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 \( C_0 \) 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.

An obvious advantage of this test strategy is the high fault coverage, within a cone all combinational 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], [Ude86], 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], [Kun89]). 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], [Kun89]). 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

1989 International Test Conference
Paper 2.2

CH2742-5/0000/0019S01.00 © 1989 IEEE
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 \( E \subseteq V \). \( V := V_C \cup V_S \cup I \) 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.

![Example circuit](image)

Figure 2: Example circuit.

The circuit graph \( G := (V,E) \) consists of the nodes \( V := \{P1, \ldots, P15, K1, \ldots, K8, K11, K12, P01, P02, P03\} \) 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
\( I = \{P11, P12, P13, P14, P15\} \).

![Circuit graph](image)

Figure 3: Circuit graph.

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

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

**Definition 3:** A circuit graph \( G := (V,E) \) is called consistent, if \( I = \{v \in V | p(v) = \emptyset\} \).

We only deal with consistent circuit graphs.

**Definition 4:** Let \( u, v \in V \). A path \( \omega \) 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 \( i = 1, \ldots, n \). \( \omega \) is called the length \( |\omega| \). If \( k_0 = k_n \) \( \omega \) is called a cycle.

**Definition 5:** Let \( G := (V,E) \) be a circuit graph, let \( v \in V \).
\( p(v) := \{w \in V | \) there is a path from \( w \) to \( v \}\} \) is the set of predecessors of \( v \), \( s(v) := \{w \in V | \) 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
\( V_{CI} := V_C \cup V_S \cup I \) and \( O_{CI} \). Its S-graph \( G_{S} := (V_S,E_S) \) is defined by:

a) \( V_S := O_{CI} \cup V_S \cup I \) and \( E_S := \{(v,w) | \) there is a path \( \omega \) from \( v \) to \( w \) in \( G_{CI} \) and \( \omega \cap V_S \neq \emptyset \} \)

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

The pseudo-exhaustive test of sequential circuits requires the application of pattern sequences instead of single patterns. Using Roth's notion 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. $rf(v) := \max \{I(o) | o$ is a path in $G$ with end point $v\}$ is called forward-rank of $v$, and $rb(v) := \max \{I(o) | o$ 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 $\text{rank}(G) := \max_{v \in V} \{rf(v)\} = \max_{v \in V} \{rb(v)\}$.

**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 $P: P(V_S \cup I) \rightarrow P(V_S \cup O)$; $P(W) := \bigcup_{w \in W} pd(w)$. The nodes of a subset $W \subseteq V$ have defined values at time step $t$, if the nodes $W^{-1} := P(W)$ 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 $P(V_S \cup I) \subseteq I$.  

**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 $G^O := (V^O,E^O)$ be its circuit graph. Set $W_t := \{v \in V^O | \exists u \in V^S \exists \tau \subseteq \{0,1\} \text{ where } (v,u) \in E^O \}$ and for $0 \leq t < r$:

$W_t := P(W_{t+1})$.

$V_t := \{v \in V^O | \exists \omega \in W_t \exists \tau \subseteq \{0,1\} \text{ where } (v,u) \in E^O \}$ and

$W_t := P(W_{t+1})$.

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

**Theorem 1:** Let $G := (V,E)$ be an acyclic circuit graph, with rank $r$, and let $G := (V,E)$ be its combinational representation. A pattern sequence $\langle \rho \rangle := \{0,1\} \text{ where } (v,t) \in 0stfr >$ detects a given fault at a node $v \in V$ exactly at time $t$, if and only if in $G$ the corresponding multiple fault of the nodes $(v,t), 0stfr$ if defined, is detected by the pattern $\langle \rho \rangle := \{0,1\} \text{ where } (v,t) \in 0stfr >$. 

**Figure 4:** S-graph.

**Figure 5:** Acyclic circuit graph.

The combinational representation of $G^O$ is the graph $G := (V,E)$, where

$V := \bigcup_{0 \leq t < r} V_t \times \{t\}$

$E := \bigcup_{0 \leq t < r} \{((x,t),(y,t)) | (x,y) \in V_t \times V_t \cap E \} \bigcup \{((x,t),(y,t+1)) | x \in V_t \wedge y \in V_{t+1} \wedge (x,y) \in E \} \bigcup \{((x,t),(y,t+1)) | x \in V_{t+1} \wedge (x,y) \in E \} \bigcup \{((x,t),(y,t+1)) | x \in V_{t+1} \wedge (x,y) \in E \}$

$V_C := \bigcup_{0 \leq t < r} \{V_t \times V_t \cap E \}$

$V_C := \bigcup_{0 \leq t < r} \{V_t \times V_t \cap E \}$

$O := \{(o,r) | o \in O \}$

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:
If a cone \( C_0 \) of the combinational representation has \( \ell \) primary inputs, it is tested exhaustively by \( 2^\ell \) 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 irredudant, 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 [EIW77]. 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 6: Combinational representation.

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 8: Cut of a node \( v \).

In [Bha86] 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 \( L_1(L_2) \)-latch with data-input \( D \), clock \( CLK \), shift input \( SDI \) and shift-clock \( A(B) \).
In the next section, we discuss how to place the pseudo-exhaustive latches and segmentation cells.

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 \( \overline{G} = (\overline{V}, \overline{E}) \) be a combinational representation, and let \( v \in \overline{V} \). The natural number \( d(v) := l \cap p(v) \) is called the \textit{dependence level} of \( v \).

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

\textbf{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 \( W \subseteq V \) of size \( k \leq \ell \) such that all vertices \( v \in W \) in \( G_W := (V_W, E_W) \) have dependence level at most \( \ell \), 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-
Problems OCRS: Let \( G = (V, E) \) be an acyclic graph, and let \( \mathcal{S} = (\mathcal{S}, \mathcal{E}) \) be a combination of admissible representations, and let \( \mathcal{S} \in \mathcal{S} \). Is there a set \( W \subseteq V \) of size \( k \leq |V| \), such that all vertices

\[ v \in \mathcal{G} \cup \mathcal{Q}(v) \]

have dependence level at most \( \ell \), i.e., \( d(v) \leq \ell \) in \( \mathcal{G} \cup \mathcal{Q}(v) \)?

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

**CO (Combinatorial Optimization):** Let \( \mathcal{S} \) be a set of states, \( \mathcal{S} \subseteq \mathcal{S} \) be a set of admissible states, and let \( k: \mathcal{S} \rightarrow \mathbb{R} \) be a cost function. Find an admissible state \( Z \in \mathcal{S} \) with minimal costs

\[ k(Z) = \min \{ k(X) \mid X \in \mathcal{S} \} \]

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

\[ \mathcal{G}(\mathcal{S}) = (V \cup \mathcal{Q}(v)) \mathcal{E} \cup \mathcal{Q}(v) \]

The admissible states are

\[ \mathcal{S}^* = \{ Z \subseteq V \mathcal{S} \mid \forall v \in Z \mathcal{G}(\mathcal{S}) (d(v) \leq \ell \text{ in } \mathcal{G}(\mathcal{S})) \} \]

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

We define a heuristic function \( h: \mathcal{S} \rightarrow \mathbb{R} \) to evaluate states:

\[ h(Z) = \sum_{v \in V} \ln(d(v)) \]

where \( V^* = \{ v \in V \mathcal{S} \mid d(v) > \ell \text{ in } \mathcal{G}(\mathcal{S}) \} \).

This function is an estimation of the number of vertices which have to be cut in the combinational representation. We assume an enumeration of vertices \( v_1, \ldots, v_n \) of \( V \) with \( d(v_i) \Rightarrow i < j \).

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

**Definition 16:** Let \( \mathcal{G} = (V, E) \) be a combinational representation, and let \( \mathcal{S} \subseteq \mathcal{S} \). The first violation \( v \) in \( \mathcal{G} \) is the node \( v \) where

a) \( v \in V \mathcal{S} \) is the first violation in \( \bigcup_{Z \in Z} \mathcal{Q}(v) \)

b) \( Z_1 \cup \{ v \}, \) where \( \mathcal{Q}(v) \cap \mathcal{Q}(v) \neq \emptyset \) and \( h(Z_1 \cup \{ v \}) \) is minimal.

The search is started at \( Z_0 = \emptyset \). One branches from \( Z \) to \( Z_1 \in \mathcal{S} \), 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 \( \ell = 3 \).
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.

6. Examples

We discuss three examples: the operation unit of the signal processor (SP) proposed in [Blan84], a multiplier presented in [Guth88], and a PROLOG-coprocessor (PP) [Habe87].

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Inputs</th>
<th>Outputs</th>
<th>Gates</th>
<th>Flipflops</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP</td>
<td>83</td>
<td>55</td>
<td>1675</td>
<td>239</td>
</tr>
<tr>
<td>MU</td>
<td>43</td>
<td>26</td>
<td>993</td>
<td>183</td>
</tr>
<tr>
<td>PP</td>
<td>36</td>
<td>73</td>
<td>1428</td>
<td>136</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th></th>
<th>PP</th>
<th>MU</th>
<th>SP</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.2 %</td>
<td>9.8 %</td>
<td>8.7 %</td>
<td></td>
</tr>
</tbody>
</table>

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).
12. The for 2. depends on cone hu algoriihms de$cribed. where the maximaJ cells.

Table

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.

$$
\begin{array}{|c|c|c|c|}
\hline
\ell & \text{PP} & \text{MU} & \text{SP} \\
\hline
20 & 28 (20.6\%) & 72 (39.3\%) & 41 (17.2\%) \\
\hline

depends on cell haw: generated representation . The representations have been scgmenlC:d resulting in additional scan pa th siales are more eIL:pensive SCiffiCntation pattern s.

$$
\begin{array}{|c|c|c|c|}
\hline
\ell & \text{segmentation cells} & \text{additional flipflops} & \text{overall flipflops in the scan path (percentage)} \\
\hline
20 & 3 & 6 & 34 (25\%) \\
\hline
16 & 6 & 12 & 40 (29\%) \\
\hline
12 & 23 & 22 & 50 (37\%) \\
\hline
\end{array}
$$

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.

$$
\begin{array}{|c|c|c|c|}
\hline
\ell & \text{PP} & \text{MU} & \text{SP} \\
\hline
20 & 3 & 7 & 5 \\
\hline
16 & 6 & 4 & 9 \\
\hline
12 & 23 & 24 & 16 \\
\hline
\end{array}
$$

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

In most cases the additional 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,

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,

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

Bian84 LeBlanc, J. J.,
LOCST: A Built-in Self-Test Technique,
IEEE Design & Test, Vol. 1, No. 4, 1984

ChKo75 Christofides, N.; Korman, S.,

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

HeWu89 Helebrand, S.; Wunderlich, H.-J.,
Automatischer Entwurf vollständig testbarer Schaltungen;

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,

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.; Bozorgi-Nesbat, S.,
Design for Autonomous Test

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., Shen, 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

Ude186 Udell, J.G., Jr.,
Test Set Generation for Pseudo-Exhaustive BIST,

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

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