Proceedings First European Design Automation Conference (EDAC), Glasgow, 1990

Tools and Devices Supporting the Pseudo-Exhaustive Test

Sybille Hellebrand, Hans-Joachim Wunderlich

Institute of Computer Design and Fault-Tolerance
(Prof. Dr. D. Schmid)
University of Karlsruhe
P. O. Box 6980
D-7500 Karlsruhe
Federal Republic of Germany

Abstract:

In this paper logical cells and algorithms are presented supporting the design of pseudo-exhaustively testable circuits. The approach is based on real hardware segmentation, instead of path-sensitizing. The developed cells segment the entire circuits into exhaustively testable parts, and the presented algorithms place these cells, under the objective to minimize the hardware overhead.

The approach is completely compatible with the usual LSSD-rules. The analysis of the well-known benchmark circuits shows only little additional hardware costs.

Keywords: Pseudo-exhaustive test, automatic design for testability.

1. Introduction

The pseudo-exhaustive test, sometimes called verification test, retains almost all benefits of an exhaustive test, whereas the complexity is reduced in many cases [McBo81].

For a primary output o of a combinational circuit, the cone C_o is the minimal subcircuit containing all nodes, which are on a path to o (fig. 1).

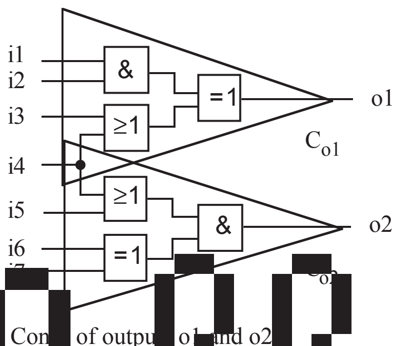

Figure 1: Cones of outputs o1 and o2

A cone is tested by enumerating all patterns at its primary inputs. The sum of all these patterns is smaller than an exhaustive test, if the cones are sufficiently small. The total size t of a pseudo-exhaustive test set is estimated by the inequation

$$2 \le t \le m \cdot 2 , \qquad (1)$$

where is the maximal number of inputs of the cones, and m is the number of primary outputs. This test strategy is also applicable to sequential circuits, if a scan path is integrated. An obvious advantage 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].

The high fault coverage is obtained without an expensive test pattern generation, optionally the number of necessary patterns can be reduced by some compaction techniques [McCl84a], [Aker85]. Unfortunately a pseudo-exhaustive test is not applicable, if a primary output depends on a very large number of primary inputs. This problem is solved by circuit segmentation, where essentially two techniques are proposed: Sensitizing techniques assign fixed values to some primary inputs, in order to sensitize paths to some regions of the circuit which are exhaustively tested [Udel86], [McBo81]. Hardware segmentation cuts some circuit lines logically during a test mode by some additional circuitry [McBo81].

In the presented approach, the hardware segmentation is supported. A uniform technique is presented integrating the scan design and the segmentation of the combinational network, in a similar way as proposed in [Bhat86]. The authors of this paper propose to integrate LSSD-latches into the combinational network, such that the circuit function is not altered. But the system operation of the circuit is slowed down, and they have to integrate additional latches only for timing reasons. This leads to a considerable hardware-overhead.

In the presented approach, we use special segmentation cells, which are compatible with the usual LSSD design styles, as e.g. the L1/L2*-technique, and which do not alter the system mode of the circuit in any way. Benefits are a less complex partitioning problem, lower hardware-overhead, and minimal drawbacks concerning the circuit speed. The cells are placed into the combinational network and can be used as pseudo-primary inputs or outputs. In order to minimize the hardware overhead, several cell placement algorithms based on hill-climbing and simulated annealing techniques have been implemented.

In the second section of the paper, we discuss the classical hardware segmentation techniques, sketch some rules of the LSSD-technique and state some requirements the segmentation cells have to meet. The new cells are presented in more detail in section 3. Section 4 gives a graph-theoretic formulation of the cell placement problem. Since this problem is np-complete, we use some heuristics to obtain suboptimal solutions by hill-climbing algorithms. The algorithms and results are discussed in section 5. In order to validate the quality of the obtained results a simulated annealing procedure has been implemented additionally.

2. Classical Segmentation Techniques

As the pseudo-exhaustive test in the classical sense is only applicable to combinational networks, a scan design is mandatory. The classical LSSD-approach puts some restrictions on the network, which are sketched in this section [EiWi77]. Finally, we investigate some drawbacks of the classical multiplexer partitioning technique to these LSSD-requirements.

2.1. LSSD-techniques

A detailed description of LSSD-techniques is found in survey papers [McCl84b] or textbooks [Fuji85], [McCl86]. Therefore we only discuss the principles, which are of importance to the segmentation cells proposed in section 3. For the sake of simplicity, we only describe the 2-phase clock technique. But the presented approach is compatible with a general LSSD-architecture. The basic storage device is a hazard-free polarity-hold latch, which may be implemented by transmission gates as shown in figure 2.

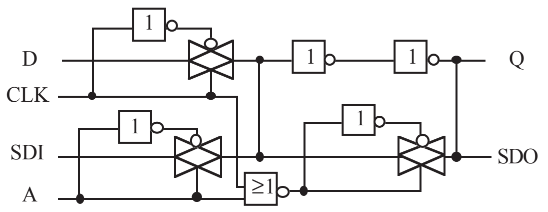

Figure 2: Implementation of a polarity-hold latch.

This latch has the data input D clocked by system-clock CLK, and the shift data input SDI clocked by A.

Design rules have to guarantee a stable behavior of the circuit function. These rules are fulfilled if for example the circuit is organized in a so-called L1/L2*-configuration. Two non-overlapping system-clocks CLK1 and CLK2 control two disjoint sets of latches. The latches controlled by CLK1 are called L1-, and the latches controlled by CLK2 are called L2*-latches. The combinational network is partitioned into two disjoint sets N1 and N2, such that the L1-latches receive their data from the N1-network, but they do not give any data to N1. Analogously, the L2*-latches receive their data from N2, but do not give data to N2. Figure 3 shows the structure of a circuit designed in L1/L2*-technique.

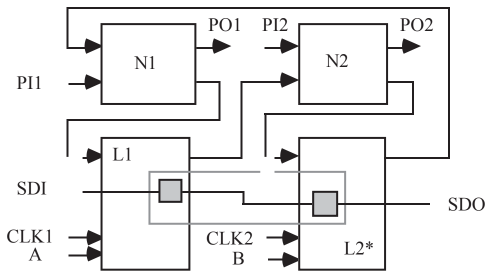

Figure 3: L1/L2*-technique.

During test mode the latches are combined to shift register latches (SRLs) as shown in figure 4. All SRLs are interconnected into a scan path, which is controlled by the two non-overlapping shift-clocks A and B.

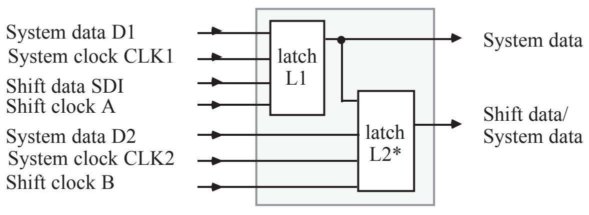

Figure 4: Shift register latch within an L1/L2*-configuration.

There are programs reported for checking a partitioning of the circuit according to these rules, and supporting the partitioning [GoFB77]. The rules are easily extended according to schemes of multiple clocks, and any additional "design for testability"-features have to be compatible with these LSSD-rules.

2.2. Multiplexer partitioning

One of the first papers describing the design of pseudo-exhaustively testable circuits proposed multiplexer partitioning [McBo81]. Figure 5 shows a partitioning into two parts by multiplexers.

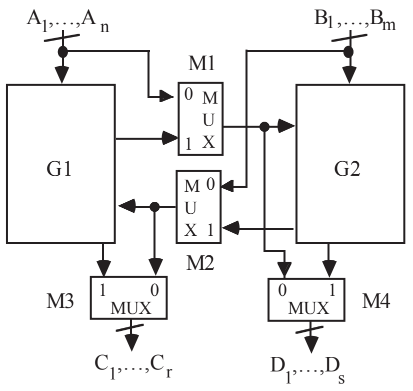

Figure 5: Multiplexer partitioning.

During system operation, the 1-inputs of the multiplexers are sensitized, and the subcircuits G1 and G2 are connected. During the test of G2, at M1 and M3 the 0-inputs are sensitized, and at M2 and M4 the 1-inputs are activated. The inverse controls test subcircuit G1. This original technique has some disadvantages:

1) There is a large overhead due to wiring, since there are multiple control lines for the multiplexers, and additional lines are necessary to control and observe the node itself.
2) For every cut node, there are two multiplexers within the data path.
3) If several segments have to be tested, a complex control of the multiplexers is required. Moreover the segments can not be tested in parallel, and the multiplexers are not tested exhaustively.
4) Some layout-dependent faults are not tested. These faults can only be detected during the system mode.
5) The inputs A, B, and the outputs C, D are latches in the LSSD design, where new dependences are introduced. This may destroy a given partitioning according to the LSSD rules.

3. Segmentation cells

The concept of segmentation cells integrates network partitioning and scan design. The cells are placed into the combinational network, where during system operation they are a simple connecting line, but during the test mode they are a latch within the scan path (fig. 6).

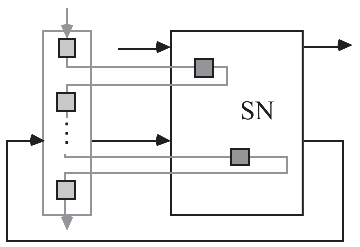

Figure 6: Integrating segmentation and scan design.

The additional test circuitry should fulfill the following requirements:

a) All faults within the segmentation cells are detectable during the pseudo-exhaustive test.

b) The additional silicon area is minimized.

c) There are no serious impacts on the system behavior of the circuit.

These requirements are met by the segmentation cell of fig. 7 to a wide extent. Essentially the cell is a L1-latch, and two of them form a L1/L2*-configuration. They may be part of a usual scan path.

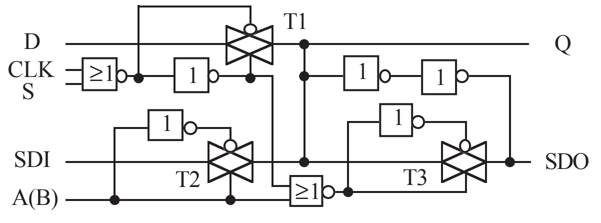

Figure 7: Segmentation cell.

Signal S determines system and test modes. For S = 1, data-input D and output Q are directly connected. During the test mode, if S = 0, the cell works like a usual latch within the scan path. The cell satisfies the segmentation function like the multiplexer solution, but avoids the mentioned disadvantages:

1) ***Wiring:*** The segmentation cells are members of the scan path, and they can be wired by a suitable program, automatically after the design, regardless of their order. This very high degree of freedom minimizes the overhead (see [AgJS84]).
2) ***Delay:*** Only the transmission gate T1 is member of the data path, in contrast to the multiplexer approach.
3) ***Control:*** For segmentation, only S := 0 is set, and all cells are controlled by the same signal.
4) ***Fault coverage:*** Every part of the data path D-Q is used during the test mode, and a pseudo-exhaustive test is an exhaustive test of most parts of the segmentation cell. The system signal S is easily tested, since CLK and S are symmetric and directly accessible.
5) ***LSSD-compatibility:*** The placement of the segmentation cells does not introduce any new dependences of state variables within the original circuit. For this reason a LSSD-partitioning of a circuit remains valid after segmentation. But additionally the segmentation cells have to be partitioned according to the system clocks, too. In few cases, this may require a double-latch configuration of the se[redacted]ntation cells.

4. A gra[redacted]theoretic formulation of t[redacted]e se[redacted]entation pro[redacted]em

The ad[redacted]iona[redacted]costs of a pseudo-exhaust[redacted]test are directly measur[redacted] by[redacted]e required number of seg[redacted]ntation cells. The number[redacted]d be minimal, but still ensu[redacted]a short test length. This is [redacted]pressed by the following problem:

Let C b[redacted] combi[redacted]ational circuit, and let ∈ I[redacted]ace a minimal set of s[redacted]men[redacted]ion cells, such that every node depends on at most pri[redacted]y or pseudo-primary inputs.

This problem can regarded as a graph-theoretic problem. In the following a combinational circuit is represented by an acyclic directed graph G = (V,E) with vertices V and edges $E \subset V^2$. The vertices of G correspond to the nodes of the circuit, and there is an edge (v,w) ∈ E, if and only if node v is input of a gate with output w. If a segmentation cell is built in a circuit node, this can be modeled as a "cut" of the corresponding vertex in G. Figure 8 shows a circuit with segmentation cells and the corresponding circuit graph.

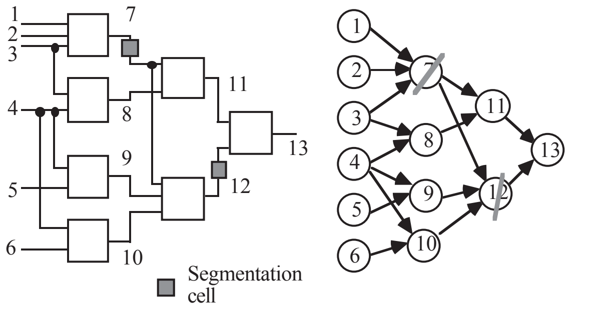

Figure 8: Circuit with segmentation cells and its representation as directed graph.

The cut of a vertex v∈V transforms G into a graph $G_{\{v\}}$; v is replaced by two new vertices v_o and v_i, such that v_o represents an additional output and v_i an additional input of the circuit. As illustrated in figure 9, v_o has no successors, and its predecessors are the predecessors of v, whereas v_i has no predecessors and its successors are the successors of v.

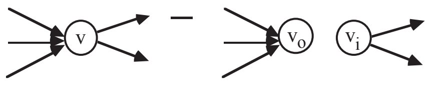

Figure 9: Cut of a vertex v∈V.

One easily verifies that for v,w ∈ V the cuts are ind[redacted]dent of their order. Thus for a set W := $\{w_1,\ldots,w_m\}\subset V$ [redacted]e def[redacted]e the cut of G along W as $G_W := (V_W,E_W) := G_{\{w$[redacted]

For a vertex v∈V the natural numb[redacted](v) [redacted]notes[redacted]u[redacted]ber o[redacted] primary inputs which are conne[redacted]ted t[redacted] v [redacted] a p[redacted] in [redacted] Wit[redacted] this we can state our segmentatio[redacted]probl[redacted]m [redacted]ac[redacted]

***Problem OCS** (**O**ptimal **C**ircu*[redacted]*Se*[redacted]*ent*[redacted]*ion):*

Let G := (V,E) be an acyclic dire[redacted] gra[redacted] ∈ I[redacted]here a set W⊂V of size k ≤ |V| such[redacted]t all ve[redacted]es [redacted]$_W$ in G_W := [redacted] (V_W,E_W) have dependence level[redacted] most [redacted]) ≤ in [redacted]

For any cut along W, d(v) ≤ in [redacted]ca[redacted] checked in nearly linear time. Generating and checking al[redacted]|V| cuts would take exponential effort. Unfortunately we cann[redacted]xpect an algorithm of a better worst case complexity:

Theorem 1: OCS is np-complete for > [redacted]

A complex proof of this theorem is found in [Bhat86], a shorter reduction in [Hell90, Wu90].

Since OCS is np-complete, we refrain from looking for minimal solutions, but present some efficient heuristics in the next section.

5. Hill-Climbing procedures for OCS

In this sect[redacted]present an[redacted]opt[redacted]al s[redacted]lution of OCS. [redacted]nce it ha[redacted]n ex[redacted]ors[redacted]case[redacted]mp[redacted]ity [redacted]the algorith[redacted]s [redacted]d[redacted] [redacted]iving suboptimal[redacted]lu[redacted]ons wit[redacted]her[redacted]cy. Th[redacted]resu[redacted]s are compared with an a[redacted] propo[redacted] L[redacted]ha[redacted]ng a similar objective. O[redacted]S is[redacted]nsta[redacted] a [redacted] a[redacted]nbi[redacted]torial optimizing problem:

***CO** (**C**[redacted]nbinati[redacted]al **O**ptim[redacted]*

Let b[redacted] states, and let k: [redacted]e a[redacted]on. Find an admissible state

$Z \in$ [illegible]* [illegible]ini[illegible](Z) = [illegible] $X \in$ [illegible]*}

For OCS the se[illegible] states is [illegible] :[illegible] determines a cut. [illegible]e ad[illegible]ssible states are [illegible]* [illegible] $\forall v \in V_Z : d(v) \leq$ [illegible] in [illegible]. The cost function k: [illegible] |Z|, corresponds to the necessary number of segmenta[illegible]n cells. In addition to that we d[illegible]e a heuristic function h: [illegible]o evaluate states:

$$h(Z) := \sum_{v \in V'} \ln(d(v[illegible] w[illegible] = \{v \in V_Z \mid d(v) > [illegible] \text{ in } [illegible]\}.$$

This fu[illegible] an e[illegible]ion [illegible] the number of vertices which have [illegible] be cut [illegible] a[illegible]n[illegible]. We assume an enumeration $<v_1,\ldots v$[illegible] of V w[illegible] $i <$[illegible]

A global o[illegible]al [illegible]ion [illegible]CS can [illegible] obtained by th[illegible] procedure glob([illegible]) f[illegible]

Pr[illegible](G, [illegible]:

[illegible]:[illegible] $i_j < i_j$[illegible]} is the se[illegible] of all ordered tupels instead of sets. [illegible] is [illegible]iz[illegible] T_G, where the root Z_0 is the empty tupel, and the direct successors of $Z \in$ [illegible] a[illegible] $sd(Z) := \{<v_{i_1},\ldots,v_{i_k},v_{i_{k+1}}> \mid Z = <v_{i_1},\ldots,v_{i_k}>, i_k < i_{k+1} \leq n\}$.

We have to look for an admissible state having shortest distance to the root. Of course $W := <v_1,\ldots,v_n>$ is admissible, and we set **current** := n. We carry out a depth-first search within this tree, but never go deeper than **current**. If we find a new admissible state at depth k<n, we update **current** := k.

end-glob.

The procedure glob provides all admissible states of minimal costs, but it has an exponential worst case complexity. More efficiently problems of CO can be dealt with by hill-climbing procedures [Rich83]. A hill-climbing procedure for OCS ba[illegible]d on a "divide-and-co[illegible]-principle can be built using the fo[illegible]wing concepts.

Definition 1: L[illegible] G := [illegible],E) be a circuit graph, $v \in V$ an[illegible]t p(v) deno[illegible] a[illegible]predecessor[illegible] C([illegible] is the s[illegible]graph [illegible] (p(v) ∪ [illegible]},[illegible]

Defi[illegible]ion 2 [illegible] G [illegible](V,E) be[illegible] cir[illegible] [illegible]. The *first* [illegible] fv[illegible] is the [illegible] f[illegible] := min{ j | d(v_j) >[illegible]}.

We [illegible]struc[illegible] search graph [illegible]=([illegible] = [illegible](V[illegible]ne cuts, and an edge $(Z_1,Z_2) \in$ [illegible] exist[illegible] if

a) $fv \in V \backslash Z_1$ is the first viola[illegible]on in G[illegible], and

b) $W \subset p(fv)$ is an optimal [illegible]lutio[illegible] for OCS of the subgraph C(fv) in G_{Z_1} and $Z_2 :=$ [illegible]$\cup W$

The search is started at Z[illegible] = Ø, and [illegible]til an admissible st[illegible]e Z is reached, one branches fro[illegible] Z to a Z_1 w[illegible]h $(Z,Z_1) \in$ [illegible] and [illegible] $= \min\{ h(\tilde{Z}) \mid (Z,\tilde{Z}) \in$ [illegible]}.

Searching $W \subset p(fv)$ in b) is done by the procedure glob. Let pd(v) be the set of direct successors of a vertex $v \in V$. Then W := pd(fv) and **current** := |pd(fv)| form a (sub-) optimal solution. Therefore the search space can be reduced drastically.

A further reduction of the search space is possible if not all optimal solutions for OCS in C(fv) are taken into account.

[illegible] 3: Let G := (V[illegible]) [illegible] an acyclic directed graph, [illegible]d let d(v) denote th[illegible] set of direct successors for $v \in$ [illegible]. Moreo[illegible] let $fv \in V$ be th[illegible]st violation in G, and $v \in p(fv)$.

X := $\{x \in p(fv) \mid |sd(x)| > 1$[illegible]

Y[illegible] $\{y \in p(fv) \mid (d(y) >$ [illegible] $- |p$[illegible]$| + 1) \wedge (|pd(y)| > 1)\}$.

[illegible]e is a uniquely determined path $(v = \omega_0, \ldots ,\omega_k = :v^*)$ of minimal length, such that

$$\forall\, i<k: (|sd(\omega_i)| = 1) \wedge (v^* \in X \cup pd(fv) \cup \bigcup_{y \in Y} pd(y)).$$

v* is called the *push-forward* of v.

Theorem 2: Let $fv \in V$ be the first violation in G and W := [illegible]$w_1,\ldots,w$[illegible] $\subset$ pf(fv) an optimal solution of OCS in C(fv). T[illegible]n th[illegible]pus[illegible]-forward W* := $\{w_1^*,\ldots,w_m^*\}$ of W is an optimal [illegible], too.

Pro[illegible]: Left to the reader.

T[illegible] basic hill-climbing procedure is augmented by backtracking and by an additional rule. If a certain number of subsequent state transitions did not reduce the heuristic h, one re[illegible] earlier state Z and selects another successor of Z. If this back-tracking also fails in an improvement, one jumps to another state U according to the following rule:

Let $Z \subset V$ define a cut $G_Z := (V_Z,E_Z)$, let $fv \in V_Z$ be the first violation in G_Z. Let $v \in pd(fv)$ be selected such that $d(v) = \max_{w \in pd(fv)} d(w)$. Jump to the state $U := Z \cup \{v\}$.

This defines the complete[illegible]ill-c[illegible]bing proce[illegible]re, h[illegible]ing a worst-case complexity of O[illegible]pd(fv[illegible] $\cdot O(|V|^2)$. Th[illegible] segm[illegible]tation algorithm named DC (divide[illegible]nd[illegible]onquer) has bee[illegible]im[illegible]emented in PASCAL under the UN[illegible] operation system. [illegible] reasons of comparison, an algorithm [illegible]MP based on the p[illegible]er [RoLa84] has been implemented, t[illegible] Table 1 shows the[illegible]ired number of segmentation cells for th[illegible]well-kn[illegible]n benchma[illegible] circui[illegible]

	=1[illegible]		= 2[illegible]	
circuit	DC	CMP	DC	CMP
[illegible]	27	56	20	44
c[illegible]99	8	20	9	32
[illegible]880	19	34	14	48
c135[illegible]	8	20	9	32
c[illegible]8	21	51	18	53
c2670	45	108	37	77
c3540	87	138	63	105
c[illegible]5	52	96	42	82
c6288	93	115	65	74
c7552	100	118	79	75

Table 1[illegible] Nu[illegible]be[illegible] se[illegible]n[illegible]n [illegible]lls by DC [illegible]d CMP

Alth[illegible]gh the di[illegible] an[illegible] algorithm do[illegible] no[illegible] completely negle[illegible] the glo[illegible] effect[illegible] o[illegible] a cut, the main e[illegible]phasis is put on finding [illegible] o[illegible]mal so[illegible]ions. Therefore it cannot be excluded that [illegible] procee[illegible]g only [illegible] on a good [illegible]obal heuri[illegible]ic might co[illegible] t[illegible] bett[illegible] results. I[illegible] order to i[illegible]estiga[illegible] this que[illegible] and to av[illegible]d the [illegible]mp[illegible]xity of D[illegible], th[illegible] search tr[illegible] is modif[illegible]d into [illegible]' := [illegible] $(Z_1,Z_2) \in$ [illegible]' exis[illegible]only if

a) $fv \in V \backslash Z_1$ is the first violation in G_{Z_1}, and

b) $Z_2 := Z_1 \cup \{v^*\}$, where $v \in p(fv)$ is a node of V_{Z_1} with minimal $h(Z_1 \cup \{v\})$.

This proceeding uses the fact, that surely one of the predecessors of the first violation must be cut, and this may be a pushed-forward vertex[illegible] Thi[illegible] restricts the search [illegible]pac[illegible] and a vertex is cut minimizi[illegible] the global heuristic. T[illegible]lgorithm called GL has a worst [illegible]se complexity of O(|pd(f[illegible]|) · O(|V|^2). Table 2 shows for GL[illegible]d for DC the necessary [illegible]ber of cells and the computing time [illegible]easure[illegible] on a SUN 3/280 [illegible]mputer[illegible]

	=1[illegible]		= 2[illegible]	
circuit	DC	GL	DC	GL
c432	27 *(4013)*	27 *(25)*	20 *(619)*	21 *(30)*
c499	8 *(104)*	8 *(10)*	9 *(6)*	9 *(12)*
c880	19 *(8)*	16 *(14)*	14 *(8)*	14 *(18)*
c1355	8 *(7455)*	8 *(85)*	9 *(7423)*	9 *(99)*
c1908	21 *(6326)*	22 *(140)*	18 *(6083)*	17 *(349)*
c2670	45 *(55267)*	33 *(123)*	37 *(3972)*	29 *(121)*
c3540	87 *(22438)*	90 *(858)*	63 *(3661)*	68 *(877)*
c5315	52 *(19637)*	62 *(411)*	42 *(27814)*	46 *(350)*
c6288	93 *(15390)*	98 *(4647)*	65 *(40916)*	70 *(6692)*
c7552	100 *(74316)*	117 *(1667)*	79 *(29379)*	85 *(1567)*

Table 2: Number of segm[illegible]on cells and computing time (in brackets) in se[illegible] for th[illegible] hill-climbing procedures DC and GL.

An analysis of table 2 sho[illegible]s [illegible]at only for the circuits c7552 and c5315 the faster global he[illegible]ic results in a distinct deterioration. For the c2670-circuit eve[illegible] far better results are obtained.

Table 3 gives an idea of t[illegible] hardw[illegible]e-overhead required for segmentation. Column 1 de[illegible]tes th[illegible] necessary minimal number of segmentation cells for = [illegible]olumn 2 gives its percentage in terms of primary inputs, and the last column is the percentage of nodes to be cut.

circuit	number of cells	percentage of primary inputs	percentage of of nodes
c432	20	55.5	10.2
c499	9	22.0	3.7
c880	14	23.[illegible]	3.2
c1355	9	22.[illegible]	1.5
c1908	17	51.[illegible]	1.9
c2670	29	[illegible].4	2.0
c3540	63	126.0	3.7
c5315	[illegible]2	23.6	1.7
c6288	[illegible]5	203.0	2.7
c7552	[illegible]9	38.2	2.1

Table 3: Hardwa[illegible]verhead for = [illegible]

A common pro[illegible]n of hill-climbing procedures is the occurence of local minima [illegible]r apart [illegible]om the global optimum. Such local, suboptimal solut[illegible]ns ha[illegible] been reached by GL at circuits c6288 and c7552 for = 2[illegible]herefore a simulated annealing algorithm has been implemented in order to validate the obtained solutions. It tries to make use of the specific problem structure as far as possible while preserving the necessary conditions for convergence. A detailed description is found in [HeWu88].

Applying simulated annealing to the benchmark circuits we obtained the following results. Starting with an initial state of relatively high costs the algorithm came to results close to those of DC, but in no case there was an improvement of the best results received by DC or GL respectively. For the c880-circuit for example simulated annealing was started with the result produced by CMP with costs 34. The final state of the annealing procedure had costs 22, whereas DC resulted in costs 19. A detailed analysis of the annealing process for this circuit showed that for small values of c still about 1% of the generated transitions where accepted. But although transitions increasing the cost function were no longer accepted, a further improvement of the result could not be achieved.

The behavior of the annealing algorithm validates the fact, that the results obtained by the hill-climbing algorithm are very close to a global optimal solution. On the other hand experiments with simulated annealing have shown, that good tailored heuristics often are superior to simulated annealing methods [NaSS86]. This statement seems to be confirmed by our experiments with the OCS problem.

Conclusions

Hardware and software problems have been solved in order to implement a pseudo-exhaustive test. Segmentation cells have been developed compatible to the usual LSSD-rules. Efficient segmentation algorithms have been proposed, resulting in a minimal number of nodes to be cut. Combining the hardware and the algorithms, a pseudo-exhaustive test can be implemented automatically at low additional cost.

References

AgJS84	Agraval, V. D.; Jain, S. K.; Singer, D. M.: Automation in Design for Testability, Proc. IEEE Custom Integrated Circuits Conference, pp. 159 - 163, 1984
Aker85	Akers, S. B.: On the Use of Linear Sums in Exhaustive Testing, Proc. 15th International Symposium on Fault-Tolerant Computing, 1985
Bhat86	Bhatt, S. N.; Chung, F. R. K.; Rosenberg, A. L.: Partitioning Circuits for Improved Testability, Advanced Research in VLSI: proc. 4-th MIT conference, pp. 91 -106, April 7 - 9, 1986
EiWi77	Eichelberger, E.B.; Williams, T.W.: A logic design structure for LSI testability, Proc. 14th Design Automation Conference, pp. 462-468, June1977
Fuji85	Fujiwara, H.: Logic Testing and Design for Testability, MIT Press, 1985
GoFB77	Godoy, H. C.; Franklin, G. B.; Bottroff, P. S.: Automatic checking of logic design structure for compliance with testability groundrules, Proc. 14th Design Automation Conference, June 1977, pp.469 - 478
Hell90	Hellebrand, S: Synthese vollständig testbarer Schaltungen, Ph. D. Thesis, University of Karlsruhe, 1990
HeWu88	Hellebrand, S.; Wunderlich, H.-J.: Automatisierung des Entwurfs vollständig testbarer Schaltungen, Proc.18. Jahrestagung der Gesellschaft für Informatik, Hamburg 1988, Informatik-Fachberichte 188, Springer-Verlag
McBo81	McCluskey, E. J.; Bozorgui-Nesbat, S.: Design for Autonomous Test, IEEE Transactions on Circuits and Systems, Vol. Cas-28, No. 11, November 1981
McCl84a	McCluskey, E. J.: Verification Testing - A Pseudoexhaustive Test Technique, IEEE Transactions on Computers, Vol. c-33, No.6, June 1984
McCl84b	McCluskey, E.J.: A Survey of Design for Testability Scan Techniques, VLSI Design, Dec. 1984
McCl86	McCluskey, E.J.: Logic Design Principles: With Emphasis on Testable Semicustom Circuits, Prentice-Hall, 1986
NaSS86	Nahar, S.; Sahni, S.; Shragowitz, E.: Simulated Annealing and Combinatorial Optimization, Proc. 23rd Design Automation Conference, 1986
Rich83	Rich, E.: Artificial Intelligence, McGraw-Hill International Editions, 1983
RoLa84	Roberts, M. W.; Lala, M. Sc.: An Algorithm for the Partitioning of logic circuits, IEE Proceedings, Vol. 131, Pt.E., No.4,July 1984
Udel86	Udell, J.G.,Jr.: Test Set Generation for Pseudo-Exhaustive BIST, Proc. International Conference on Computer Aided Design, 1986
Wu90	Wunderlich, H.-J.: Rechnergestützte Verfahren für den prüfgerechten Entwurf hochintegrierter Schaltungen, University of Karlsruhe, 1990
WuHe88	Wunderlich, H.-J.; Hellebrand, S.: Generating Pattern Sequences for the Pseudo-Exhaustive Test of MOS-Circuits, Proc. 18th International Symposium on Fault-Tolerant Computing, Tokyo 1988