5,\ldots,x_0)+g1*(x_{11},\ldots,x_6)$

Summarizing we have r XOR-gates for the addition, and also some XOR gates for the multiplication, in the whole this is comparable to the usual linear feedback functions of pseudo-random patterns.

b) q = p, p prime

In this case the field is isomorphic to $\boldsymbol{Z}/q$, and both addition and multiplication have to be done modulo p. A straightforward implementation of the function can be simplified by means of boolean minimization, described for instance in [Bray84]. But the use of a prime q seems to be more suitable for an external test. Using a CMOS specific scan design, an external chip can generate the pseudo-exhaustive pattern sequences (fig. 10). This approach is already well known for pseudo-random patterns [EiLi83]. In order to use the external chip in a most general way, the cardinality of the field should be programmable.

Summarizing the design of a BIST-hardware can be done by proceeding as follows:

1) Choose a pseudo-exhaustive test set T.

2) Select an r with $q := 2^r \geq |T|$, or a prime number p with $q := p \geq |T|$.

3) Find an encoding function $f:T \rightarrow GF(q)$.

4) Find a primitive polynomial over GF(q) of degree 2, i. e. two constants g_0 and g_1.

5) a) For $q = 2^r$ determine addition and multiplication by theorem 3;

b) for q = p implement addition and multiplication modulo p.

6) Implement the decoding function f^{-1}.

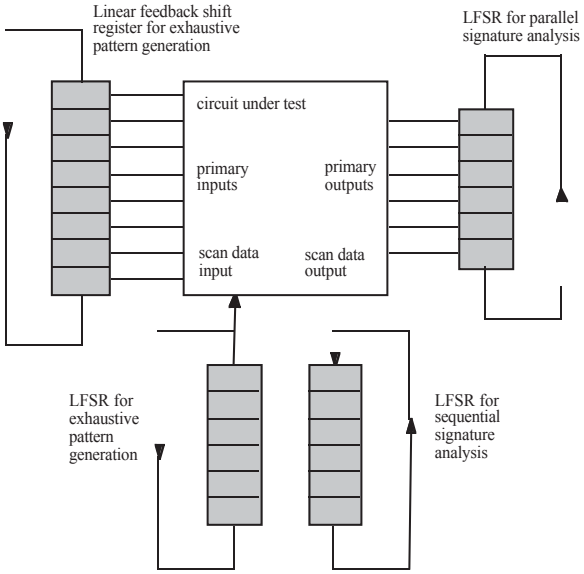

Fig. 10: Pseudo-exhaustive test by an external chip

5) Example

Let S be a combinational circuit with 16 inputs $x_1,\ldots,x_{16}$ and 8 outputs $o_1,\ldots,o_8$, where

$o_1 = f_1(x_2,\ldots,x_8)$
$o_2 = f_2(x_1,x_3,\ldots,x_8)$
$o_3 = f_3(x_1,x_2,x_4,\ldots,x_8)$
$o_4 = f_4(x_1,\ldots,x_3,x_5,\ldots,x_8)$
$o_5 = f_5(x_{10},\ldots,x_{16})$
$o_6 = f_6(x_9,x_{11},\ldots,x_{16})$
$o_7 = f_7(x_9,x_{10},x_{12},\ldots,x_8)$
$o_8 = f_8(x_9,\ldots,x_{11},x_{13},\ldots,x_8)$

Each cone depends on 7 primary inputs. The first 4 cones are tested by a set enumerating all patterns from $(x_1,\ldots,x_7)$ and setting $x_8 := x_1+x_2+x_3+x_4$ (method of linear sums by [Aker85]). For the last 4 cones we have to enumerate $(x_9,\ldots,x_{15})$ and set $x_{16} := x_9+x_{10}+x_{11}+x_{12}$. Since both subcircuits are tested by the same sets of patterns, we only consider the first 4 cones, and call this subcircuits C1. Its test set is

$T := \{(x_1,\ldots,x_8) \mid (x_1,\ldots,x_7) \in GF(2)^7 \;\&\; x_8 := x_1+x_2+x_3+x_4\}$. The encoding function is $f:T \rightarrow GF(2)^7$, $(x_1,\ldots,x_8) \rightarrow (x_1,\ldots,x_7)$.

A primitive polynomial can be derived by means given in [LiNi86], were $s(x) := x^2+g_1x+g_0$. Since 127 is prime, both g_1 and g_0 are primitive in the cyclic group of multiplication. The most suitable coefficients we find are $g_1 := g_0^2$. In order to implement multiplication by g0, we have to look for a primitive polynomial of degree 7 in GF(2), for instance x^7+x+1. Thus multiplication by g_0 is

$$g_0(x_7,\ldots,x_1) = (x_1+x_2,x_7,\ldots,x_2).$$

The decoding function is $f^{-1}(x_7,\ldots,x_1)=(x_1,\ldots,x_7,x_1+x_2+x_3+x_4)$, and the overall feedback function looks like fig. 11. Since the last four cones, C2, are tested in the same way, the whole circuit with 16 inputs is tested exhaustively for all stuck-open faults within 2^{14} clocks.

Another often cited example is the parity generator TI SN54/74LS630 with 23 inputs. Here each cone only depends on 10 inputs, and there is a pseudo-exhaustive test set of 2^{10} patterns [McCl84]. In the same way as presented above, a BIST structure can be designed, testing two circuits with 46 primary input exhaustively for stuck-open faults within 1M clocks. The reader may prove his understanding of the presented concepts, by doing this as an exercise.

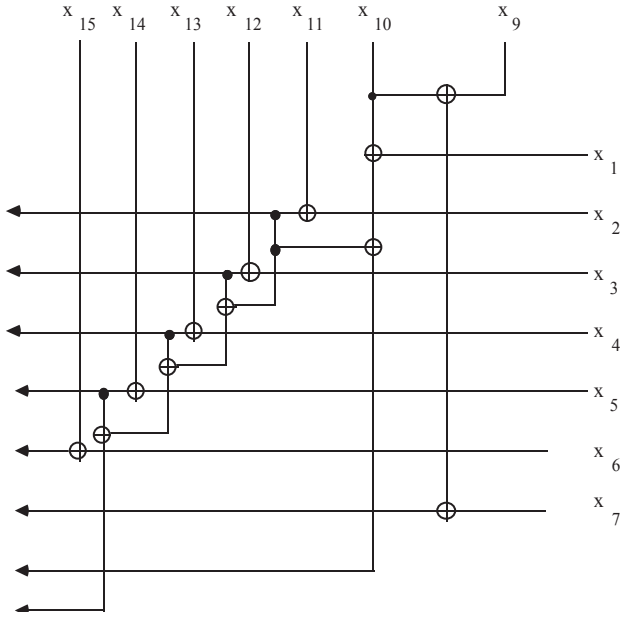

Fig. 11: Feedback function for the example

6) Further research

Mainly two points need further investigations. Firstly we are looking for an optimal design of an external chip as discussed in section 5 b). An optimal design should include the possibility to generate patterns based on many different prime numbers p.

The second problem is the evaluation of the circuit responses. The presented test circuitry can be considered as a signature register over GF(q), and one has to investigate the fault coverage achieved by such a register. Here the work should be generalized, which was done for signature registers over GF(2) by [WILL87] or [AgIv87].

7) Conclusions

A technique has been presented generating complete test sequences for CMOS circuits. The sequences are of minimum length, and can be produced either by software, by an external chip, or by a BIST-structure. Doing the latter the hardware overhead would be of the same magnitude as a conventional pseudo-random architecture.

Literature

AgIv87 Agarwal, V.K.; Ivanow: On a Fast Method to Monitor the Behaviour of Signature Analysis Registers; in: Proc. IEEE International Test Conference 1987

Arch85 Archambeau, E.: Network Segmentation for Pseudo-Exhaustive Testing; CRC Technical Report No. 85-10, 1985, Stanford

ArMc84 Archambeau, E.C.; McCluskey, E.J.: Fault Coverage of Pseudo-Exhaustive Testing; in: Proc. International Symposium on Fault Tolerant Computing, FTCS-14, 1984

Aker85 Akers,S.B.: On the Use of Linear Sums in Exhaustive Testing in: Proc. International Symposium on Fault Tolerant Computing, FTCS-15, 1985

Behr87 Behrens,B.: Entwurf von Grundzellen für einen Silicon Compiler zur Unterstützung des pseudo-erschöpfenden Tests; Diplomarbeit an der Universität Karlsruhe, Fakultät für Informatik

BoMc81 McCluskey, E.J.; Bozorgui-Nesbat, S.: Design for Autonomous Test in: IEEE Trans. Comp., Vo. C-30, No. 11, 1981

Bray84 Brayton, R. K.; Hachtel, G. D.; McMullen, C. D.; Sangiovanni-Vincentelli: Logic Minimization Algorithms for VLSI Synthesis; Kluwer Academic Publishers, 1984

BARZ83 Barzilai, Z. et al.: Exhaustive Generation of Bit Patterns with Applications to VLSI Self-Testing; in: IEEE Trans. on Comp., Vol. C-32, No. 2; Dec. 1983

Chan83 Chandramouli, R.: On Testing Stuck-Open Faults; in: Fault Tolerant Computing Symp. FTCS-13, 1983

CrKi85 Craig,G.L., Kime,C.R.: Pseudo-Exhaustive Adjacency Testing: A BIST Approach for Stuck-Open Faults; in: Proc. IEEE International Test Conference, 1985

DaMu81 Daehn, W.: Deterministische Testmustergenerierung für den eingebauten Selbsttest von integrierten Schaltungen; in: Großintegration, NTG-Fachberichte 82, 1983 Baden-Baden

EiLi83 Eichelberger, E.B.; Lindbloom, E.: Random-Pattern Coverage Enhancement and Diagnosis for LSSD Logic Self-Test; IBM J. Res. Develop., Vol. 27, No. 3, May 1983

GALI80 Galiay,J. et al.: Physical vs. Logical Fault Models in MOS LSI Circuits, Impact on their Testability; in: IEEE Trans. Comp., Vol. C-29, No. 6, June 1980

HiSi82 Hirose, F.; Singh, V.: McDDP, A Program for Partitioning Verification Testing Matrices; CRC Technical Report No. 81-13, Stanford, 1982

Koep87 Koeppe, S.: Optimal Layout to Avoid CMOS Stuck-Open Faults in: Proc. 24th Design Automation Conference, Miami Beach 1987

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

LiNi86 Lidl, R.; Niederreiter, H.: Introduction to finite fields and their applications; Cambridge University Press, Cambridge 1986

Lüne79 Lüneburg, H.: Galoisfelder, Kreisteilungskörper und Schieberegisterfolgen; Mannheim, Wien,Zürich: Bibliographisches Institut, 1979

McCl84 McCluskey,E.J.: Verification Testing - A Pseudoexhaustive Test Technique; in: IEEE Trans. Comp., Vol. c-33, No.6, June 1984

OkKo84 Oklobdzija, V. G.; Kovijanic, P. G.: On Testability of CMOS-Domino Logic; in: Proc. 14th Int. Conf. on Fault-Tolerant Computing, FTCS-14, 1984

Pata83 Patashnik,O.: Circuit Segmentation for Pseudo-Exhaustive Testing CRC Technical Report No. 83-14, 1983 Stanford

REDD83 Reddy, S.; Reddy, M.; Kuhl, J.: On Testable Design for CMOS Logic Circuits; in: Proc. IEEE International Test Conference, 1983

REDD84 Reddy, S.; Reddy, M.; Agrawal, V.: Robust Tests for Stuck-Open Faults in CMOS Combinational Logic Circuits; in: Proc. International Syposium on. Fault-Tolerant Computing, FTCS-14, 1984

Star84 Starke, C.W.: Built-In Test for CMOS Circuits; in: Proc. IEEE International Test Conference, 1984

TaWo83 Tang, D.T.; Woo, L.S.: Exhaustive Test Pattern Generation with Constant Weight Vectors; in: IEEE Trans. on Comp., Vol. C-32, No. 12, 1983

VaMa85 Vasanthavada,N., Marinos,P.N.: An Operationally Efficient Scheme for Exhaustive Test-Pattern Generation Using Linear Codes; in: Proc. IEEE International Test Conference, 1985

Wads78 Wadsack, R.L.: Fault Modeling and Logic Simulation of CMOS and MOS Integrated Circuits; in: The Bell System Technical Journal, Vol. 57, No. 5, May-June 1978

WaMc84 Wang, L.-T.; McCluskey, E.J.: A New Condensed Linear Feedback Shift Register Design for VLSI/System Testing; in: Proc. International Symposium on Fault Tolerant Computing, FTCS-14, 1984

WaMc86 Wang, L.-T.; McCluskey, E.J.: Circuits for Pseudo-Exhaustive Test Pattern Generation; in: Proc. IEEE International Test Conference, 1986

WaMc87 Wang,L.T.; McCluskey,E.J.: Circuits for Pseudo-Exhaustive Test Pattern Generation Using Shortened Cyclic Codes; in: Proc. IEEE International Conference on Computer Design, ICCD '87, 1987

Wu87 H.-J. Wunderlich: Self Test Using Unequiprobable Random Patterns; in: International Symposium on Fault-Tolerant Computing, FTCS-17, 1987, Pittsburgh

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

WILL87 Williams, T.W. et al.: Aliasing Errors in Signature Analysis Registers; in: IEEE Design & Test, April 1987

This research was supported by the BMFT (Bundesministerium für Forschung und Technologie) of the Federal Republic of Germany under grant NT 2809 A 3