The Pseudo-Exhaustive Test of Sequential Circuits

Hans-Joachim Wunderlich, Sybille Hellebrand

University of Karlsruhe
Institute of Computer Design and Fault Tolerance
(Prof. Dr. D. Schmid)
Postfach 6980, D-7500 Karlsruhe 1
Federal Republic of Germany

Abstract:

The concept of a pseudo-exhaustive test for sequential circuits is introduced in a similar way as it is used for combinational networks. Instead of test sets one has to apply pseudo-exhaustive test sequences of a limited length, which provides well-known benefits as far as fault-coverage, self-test capability and simplicity of test generation are concerned.

Design methods are presented for hardware segmentation which ensure that a pseudo-exhaustive test is feasible. Example circuits show that the presented test-strategy requires less additional silicon area than a complete scan path.

Keywords: Pseudo-exhaustive test, sequential circuits, design for testability.

1. Introduction

In [McBo81], [McCl84] the pseudo-exhaustive test has been proposed in order to reduce the costs of test pattern generation and test application. For a primary output o of a combinational circuit, the cone C_o is the subcircuit containing all predecessors of o (figure 1). A cone is tested by applying all possible patterns at its primary inputs. The total number of all these patterns is smaller than an exhaustive test, if the cones are sufficiently small.

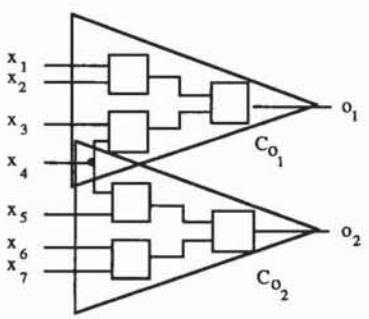

Figure 1: Cones of outputs o_1 and o_2.

An obvious advantage of this test strategy is the high fault coverage, within a cone all combinationally faulty functions are detected. A faulty sequential behavior induced by stuck-open faults can be detected by applying special pattern sequences as described in [WuHe88]. Moreover the pseudo-exhaustive test sets can be generated by special feedback shift-registers [WaMc86], [Aker85], [Udel86], which may be used as a self-test technique or for an external low-cost test. A similar approach is possible for CMOS-faults [WuHe88].

In this paper, we extend the approach to sequential circuits. Using Roth's notation of time frames, a sequential circuit is transformed into a combinational representation [Roth78]. Its size increases linearly with respect to the circuit size if the data-flow graph of the circuit does not contain any cycles ([Wu89], [Kunz89]). Often the data-flow part of the circuit is acyclic by itself, otherwise some flipflops must be included into a partial scan path ([Tris83], [Agra88], [Kunz89]). To obtain a pseudo-exhaustive test, we generate a pseudo-exhaustive test set for the combinational representation, and transform these pattern sets into the respective sequences for the original sequential circuit.

A pseudo-exhaustive test of the combinational representation is not applicable, if a primary output depends on a very large number of primary inputs. In the approach presented, this problem is solved by hardware segmentation, where additional segmentation cells are used to logically disconnect some circuit lines in the test mode.

A uniform technique is presented integrating the partial scan design and the segmentation of a sequential network in a similar way as proposed in [HeWu88]. Examples show that the additional silicon area needed for the partial scan path and the segmentation cells together is less than the overhead for a complete scan path. As an additional advantage we have complete fault coverage without expensive test pattern generation.

After this introductory section, we sketch some basic graph theoretical definitions and facts, which are necessary for our way of circuit modeling. In section 3 we present the cells

necessary for the design of pseudo-exhaustively testable circuits. Besides the well-known LSSD-latches, these are the segmentation cells already mentioned. In section 4 we discuss placement-algorithms which make a pseudo-exhaustive test feasible using a minimum number of these cells.

In section 5 we discuss pseudo-exhaustive test sequences. The sequences can be generated by linear feedback shift-registers (LFSR) in a similar way as proposed for combinational networks [BARZ83]. Finally we present some examples.

2. Circuit modeling and restrictions

We assume that the sequential circuits are described at gate level, and that the following restrictions are fulfilled:

- The circuits are purely synchronous.
- Only D-flipflops are used.
- The D-flipflops can be augmented according to the rules of either level-sensitive or edge triggered scan-design (LSSD, ETSD).

Such a circuit is modeled by a graph:

Definition 2: A circuit graph G := (V,E) is a directed graph with vertices V and edges $E \subset V^2$. $V := V_C \cup V_S \cup I$ is a disjoint union of combinational vertices V_C, sequential vertices V_S and inputs I.

The outputs of the gates are represented by V_C, the outputs of the flipflops are represented by V_S, and I contains both the primary and the pseudo-primary inputs. The pseudo-primary inputs correspond to the flipflops within the scan-path. An example circuit and its circuit graph are shown in figures 2 and 3, respectively.

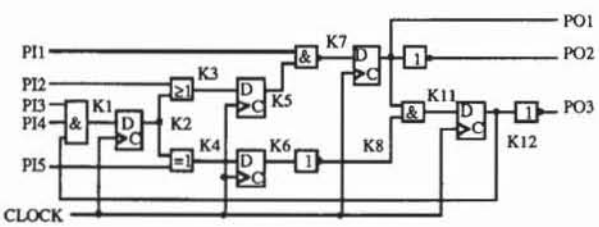

Figure 2: Example circuit.

The circuit graph G = (V,E) consists of the nodes V := {PI1,..., PI5, K1,..., K8, K11, K12, P01, P02, P03} and the corresponding edges.

The vertices of this circuit graph are partitioned into the three sets
V_C = {K1, K3, K4, K7, K8, K11, P02, P03},
V_S = {K2, K5, K6, P01, K12}, and
I = {PI1, PI2, PI3, PI4, PI5}.

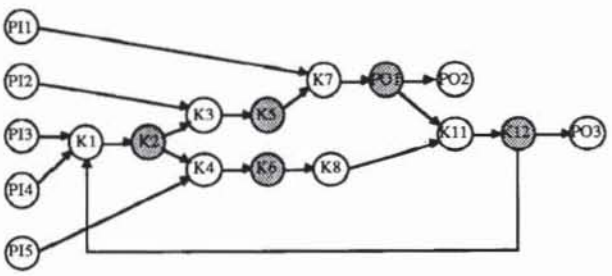

Figure 3: Circuit graph.

In general, we have (v,w) ∈ E, if node v is input of a component, gate or flipflop with output node w. The primary outputs are a subset O⊂ V. For the example circuit, we have O := {P01, P02, P03}.

Definition 2: Let G := (V,E) be a circuit graph and v∈ V. pd(v) := {w∈ V | (w,v)∈ E} is the set of *direct predecessors* of v, and sd(v) := {w∈ V | (v,w)∈ E} is the set of *direct successors*.

Definition 3: A circuit graph G := (V,E) is called *consistent*, if I = {v∈ V | pd(v) = Ø}.

We only deal with consistent circuit graphs.

Definition 4: Let u, v∈ V. A *path* ω from u to v is a sequence of vertices $k_0,\ldots,k_n$, with k_0=u, k_n=v and $(k_{i-1},k_i) \in E$ for i=1,...,n. n is called the length $\ell(\omega)$. If $k_0 = k_n$ ω is called a *cycle*.

Definition 5: Let G := (V,E) be a circuit graph, let v∈ V. p(v) := {w∈ V | there is a path from w to v} is the set of *predecessors* of v, s(v) := {w∈ V | there is a path from v to w} the set of *successors*.

Only the topology of the storage elements V_S determines the test length. It is described by the so-called S-graph:

Definition 6: Let $G^{Ci} := (V^{Ci},E^{Ci})$ be a circuit graph with $V^{Ci} := V_C^{Ci} \cup V_S^{Ci} \cup I^{Ci}$ and O^{Ci}. Its *S-graph* $G^S := (V^S,E^S)$ is defined by:

a) $V^S := O^{Ci} \cup V_S^{Ci} \cup I^{Ci}$

b) $E^S := \{(v,w) \in V^S \times V^S \mid$ There is a path ω from v to w in G^{Ci}, and $\omega \cap V^S = \{v,w\}\}$

Figure 4 shows the S-graph corresponding to the circuit graph of figure 3.

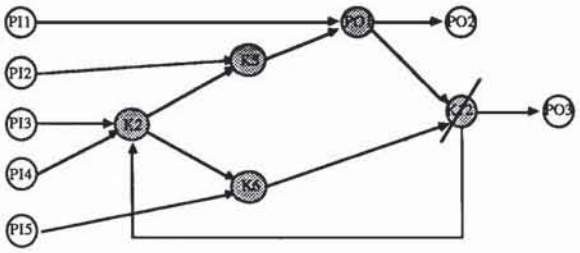

Figure 4: S-graph.

The approach presented is valid for circuits, where the S-graph does not contain any cycles. Every S-graph can be made acyclic by integrating some of the flipflops into an incomplete scan-path. For instance, if in the example circuit flipflop K12 were a scan path element, then the resulting circuit graph would be acyclic (figure 5).

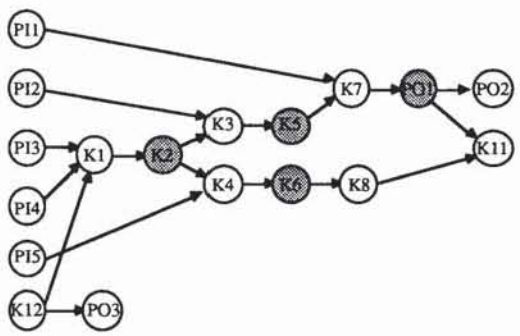

Figure 5: Acyclic circuit graph.

The pseudo-exhaustive test of sequential circuits requires the application of pattern sequences instead of single patterns. Using Roth's notation of time-frames, copies of the combinational part of the circuit are generated, and the number of time-frames corresponds to the length of the test sequences. We modify this approach, such that at each time step we only copy the small part of the combinational circuit that is actually needed for fault detection. In order to describe our solutions exactly, some more graph-theoretical definitions are required:

Definition 7: Let G := (V,E) be an acyclic graph, let $v \in V$ be a node. $rf(v) := \max\{\ell(\omega) \mid \omega$ is a path in G with end point v$\}$ is called *forward-rank* of v, and $rb(v) := \max\{\ell(\omega) \mid \omega$ is a path in G with start point v$\}$ is called *backward-rank*.

Definition 8: Let G := (V,E) be an acyclic graph. The *rank* of G is defined by $rank(G) := \max_{v \in V}\{rf(v)\} = \max_{v \in V}\{rb(v)\}$.

Definition 9: Let G := (V,E) be an S-graph with sequential nodes V_S, outputs O and inputs I. Its back-trace function P is $P: \mathcal{P}(V_S \cup I) \rightarrow \mathcal{P}(V_S \cup I)$; $P(W) := \bigcup_{w \in W} pd(w)$.

The nodes of a subset $W^t \subset V$ have defined values at time step t, if the nodes $W^{t-1} := P(W^t)$ have defined values at time step t-1, and we can use this notation for state back-tracing.

Observation 1: Let G := (V,E) be an acyclic S-graph with rank(G) = r. Then $P^r(V_S \cup I) \subset I$.

Corollary: Every state is reachable within r steps, if it is reachable at all.

Definition 10: Let $G^S := (V^S,E^S)$ be an acyclic S-graph with rank r, and let $G^{Ci} := (V^{Ci},E^{Ci})$ be its circuit graph. Set

$W^r := \{v \in V^S \mid sd(v) \cap O \neq \emptyset$ in $G^S\}$,

$V^r := \{v \in V^{Ci} \mid \exists u_0 \in W^r \exists u_1 \in O$ (v is member of a path ω from u_0 to u_1 and $\omega \cap V^S = \{u_0,u_1\})\} \cup W^r \cup O$,

and for $0 \leq t < r$:

$W^t := P(W^{t+1})$,

$V^t := \{v \in V^{Ci} \mid \exists u_0 \in W^t \exists u_1 \in W^{t+1}$ ($v \neq u_1$ is member of a path ω from u_0 to u_1 and $\omega \cap V^S = \{u_0,u_1\})\} \cup W^t$.

The *combinational representation* of G^{Ci} is the graph

$\bar{G} := (\bar{V},\bar{E})$, where

$\bar{V} := \bigcup_{0 \leq t \leq r} V^t \times \{t\}$

$\bar{E} := \bigcup_{0 \leq t \leq r} \{((x,t),(y,t)) \mid (x,y) \in V^t \times V^t \cap E\} \cup$

$\bigcup_{0 \leq t \leq r} \{((x,t),(y,t+1)) \mid x \in V^t \wedge y \in W^{t+1} \wedge (x,y) \in E\}$

and

$\bar{V}_C := \bigcup_{0 \leq t \leq r} \{(x,t) \in \bar{V} \mid x \in V_C^{Ci} \cup V_S^{Ci}\}$,

$\bar{I} := \{(x,t) \in \bar{V} \mid x \in I\}$,

$\bar{O} := \{(o,r) \mid o \in O\}$.

It should be noted, that all flipflops are mapped to combinational buffers. For the example circuit graph of figure 5 the combinational representation is shown in figure 6. This straightforward construction provides our basic theorem:

Theorem 1: Let G := (V,E) be an acyclic circuit graph, with rank r, and let $\bar{G} := (\bar{V},\bar{E})$ be its combinational representation. A pattern sequence $<< b_i^t \in \{0,1\} \mid i \in I > \mid 0 \leq t \leq r >$ detects a given fault of a node $v \in V$ exactly at time r, if and only if in $\bar{G}$ the corresponding multiple fault of the nodes (v,t), $0 \leq t \leq r$ if defined, is detected by the pattern $<b_i^t \mid (i,t) \in \bar{I} >$.

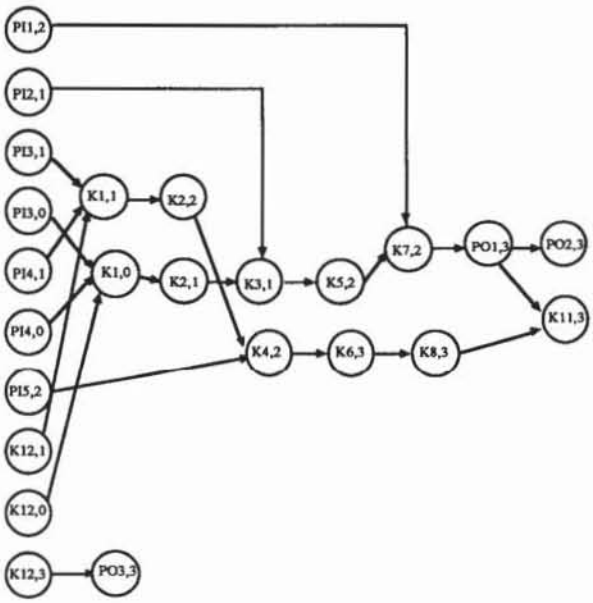

Figure 6: Combinational representation.

If a cone C_o of the combinational representation has ℓ primary inputs, it is tested exhaustively by 2^ℓ patterns. Each pattern is mapped to a sequence of the maximal length r in the original circuit. Thus the length of the pseudo-exhaustive test sequence is bounded by $r \cdot 2^\ell$. In section 5 we discuss some further compactions.

This approach is applicable to all fault models concerning the combinational function of a single or of multiple nodes. If the design is irredundant, a complete fault coverage is obtained. Faults affecting the topology of the S-graph are not guaranteed to be detected. Bridges might connect various cones, and they are hard to detect in purely combinational circuits, too [ArMc84].

3. Devices supporting the pseudo-exhaustive test

A pseudo-exhaustive test is only feasible, if the corresponding S-graph is acyclic, and if each cone of the combinational representation only has a limited number of inputs. The first condition can be satisfied by integrating some flipflops or latches into a partial scan path P extending the well-known LSSD-rules [EiWi77]. This results in the circuit structure of figure 7. In order to keep the hardware overhead small, the size of P should be minimal.

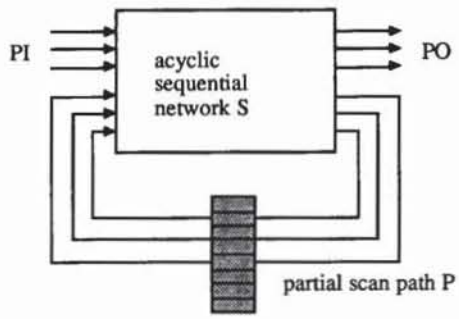

Figure 7: Partial scan design.

In order to fulfill the second requirement, some nodes within the sequential network have to be cut such that they are directly accessible. This way new pseudoprimary inputs p_i and outputs p_o are introduced to replace a cut node v (figure 8).

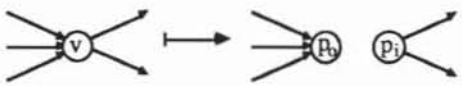

Figure 8: Cut of a node v.

If node v corresponds to a latch in the original network, it is cut easily by its integration into the partial scan path. For the general case multiplexer partitioning has been proposed originally [McBo81], which has some serious drawbacks with regard to area, speed, test control, and fault coverage [HeWu88]. They are avoided by the use of segmentation cells (figure 9).

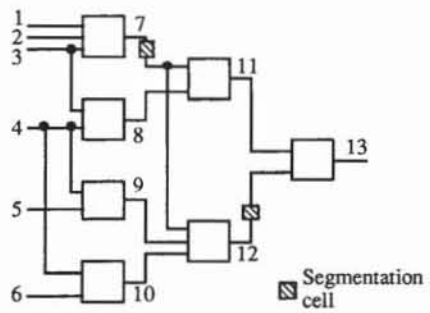

Figure 9: Hardware segmentation by special cells.

In [Bhat86] unmodified latches have been proposed for segmentation purposes. But this alters the clocking scheme, and the speed of the entire circuit is slowed down. For this reason, we use the more sophisticated cell shown in figure 10. In system mode S = 1 is asserted so that D and Q are directly connected. For S = 0 the cell works like the usual LSSD L1(L2*)-latch with data-input D, clock CLK, shift input SDI and shift-clock A(B).

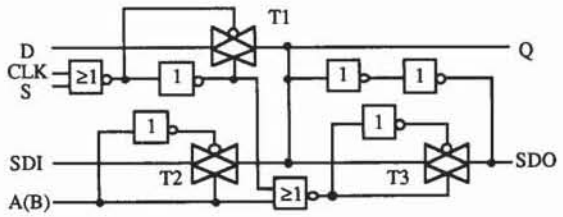

Figure 10: Segmentation cell.

These cells are added to the partial scan path, but they do not affect the system operation of the circuit (figure 11).

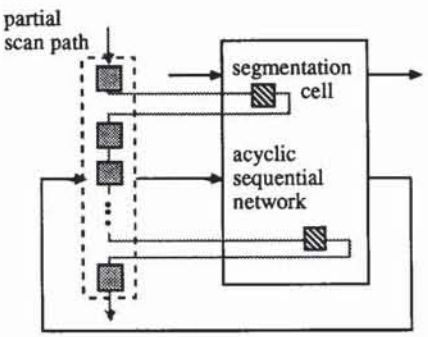

Figure 11: Integrating segmentation and scan design.

In the next section, we discuss how to place the directly accessible latches and segmentation cells.

4. Design algorithms

The following modifications of a design are required to realize a pseudo-exhaustive test strategy:

a) A small number of latches must be integrated into a partial scan path in order to obtain an acyclic S-graph.

b) A minimal number of lines within the original circuit must be cut, in order to obtain small sets of inputs of the cones within the combinational representation. Since the integration of an existing latch into the scan path requires less hardware overhead than adding a new segmentation cell, cutting nodes corresponding to latches is given preference.

Now we want to describe these tasks in graph theoretical terms.

Definition 11: Let $v \in V$. The cut of $G := (V,E)$ in v is the graph $G_{\{v\}} := (V_{\{v\}},E_{\{v\}})$, where

$V_{\{v\}} := \{p_i,p_o\} \cup V \setminus \{v\}$, $p_i,p_o \notin V$

$E_{\{v\}} := \{(x,y) \mid (x = p_i \wedge (v,y) \in E) \vee (y = p_o \wedge (x,v) \in E)\}$
$\cup\, E \setminus \{(x,y) \mid x = v \vee y = v\}$.

One easily verifies that for $v,w \in V$ the cuts are independent of their order, $G_{\{v\}\{w\}} = G_{\{w\}\{v\}}$. Thus we can define:

Definition 12: Let $W := \{w_1,\ldots,w_m\} \subset V$. The *cut* of $G(V,E)$ along W is the graph $G_W := (V_W,E_W) := G_{\{w_1\}\ldots\{w_m\}}$.

The subproblem to generate acyclic S-graphs can now be stated as follows:

Problem FBN(Feedback Node): Let $G = (V,E)$ be an S-graph. Find a set $W \subset V$ of minimal cardinality such that $G_W = (V_W,E_W)$ is acyclic.

FBN is known to be np-complete [Karp72], and heuristics are used in order to obtain good, suboptimal solutions. Let Z_G be the set of all elementary cycles of G. For each cycle $z \in Z_G$, we define $n(z) := \{v \in V \mid v \in z\}$, the set of all nodes of z. Now the scan selection problem is divided into two subproblems:

i) For the S-graph $G=(V,E)$, create the set of all elementary cycles Z_G, i. e. all cycles, where each node only appears once.

ii) Set $H := \bigcup_{z \in Z_G} n(z)$. Find a set $W \subset H$ of minimal cardinality, such that $\forall z \in Z_G\ W \cap n(z) \neq \emptyset$.

These are standard-problems of graph-theory, and there are well-known solutions. The implemented algorithms are based on methods described in [ChKo75], [John75], and additional heuristics are used. Alternatingly we select a bounded set Z'_G of elementary cycles, solve the hitting set problem ii), and select another bounded Z'_G.

The solution of subproblem b) is more complicated. First we explain how to segment purely combinational circuits, and then we extend this approach to general combinational representations.

Definition 13: Let $\bar{G} = (\bar{V},\bar{E})$ be a combinational representation, and let $v \in \bar{V}$. The natural number $d(v) := |I \cap p(v)|$ is called the *dependence level* of v.

Now we can state the segmentation problem of combinational circuits exactly:

Problem OCS (Optimal Circuit Segmentation): Let $G := (V,E)$ be a circuit graph of a combinational circuit, and $\ell \in \mathbb{N}$. Is there a set $W \subset V$ of size $k \leq |V|$ such that all vertices $v \in V_W$ in $G_W := (V_W,E_W)$ have dependence level at most ℓ, i. e. $d(v) \leq \ell$ in G_W?

For any cut along W, $d(v) \leq \ell$ in G_W can be checked in nearly linear time. Generating and checking all $2^{|V|}$ cuts would take ex-

ponential effort. Unfortunately, we cannot expect an algorithm of a better worst case complexity, since OCS is np-complete for $\ell > 2$. A complex proof of this theorem is found in [Bhat86], a shorter reduction in [HeWu89].

As OCS is np-complete, we refrain from looking for minimal solutions, but present some efficient heuristics. Dealing with general combinational representations is even more complex, since one physical cut of the circuit corresponds to multiple cuts in various time frames.

Definition 14: Let $\bar{G} = (\bar{V},\bar{E})$ be a combinational representation, and let $\bar{v} := (v,t) \in \bar{V}$. The *relevant time steps of* v are in set $T(v) := \{d | (v,d) \in \bar{V}\}$. The *equivalent node set of* v is $Q(v) := \{(w,d) \in \bar{V} \mid w=v\} \subset \bar{V}$.

Now we can formulate the general segmentation problem:

Problem OCRS: Let $G := (V,E)$ be an acyclic circuit graph, let $\bar{G} = (\bar{V},\bar{E})$ be its combinational representation, and let $\ell \in \mathbb{N}$. Is there a set $W \subset V$ of size $k \le |V|$, such that all vertices

$$v \in \bar{V}_{\bigcup_{w\in W} Q(w)} \text{ in } \bar{G}_{\bigcup_{w\in W} Q(w)} = (\bar{V}_{\bigcup_{w\in W} Q(w)}, \bar{E}_{\bigcup_{w\in W} Q(w)})$$

have dependence level at most ℓ, i.e. $d(v) \le \ell$ in $\bar{G}_{\bigcup_{w\in W} Q(w)}$?

We use some heuristics for an approximative solution of OCRS applying well-known methods, since OCRS is an instance of a general combinatorial optimizing problem:

CO (Combinatiorial Optimization): Let $\mathfrak{Z}$ be a set of states, $\mathfrak{Z}^* \subset \mathfrak{Z}$ be a set of admissible states, and let $k: \mathfrak{Z} \to \mathbb{R}$ be a cost function. Find an admissible state $Z \in \mathfrak{Z}^*$ with minimal costs $k(Z) = \min\{k(X) \mid X \in \mathfrak{Z}^*\}$.

For OCRS the set of states is $\mathfrak{Z} := \mathcal{P}(V)$, since every $Z \subset V$ determines a set of cuts with resulting graph

$$\bar{G}_{\bigcup_{z\in Z} Q(z)} = (\bar{V}_{\bigcup_{z\in Z} Q(z)}, \bar{E}_{\bigcup_{z\in Z} Q(z)}).$$

The admissible states are

$$\mathfrak{Z}^* := \{Z \in \mathfrak{Z} \mid \forall v \in \bar{V}_{\bigcup_{z\in Z} Q(z)} (d(v) \le \ell \text{ in } \bar{G}_{\bigcup_{z\in Z} Q(z)})\}.$$

The cost function $k: \mathfrak{Z} \to \mathbb{R}$, $k(Z) := |Z|$, corresponds to the necessary number of segmentation cells.

We define a heuristical function $h: \mathfrak{Z} \to \mathbb{R}$ to evaluate states:

$$h(Z) := \sum_{v\in V'} \ln(d(v)),$$

$$\text{where } V' := \{v \in \bar{V}_{\bigcup_{z\in Z} Q(z)} \mid d(v) > \ell \text{ in } \bar{G}_{\bigcup_{z\in Z} Q(z)}\}.$$

This function is an estimation of the number of vertices which have to be cut in the combinational representation. We assume an enumeration $\langle v_1,\ldots,v_n\rangle$ of $\bar{V}$ with $v_i \in pd(v_j) \Rightarrow i < j$.

Definition 15: Let $\bar{G} = (\bar{V},\bar{E})$ be a combinational representation, and let $v \in \bar{V}$. The *cone* $C(v)$ of v is the subgraph $C(v) := (p(v) \cup \{v\}, (p(v) \cup \{v\})^2 \cap \bar{E})$.

Definition 16: Let $\bar{G} = (\bar{V},\bar{E})$ be a combinational representation, and $\ell \in \mathbb{N}$. The *first violation* $fv \in \bar{V}$ is the node $fv = v_i$, where $i := \min\{j \mid d(v_j) > \ell\}$.

We construct a search graph $\mathcal{S} = (\mathfrak{Z}, \mathcal{E})$, where the nodes $Z \in \mathfrak{Z} = \mathcal{P}(V)$ define cuts in the sequential network, and an edge $(Z_1,Z_2) \in \mathcal{E}$ exists if and only if

a) $fv \in V \setminus Z_1$ is the first violation in $\bar{G}_{\bigcup_{z\in Z_1} Q(z)}$,

b) $Z_2 := Z_1 \cup \{v\}$, where $Q(v) \cap p(fv) \neq \emptyset$ and $h(Z_1 \cup \{v\})$ is minimal.

The search is started at $Z_0 = \emptyset$. One branches from Z to a $Z_1 \in \{\bar{Z} \mid (Z,\bar{Z}) \in \mathcal{E}\}$, preferably cutting lines corresponding to latches or flipflops in order to reduce the hardware overhead, until an admissible state Z is reached. The results of this process are presented in section 6.

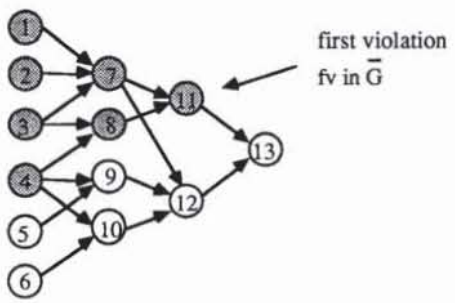

Figure 12: Example for $\ell = 3$.

For the example circuit of figure 2 and $\ell = 3$, the algorithm needs two steps to determine a solution for OCRS. This is illustrated in figure 13 showing the combinational representation of the circuit with an enumeration according to the signal flow. In the first step, node 14 is the first violation and the two equivalent nodes 12 and 13 are chosen to be cut. In the second step the first violation is node 22 and the algorithm decides to cut node 20. Nodes 12 and 13 correspond to node K2 in the original circuit and node 20 corresponds to K7. As both K2 and K7 are flip-flops, they simply have to be integrated into the partial scan path.

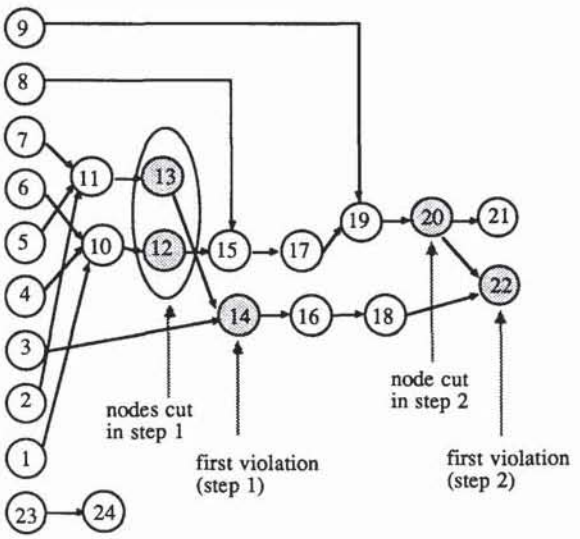

Figure 13: Steps of the segmentation algorithm for the example circuit of figure 2.

Since K12 has been cut before to guarantee an acyclic S-graph, there are altogether 3 of 5 flip-flops in the partial scan path. Figure 14 shows the resulting circuit.

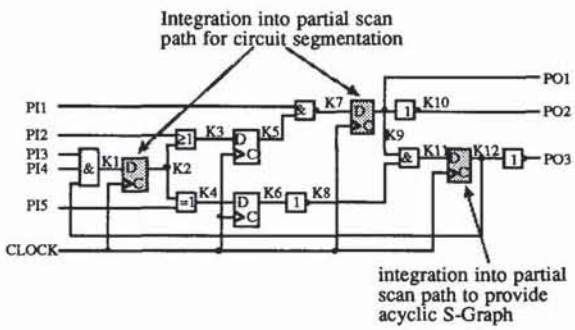

Figure 14: Resulting circuit after integration of a partial scan path and segmentation.

5. Pseudo-exhaustive test sequences

A considerable amount of work has already been done in investigating the generation of pseudo-exhaustive test sets for combinational circuits. Dependence-matrices [HiSi82], linear sums [Aker85], cyclic codes ([TaCh84], [WaMc86]) and special polynomials for linear feedback shift-registers [BARZ83] have been proposed. They are all applicable to combinational representations, and only little effort is needed to transform them into pseudo-exhaustive test sequences. Each pattern $p := \langle b_i \in \{0,1\} \mid i \in \bar{I} \rangle$ of the combinational representation corresponds to a pattern sequence. We remember $\bar{I} \subset I \times T$, hence the sequence is $S(p) := \langle\langle b_j^t \in \{0,1\} \mid j \in I \wedge t \in T(j)\rangle \mid t \in T\rangle$.

It should be noted, that not for the entire set $I \times T$ values are defined in S(p). This can be used for pattern compaction.

A further compaction is possible, if some sequences have common parts, e. g. the last patterns of sequence $S(p_1)$ are identical to the first of $S(p_2)$. These merged sequences can be generated with the help of cyclic codes by feedback shift registers, supporting a low-cost external test or a self-test. The details are beyond the scope of this paper, which aims at establishing a linear bound on the length of the pseudo-exhaustive test sequence.

Let $O \subset V$ be the set of primary and pseudo-primary outputs of the sequential network. Each output function of $o \in O$ is tested by at most 2^ℓ patterns in the combinational representation. Hence the size of the pseudo-exhaustive test set is bounded by $|O| \cdot 2^\ell$. Each pattern is mapped to a sequence of at most length $r := \text{rank}(G)$, hence the entire pseudo-exhaustive test sequence is bounded by $r \cdot |O| \cdot 2^\ell$.

6. Examples

We discuss three examples: the operation unit of the signal processor (SP) proposed in [Blan84], a multiplier presented in [Gutb88], and a PROLOG-coprocessor (PP) [Habe87].

Circuit	Inputs	Outputs	Gates	Flipflops
SP	83	55	1675	239
MU	43	26	993	183
PP	36	73	1428	136

Table 1: Circuit characteristics.

The unmodified circuits are very hard to test, which is proven with the help of the program LASAR [LASA85]. Fault coverages obtained after 3600 seconds of computing time are listed below.

PP	MU	SP
11.2 %	9.8 %	8.7 %

Table 2: Fault coverage by LASAR after 1h of computing time.

First, we have selected a small number of flipflops in order to obtain acyclic S-graphs (table 3).

PP	MU	SP
28 (20.6 %)	72 (39.3 %)	41 (17.2 %)

Table 3: Number and percentage of flipflops in a partial scan path in order to obtain an acyclic S-graph.

For these modified circuits, we have generated the combinational representation. The representations have been segmented by the algorithms described, where the maximal number of inputs to a cone has been varied from $\ell = 20$ to $\ell = 12$. The required test sizes are between some millions and a few thousands of patterns, which is competitive with a usual deterministic test.

In the table below we distinguish between cuts of flipflops resulting in additional scan path elements and other cuts requiring more expensive segmentation cells.

$\ell = 20$	PP	MU	SP
# segmentation cells	3	7	5
# additonal flipflops in the scan path	6	6	72
# overall flipflops in the scan path (percentage)	34 (25 %)	78 (43 %)	113 (47 %)

$\ell = 16$	PP	MU	SP
# segmentation cells	6	4	9
# additonal flipflops in the scan path	12	13	95
# overall flipflops in the scan path (percentage)	40 (29 %)	85 (46 %)	136 (57 %)

$\ell = 12$	PP	MU	SP
# segmentation cells	23	24	16
# additonal flipflops in the scan path	22	30	131
# overall flipflops in the scan path (percentage)	50 (37 %)	102 (56 %)	172 (72 %)

Table 4: Necessary number of segmentation cells and number of flipflops in the partial scan path, in order to make the circuits pseudo-exhaustively testable.

Table 4 shows significant savings of silicon area compared with the conventional complete scan design. The exact quantification depends on the layout of the used LSSD- and segmentation-cells. A rough estimation shows savings of approximately 50% for $\ell = 20$ and $\ell = 16$. But even for $\ell = 12$, the hardware overhead is competitive with a conventional scan design, since the larger number of segmentation cells is balanced by the shorter partial scan path.

In all cases the advantages are obvious:

- Complete fault coverage with respect to the usual fault models;
- No expensive test pattern generation;
- Simple test application.

Also with respect to the number of necessary segmentation cells the partial scan design is in most cases superior to the complete scan path. This is due to the fact that the integration of a complete scan path in general does not provide a pseudo-exhaustively testable circuit. Additional segmentation cells are necessary. Table 5 shows the number of segmentation cells required for an efficient pseudo-exhaustive test based on a complete and on a partial scan design.

$\ell = 20$	PP	MU	SP
partial scan path	3	7	5
complete scan path	9	4	4

$\ell = 16$	PP	MU	SP
partial scan path	6	4	9
complete scan path	14	7	10

$\ell = 12$	PP	MU	SP
partial scan path	23	24	16
complete scan path	40	11	56

Table 5: Number of necessary segmentation cells using complete and partial scan design, respectively.

In most cases the additonal number of segmentation cells increases, if all flipflops are integrated into a complete scan path.

7. Conclusions

The new concept of a pseudo-exhaustive test of sequential circuits has been introduced. Some flipflops and latches are integrated into an incomplete scan path, such that each possible state of the circuit is reachable within a few steps. Some more flipflops and some new segmentation cells are added to the partial scan path in order to make a pseudo-exhaustive test feasible. Algorithms have been presented for placing these devices automatically. Moreover it has been shown how to transform a pseudo-exhaustive test set into a pseudo-exhaustive test sequence of a similar size.

The analyzed examples show that a conventional complete scan path without additional testability features requires more hardware overhead than the presented test strategy which retains all the known benefits of a pseudo-exhaustive test.

Literature

Agra88 Agrawal, V.D.; Cheng, K.; Johnson, D.D.; Lin, T., Designing Circuits with Partial Scan, IEEE Design & Test of Computers, April 1988

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

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

BARZ83 Barzilai,Z.; Coppersmith,D.; Rosenberg,A.L., Exhaustive Generation of Bit Patterns with Applications to VLSI Self-Testing, IEEE Trans. on Comp., Vol. c-32, No. 2, Feb. 1983

Bhat86 Bhatt, S. N.; Chung, F. R. K.; Rosenberg, A. L., Partitioning Circuits for Improved Testability, Advanced Research in VLSI: proceedings of the 4-th MIT conference, April 7 - 9, 1986

Blan84 LeBlanc, J. J., LOCST: A Built-in Self-Test Technique, IEEE Design & Test, Vol. 1, No. 4, 1984

ChKo75 Christofides, N.; Korman, S., A Computational Survey of Methods for the Set Covering Problem; in: Management Science, Vol. 21, No. 5, Jan. 1975

EiWi77 Eichelberger, E.B.,Williams, T.W.: A Logic Design Structure for LSI Testability Proceedings 14th Design Automation Conference, June1977

Gutb88 Gutberlet, P., Entwurf eines schnellen Matrizen-Multiplizierers; Diploma Thesis at the Institute of Computer Design and Fault Tolerance, University of Karlsruhe, FRG, 1988

Habe87 Haberl, O., Entwurf und Implementierung eines PROLOG-Preprozessors als Standardzellen-Chip mit dem Entwurfssystem VENUS; Diploma Thesis at the Institute of Computer Design and Fault Tolerance, University of Karlsruhe, FRG, 1987

HeWu88 Hellebrand, S.; Wunderlich, H.-J., Automatischer Entwurf vollständig testbarer Schaltungen; Proceedings der 18. Jahrestagung der Gesellschaft für Informatik, Hamburg 1988, Informatik-Fachberichte 188, Springer-Verlag

HeWu89 Hellebrand, S.; Wunderlich, H.-J., Tools and Devices Supporting the Pseudo-Exhaustive Test submitted

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

John75 Johnson, D.B. Finding all the Elementary Circuits of a Directed Graph; SIAM J. Comput., Vol. 4, No. 1, March 1975

Karp72 Karp, R.M., Reducibility among Combinatorial Problems, in : R. E. Miller and J. W. Thatcher (eds.), Complexity of Computer Computations, Plenum Press, New York, 1972

Kunz89 Kunzmann, A., Test synchroner Schaltwerke auf der Basis partieller Prüfpfade; Ph. D. Thesis at the Institute of Computer Design and Fault Tolerance, University of Karlsruhe, FRG, 1989

LASA85 ETS LASAR Users Guide, Vers. 4.7, November 1985

McBo81 McCluskey, E.J.; Bozorgui-Nesbat, S., Design for Autonomous Test IEEE Trans. on Comp., Vol. C-30, No. 11, 1981

McCl84 McCluskey,E.J., Verification Testing - A Pseudoexhaustive Test Technique IEEE Trans. on Comp., Vol. C-33, No.6, June 1984

Roth78 Roth, J.P., Sequential Test Generation; Technical Disclosure Bulletin, IBM, Jan. 1978

TaCh84 Tang,D.T., Chen,C.-L., Logic Test Pattern Generation Using Linear Codes, IEEE Trans. on Comp., Vol. c-33, No. 9, Sep. 1984

Tris83 Trischler, E., Testability analysis and incomplete scan path, Proc. ICCAD 1983

Udel86 Udell, J.G., Jr., Test Set Generation for Pseudo-Exhaustive BIST, Proceedings International Conference on Computer Aided Design, 1986

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

Wu89 Wunderlich, H.-J., The Design of Random-Testable Sequential Circuits Proceedings 19th International Symposium on Fault Tolerant Computing, Chicago 1989

WuHe88 Wunderlich, H.-J.; Hellebrand, S., Generating Pattern Sequences for the Pseudo-Exhaustive Test of MOS-Circuits, Proceedings 18th International Symposium on Fault