Full Scan Fault Coverage With Partial Scan*

Xijiang Lin†
Mentor Graphics Corporation
8005 S.W. Boeckman
Wilsonville, OR 97070-7777

Irith Pomeranz Sudhakar M. Reddy
Electrical and Computer Engineering Department
University of Iowa
Iowa City, IA 52242

Abstract

In this paper, a test generation based partial scan selection procedure is proposed. The procedure is able to achieve the same level of fault coverage as in a full scan design by scanning only a subset of the flip-flops. New measures are used to guide the flip-flop selection during the procedure. The proposed procedure is applied to the ISCAS-89 and the ADDENDUM-93 benchmark circuits. For all the circuits, it is possible to achieve the same fault coverage as that for full scan while scanning a portion of the flip-flops.

1 Introduction

Previous research on the selection of flip-flops for partial scan can be classified into three main categories: testability analysis based methods [1]-[4], structural analysis based methods [5]-[15], and test generation based methods [16]-[20]. Structural analysis based methods analyze the structure dependency graph(*S-graph*) [7] of the flip-flops and select the scanned flip-flops based on the properties of the S-graph. The methods in [5][6] cut all feedback loops in the S-graph such that the resulting circuit works in a pipeline fashion. In [7], Cheng and Agrawal proposed to cut all feedback loops except self-loops. In [13][14], valid state analysis is used to assess the impact of the feedback loops and hence, to guide the loop-breaking. Test generation based approaches utilize the information provided by a test generator to guide the scan selection process. In [16]-[20], sequential test generators are used to identify the aborted/undetectable faults in the circuit and the flip-flops are selected to make these faults detectable. In general, test generation based partial scan design leads to fewer scanned flip-flops than the other approaches. However the computational costs of test generation based partial scan selection are higher, especially if sequential ATPG is used throughout the partial scan selection procedure.

Although several partial scan procedures published before have improved the fault coverage of the circuits significantly, none of them have reported the same fault coverage as that achievable in full scan designs for ISCAS-89 benchmark circuits. In this paper, we propose a test generation based partial scan procedure driven by sequentially undetectable faults to achieve the same fault coverage as full scan design. Additionally we show that when it is used to achieve the same fault coverage as the previously published procedures it scans fewer flip-flops. The efficiency of the proposed procedure is due to the fact that the sequential ATPG is not invoked until after several flip-flops are selected for scanning using computationally efficient procedures and due to the new measures used to guide the selection of flip-flops to be scanned.

*Research supported in part by NSF Grand No. MIP-9725053

†Work reported was done while at the University of Iowa

In the proposed procedure we identify faults that cannot be detected by sequential ATPG, and select flip-flops to render them testable. We do this in three phases. In the first phase, we identify flip-flops whose states cannot be controlled or observed, or it is "difficult" to control their states. This information is obtainable without test generation and at relatively low computational cost and is used to select a subset of flip-flops to scan. In the second phase, we identify faults that cannot be detected by *any* sequential ATPG. These faults include sequentially redundant faults and partially detectable faults [23]. To identify these faults we use a computationally efficient method recently developed [21]. This method does not invoke sequential ATPG and hence determines the sequentially undetectable faults efficiently. Combinational tests for this set of faults in a fully scanned version of the circuit are used to guide the selection of flip-flops to be scanned in the second phase. We use two new concepts called matching factor and selectivity measure, derived from properties of the combinational test set, to determine the flip-flops to be scanned. In the third phase of the partial scan selection procedure, we use a sequential circuit ATPG to determine undetectable and hard to detect faults (aborted faults). Here, too, we use the concepts of matching factor and selectivity measure in choosing the flip-flops to be scanned.

The paper is organized as follows. In Section 2, we introduce measures used to guide the selection of flip-flops to be scanned, and describe the use of these measures. In Section 3, we describe the three-phase selection procedure. In Section 4, we present experimental results. Section 5 concludes the paper.

2 Measures for Flip-Flop Selection

In this section, we describe the measures we use to guide the selection of flip-flops for scanning. We also describe a procedure used later in the selection of flip-flops to be scanned.

2.1 Selection Measures

We define the activeness of a flip-flop with respect to a given test sequence and initial state as follows.

Definition 1 *The* **0(1)-activeness** *for flip-flop ff_i in a circuit is the number of times ff_i is set to 0(1) by an input sequence T.*

Note that the activeness depends on the sequence T and the initial state of the circuit before T is applied. In our procedure we measure the 0(1)-activeness of a flip-flop by starting the fault-free circuit in a known state and applying a random input sequence. The initial state is the state the circuit is driven into by a synchronizing sequence if one exists, else it is chosen randomly.

Let sn_i^v, $v \in \{0,1\}$, be the v-activeness of the flip-flop ff_i in the sequential circuit. The sequential 0(1)-possibility

of setting ff_i to 0(1), denoted by $SP_i(0)$ ($SP_i(1)$), is defined as:

$$SP_i(0) = \frac{sn_i^0}{sn_i^0 + sn_i^1} \quad SP_i(1) = \frac{sn_i^1}{sn_i^0 + sn_i^1}$$

In full scan design, sequential test generation is transformed into combinational test generation. Considering the test set generated by combinational test generation for the full-scan circuit, the number of times logic values 1 and 0 are assigned to a state variable indicates the desirability of the given logic value in a test set for the set of faults considered. Motivated by this observation, we propose the following measure.

Let N_c be the number of combinational test vectors for a set of faults F in a full scan design. Let cn_i^v be the number of times logic value v, $v \in \{0, 1, X\}$, is required on the state variable corresponding to ff_i in the combinational test set. We define the 0(1) combinational-desirability of ff_i as:

$$CD_i(0) = \frac{cn_i^0}{N_c} + 0.5\frac{cn_i^X}{N_c}$$

$$CD_i(1) = \frac{cn_i^1}{N_c} + 0.5\frac{cn_i^X}{N_c}$$

If the 0(1)-possibility and the (0)1-desirability of a flip-flop ff_i in a sequential circuit are the same or have close values, it is unlikely that ff_i will cause the sequential test generation for the selected faults to be difficult. However, if they are mismatched, the states required for detecting some faults may be hard to reach. We use this observation in deriving a measure called the *matching factor* M_i of flip-flop ff_i defined as follows:

$$M_i = |(SP_i(0) - CD_i(0))(SP_i(1) - CD_i(1))|$$

In the proposed procedure, a flip-flop with a large value of the matching factor is considered as a good candidate to be scanned. In the next subsection, we describe how this measure is used.

2.2 Flip-Flop Selection Based on Combinational Test Vectors

After a set of hard-to-detect and sequentially undetectable faults is identified (as described in the following section), our goal is to potentially make these faults detectable by scanning a subset of flip-flops.

Let CV be a set of test vectors generated by a combinational test generator for a set of faults F. We associate a combinational zero counter cfn_i^0, a combinational one counter cn_i^1, and a combinational observation counter cfo_i, with each flip-flop. If a combinational test vector V sets the flip-flop ff_i to 0(1) and V detects n faults in F, we increase $cfn_i^0(cfn_i^1)$ by n. If V detects a fault at an output corresponding to ff_i, cfo_i is increased by 1. The *selectivity measure* SM_i of a flip-flop ff_i is defined as:

$$SM_i = (1 + M_i) \times (0.8 \times (cfn_i^0 + cfn_i^1) + 0.2 \times cfo_i)$$

where the matching factor M_i is used as a weight to relatively increase the selectivity measures of flip-flops with mismatched sequential-possibility and combinational-desirability. In computing SM_i it can be seen that more weight is given to the combinational zero and one counts relative to the observation count. This was done to accommodate the fact that most sequentially undetectable faults are caused by the so-called unreachable states that are necessary to activate the faults, and are thus related to controllability and not to observability.

Given a set of hard-to-detect and sequentially undetectable faults F, we use the following procedure to select the flip-flops for scan based on the selectivity measure defined above.

Procedure Select_FF_For_Scan(*select_percentage*, *max_selected*, F)

(1) Apply a combinational test generator to find a test set CV that detects all the faults in F, assuming full scan.
(2) Determine the selectivity measures of the flip-flops that are not scanned and let n be the number of flip-flops with non-zero selectivity measures.
(3) Select min{*select_percentage* $\times$ n, *max_selected*} flip-flops with the highest selectivity measures to scan.
(4) Return the selected flip-flop set.

In the procedure *Select_FF_For_Scan()*, only the flip-flops with non-zero selectivity measures are considered as candidates to scan. The parameters *select_percentage* and *max_select* are used to control the number of flip-flops to be selected for scanning.

3 Partial Scan Selection Procedure

The complete three-phase partial scan selection procedure is described next.

3.1 Flip-Flop Selection: Phase I

During Phase I of the selection procedure, we identify flip-flops whose states cannot be observed at any primary output, and whose states cannot be controlled, or cannot be easily controlled. We use structural analysis to identify some of these flip-flops, and use functional analysis to identify the others.

3.1.1 Structural Analysis

Structural analysis is used to identify two types of flip-flops. The first type includes the flip-flops whose outputs do not have a path through the logic of the circuit(including flip-flops) to a primary output. These flip-flops are called *unobservable flip-flops*. The second type includes the flip-flops whose inputs do not have any path from the primary inputs. These are called *uncontrollable flip-flops*.

It is unnecessary to include all unobservable and uncontrollable flip-flops in the scan chain. We only need to include a minimum number of these flip-flops such that the resulting partial scan circuit does not have any unobservable and uncontrollable flip-flops. We use heuristics to select a minimal number of these flip-flops as described in the two procedures below.

Procedure Make_FFs_Observable( )

(1) Set $R = \emptyset$.
(2) Find all the unobservable flip-flops.
(3) If no unobservable flip-flop exists, return R.
(4) For each unobservable flip-flop, count the number of unobservable flip-flops that can reach it through paths in the circuit.
(5) Select the flip-flop ff_i that is reachable from the largest number of unobservable flip-flops. In case of ties, pick ff_i that has the maximum out-degree in the S-graph of the circuit.
(6) Set $R = R \cup \{ff_i\}$ and go to Step 2.

Procedure Make_FFs_Controllable( )

(1) Set $R = \emptyset$.
(2) Find all the uncontrollable flip-flops.
(3) If no uncontrollable flip-flop exists, return R.
(4) For each uncontrollable flip-flop ff_i, count the number of uncontrollable flip-flops to which the output of ff_i has a path.
(5) Select the flip-flop ff_i that can reach the maximum number of uncontrollable flip-flops. In case of ties, pick ff_i that has the maximum out-degree in the S-graph.
(6) Set $R = R \cup \{ff_i\}$ and go to step (2).

3.1.2 Functional Analysis

After including a minimal number of unobservable and uncontrollable flip-flops in the scan chain, functional analysis is performed to identify flip-flops whose state cannot be set to a desired binary state. These flip-flops are called unjustifiable flip-flops. We use random sequence simulation to identify the potentially unjustifiable flip-flips, and then apply a state justification procedure to verify that they are actually unjustifiable.

Again, because of correlation among the states of the flip-flops, not all unjustifiable flip-flops need be included in the scan chain. In our implementation, we use heuristics, similar to the ones used for uncontrollable flip-flops, to select a minimal set of unjustifiable flip-flops to be included in the scan chain. The procedure is given next for a value v of flip-flop ff_i.

Procedure Unjustifiable_FF_Selection(ff_i, v)

(1) Create an iterative array logic (ILA) model of the circuit including time frame 0 only. Assign v to ff_i.
(2) Set $done = FALSE$ and $u = -1$.
(3) While $done == FALSE$, do
   (a) Add time frame u to the ILA and perform implications starting from the present state lines of time frame $(u + 1)$.
   (b) If an implication cycle is found (i.e. setting ff_i to v in time frame zero implies setting ff_i to a known value at a previous time frame), return ff_i.
   (c) If no present state lines have known values at time frame u and no conflict is found, set $done = TRUE$. Else, set $u = u - 1$.
(4) Select a flip-flop ff_j with a known value at time frame $(u + 1)$ and having the maximum out-degree in the circuit S-graph among all the flip-flops with known values at time frame $(u+1)$. (Note that when this step is encountered it has been determined that setting ff_i to v at time frame zero implies setting ff_j to a known value at a previous time frame).
(5) Return ff_j.

After a minimal set of unobservable, uncontrollable, and unjustifiable flip-flops are included in the scan chain, we attempt to identify flip-flops which are "difficult" to control to zero or one. We do this by simulating a sequence of 64000 randomly selected input vectors on the fault-free circuit and noting the number of times a flip-flop has the value 0 and the value 1. Prior to simulating the random sequence of length 64000, we apply a synchronizing sequence of the circuit if one exists, else we initialize the circuit to a random state. The flip-flops that do not take either a zero or a one value at least 640 times are identified as difficult to justify flip-flops. Using the difficult to justify flip-flops, we construct a subgraph of the S-graph, called the S_d-graph. The nodes of the S_d-graph are those corresponding to the difficult to justify flip-flops and the edges between these nodes are those between the nodes in the original S-graph. Next, we determine a minimal feedback vertex set of the S_d-graph. The flip-flops corresponding to the minimal feedback vertex set of the S_d-graph are included in the scan chain. This completes Phase I of the selection process to identify scan flip-flops.

3.2 Flip-Flop Selection: Phase II

To achieve the same fault coverage as full scan design, all combinationally irredundant but sequentially undetectable faults must become detectable after a subset of the flip-flops is scanned. In [21], an efficient procedure was described to identify a large number of sequentially undetectable faults by using a combinational test generator. In Phase II, this procedure is used to identify sequentially undetectable faults, and flip-flops are selected to be scanned such that the identified sequentially undetectable faults become detectable. The procedure of Phase II is given next.

Procedure Selection_Phase_II(*untest_left*)

(1) Set $R = \emptyset$.
(2) Identify all the combinational undetectable faults and remove them from further consideration.
(3) Apply the procedure from [21] to determine a set of sequentially undetectable faults, F.
(4) While the number of faults in F is greater than *untest_left*:
   (a) Set D = Select_FF_For_Scan(*select_percentage*, *max_selected*, F).
   (b) Include all the flip-flops in D in the scan chain.
   (c) Set $R = R \cup D$.
   (d) Apply the procedure from [21] to remove the faults in F that are no longer declared undetectable by the procedure of [21].
(5) Return R.

The parameter *untest_left* is used as follows. In general, the more sequentially undetectable faults are identified, the better the procedure *Select_FF_To_Scan( )* can select appropriate flip-flops to scan. As the number of undetectable faults identified decreases with additional flip-flops included in the scan chain, the number of faults identified as sequentially undetectable by the procedure of [21] may not be sufficient to guide the flip-flop selection. As a result, more than the necessary number of flip-flops may be selected for scan. Therefore, the procedure *Selection_Phase_II( )* is terminated when *untest_left* or fewer sequentially undetectable faults are identified. The sequentially undetectable faults left will be passed to Phase III to avoid considering them again by the sequential test generator.

3.3 Flip-Flop Selection: Phase III

In this phase, a sequential test generator is used to identify sequentially undetectable and hard-to-detect faults. The flip-flops for scan are selected by considering these faults until the desired fault coverage is obtained.

Because sequential test generation is a computationally intensive procedure, we do not attempt to identify all the sequentially undetectable and hard-to-detect faults at the beginning of Phase III. Instead, Phase III is divided into iterations, where in each iteration, only a subset of the sequentially undetectable and hard-to-detect faults are identified. To alleviate the inaccuracy of incomplete information about sequentially undetectable and hard-to-detect faults, we assume that the faults that are not detected so far are hard-to-detect, and we use them when determining the flip-flops to be scanned.

The *combinational fault efficiency*(CFE), defined as the number of detected faults over the number of combinationally irredundant faults, is used as the termination condition for Phase III. When the preselected value of CFE is reached, Phase III is complete.

Procedure Selection_Phase_III(*max_number_notdetected*, *CFE*)

(1) Set $R = \emptyset$.
(2) Set *number_notdetected* = 0.
(3) While *number_notdetected* < *max_number_notdetected*:
   (a) Pick the next untried fault.
   (b) Apply the sequential test generator to generate a test for the fault picked.
   (c) If the fault is detected, go to step (a).
   (d) If the fault is aborted or if the fault is undetectable, increase *number_notdetected* by 1.
(4) Let F be the set of faults not detected in Step 3.

(5) Set D = Select_FF_For_Scan(*select_percentage*, *max_selected*, F).
(6) Include the flip-flops in D in the scan chain.
(7) Set $R = R \cup D$.
(8) Update the test sequence generated so far and carry out fault simulation by applying the updated test sequence.
(9) Compute the combinationally detectable fault coverage as the ratio of the number of faults detected so far to the number of combinationally irredundant faults. If it is less than *CFE*, go to Step (2).
(10) Return R.

In the procedure *Selection_Phase_III()*, the flip-flops for scan are selected iteratively. The parameter used to terminate each iteration is called *max_number_notdetected*, and the counter *number_notdetected* is modified dynamically.

After a flip-flop is selected to be scanned, its present-state variable can be treated as a primary input, and its next-state variable can be treated as a primary output. Therefore, the test sequence generated so far may detect additional faults if the fault effects are propagated to the scanned flip-flops. To take advantage of these detections, we update the test sequence generated so far by treating the scanned flip-flops as additional primary inputs. We assign the corresponding present state values of the fault-free circuit at each time frame to the new primary inputs. This guarantees that the updated test sequence obtains at least the same fault coverage as before. Then, all the yet undetected faults are simulated by applying the updated test sequence. The advantage of carrying out this modification is that the test sequence is available at the end of the selection procedure.

4 Experimental Results

The proposed procedure named *FALCON* (FAuLt COverage based scaN) was implemented in C++ and run on a HP-C180 workstation. It was applied to ISCAS-89 and ADDENDUM-93 benchmark circuits.

The partial scan results for the smaller benchmark circuits are shown in Table 1(Given on the next page.). Under columns *# of FF* and *Orig. FC*, the number of flip-flops and the fault coverage in the original circuits are shown. The numbers of flip-flops selected, execution times, and fault coverages obtained after each selection phase are shown under columns *FF*, *CPU*, and *FC*. The total number of scanned flip-flops is shown under column *Total Scanned FF*. Column *CPU* shows the total execution time in seconds taken by the procedure. It should be noted that a test sequence that achieves complete fault coverage is also generated by the procedure. For all the circuits listed in Table 1, the same fault coverages as full scan design are obtained (i.e., $CFE = 100\%$ was achieved). The control parameters used in the experiments are listed as follows:

Table 2. The results of applying *FALCON* to larger benchmark circuits

Circuit	#	*CFE*=80%			*CFE*=95%			*CFE*=100%		
Name	FF	Scan	FC	CPU (hr)	Scan	FC	CPU (hr)	Scan	FC	CPU (hr)
s9234	228	61	77.59	1.59	67	89.56	1.92	123	93.47	2.65
s13207	669	106	82.46	5.39	126	94.48	6.09	216	98.46	8.65
s15850	597	161	78.41	6.75	201	92.22	11.4	290	96.68	18.9
s38584	1452	84	77.41	5.55	184	92.84	19.1	339	95.85	31

1. In Phase I, 64,000 random vectors are applied to find the activenesses of each flip-flop.
2. In Phase II, 6,400 random vectors are applied to find the activenesses of the flip-flops. Moreover, we set $untest\_left = 30$, $select\_percentage = 20\%$, and $max\_selected = 4$.
3. In Phase III, we set $max\_number\_notdetected = 100$, $select\_percentage = 5\%$, and $max\_selected = 10$.

In the next experiment, reported in Table 2, the proposed procedure is applied to the larger ISCAS-89 benchmark circuits. The parameter *CFE* is set to be 80%, 95%, and 100%. For different *CFE*, the number of scanned flip-flops, fault coverage after scan, and the execution time are shown under columns *Scan*, *FC*, and *CPU*. It must be pointed out that the fault coverages for *CFE*=80% and *CFE*=95% shown in Table 2 are lower bounds on the fault coverage that can be obtained by the test generator. This is because faults that are not targeted due to the limit on the number of faults considered in each iteration of Phase III may be detectable if targeted by the ATPG.

Table 3. Comparison of partial scan designs by different systems

Circuit	*FALCON*		IDROPS		OPUS		E-STG	
Name	Scan	FC	Scan	FC	Scan	FC	Scan	FC
s382	5	100	-	-	9	98.8	-	-
s400	4	97.64	-	-	9	97.4	4	97.64
s420	0	47.44	4	22.9	-	-	3	22.32
s444	4	96	4	94.7	9	96	-	-
s526	4	98.92	4	94.2	15	99.6	4	93.15
s1423	7	97.76	15	95.8	41	96.2	15	74.39
s5378	24	98.48	34	98.4	48	93.9	36	96.2
s9234.1	30	93.77	43	93.66	-	-	-	-
s13207.1	101	91.45	128	91.20	-	-	-	-
s15850.1	124	93.00	108	89.60	-	-	-	-
s38584.1	126	92.16	290	91.60	-	-	-	-

In Table 3, the results of the proposed procedure are compared with the partial scan procedures IDROPS[20], OPUS[17], and E-STG[19]. For the comparison, we stopped selecting flip-flops to be scanned when the achieved fault coverage became greater than or equal to the highest reported by any of the three methods we are using for comparison. Only IDROPS considered larger ISCAS-89 benchmark circuits. However, these circuits are the ".1" circuits. In Table 3 we include these circuits. It can be seen that the proposed method requires fewer scan flip-flops to achieve the fault coverage achieved by the other methods.

5 Summary

A three-phase procedure for partial scan selection was described in this paper. The procedure is test generation based and the design goal is to select as few flip-flops for scan as possible while obtaining the fault coverage of full scan design. Since sequential test generation is a time consuming process, the proposed procedure first selected flip-flops based on structural and functional analysis. In the second phase, a subset of undetectable faults was identified and flip-flops were selected for scan to make the identified undetectable faults detectable. In the third phase, a sequential ATPG was used to identify hard-to-detect and additional undetectable faults, and the scan selection was based on this information. For all the circuits considered, the proposed procedure selected a relatively small percentage of flip-flop while obtaining the same fault coverage as full scan design.

References

[1] E. Trischler, "Incomplete Scan Path with an Automatic Test Generation Methodology," *Proc. of Intl. Test Conf.*, pp. 153-162, November 1980.

Table 1. The results of applying *FALCON* to smaller benchmark circuits

Circuit Name	# of FF	Orig. FC	Scan									Total Scanned FF	CPU (sec)
			Phase I			Phase II			Phase III				
			FF	CPU	FC	FF	CPU	FC	FF	CPU	FC		
s208	8	69.77	3	0.4	93.95	0	0.23	-	1	0.86	100	4	1.5
s298	14	88.64	0	0.94	-	0	0.24	-	1	2.47	100	1	3.7
s344	15	97.95	0	0.72	-	0	0.16	-	1	1.02	100	1	1.9
s349	15	97.43	0	0.71	-	0	0.16	-	1	0.97	99.43	1	1.85
s382	21	94.24	4	0.95	99	0	0.22	-	1	2.11	100	5	3.3
s386	6	81.77	1	0.58	91.67	1	0.56	96.35	1	1.67	100	3	2.83
s400	21	93.4	4	0.98	97.64	0	0.24	-	1	2.12	98.58	5	3.36
s420	16	47.44	6	1.06	83.02	2	1.37	98.6	1	1.98	100	9	4.42
s444	21	92.2	4	1.04	95.99	0	0.4	-	2	2.42	100	6	3.88
s526	21	83.24	1	8.63	89.37	0	0.88	-	4	322	99.82	5	331
s641	19	87.37	1	1.26	100	0	0.36	-	0	2.65	100	1	4.29
s713	19	82.62	2	1.94	93.46	0	0.4	-	0	2.18	93.46	2	4.55
s820	5	95.88	0	0.75	-	1	2.59	98.94	1	7.72	100	2	11.1
s832	5	94.14	0	0.77	-	1	2.81	97.36	1	7.5	98.39	2	11.2
s838	32	35.36	21	3.04	89.38	3	11.3	98.48	1	8.2	100	25	22.6
s953	29	99.07	0	2.24	-	0	2.24	-	3	31.3	100	3	35.8
s1196	18	99.76	0	1.93	-	0	2.13	-	2	13.3	100	2	17.4
s1238	18	94.69	0	2.05	-	0	2.65	-	2	14.5	94.91	2	19.3
s1423	74	96.5	7	12.4	97.76	0	2.29	-	16	1557	99.08	23	1572
s1488	6	97.31	0	1.28	-	1	7.21	100	0	9.91	100	1	18.5
s1494	6	96.61	0	1.26	-	1	7.87	100	0	11.2	100	1	20.4
s5378	179	79.51	20	17.9	97.33	4	21.5	98.48	15	215	99.13	39	255
s967	29	98.22	0	2.18	-	0	1.96	-	2	89.3	100	2	93.5
s3330	132	74.01	1	7.58	82.75	28	29.8	99.06	6	69.2	100	35	107
s4863	104	97.36	0	11.4	-	4	9.85	99.41	9	60	100	13	81
prolog	136	70.35	5	10.4	78.55	24	32.7	94.67	7	77	95.46	36	120

[2] M. Abramovici, J. J. Kulikowski, and R. K. Roy, "The Best Flip-flops to Scan," *Proc. of the Intl. Test Conf.*, pp. 166-173, October 1991.

[3] P. S. Parikh and M. Abramovici, "Testability-Based Partial Scan Analysis," *Journal of Electronic Testing*, pp. 225-259, August/October 1995.

[4] K. S. Kim and C. R. Kime, "Partial Scan Flip-Flop Selection by Use of Empirical Testability," *Journal of Electronic Testing*, pp. 225-259, August/October 1995.

[5] R. Gupta, R. Gupta, and M. A. Breuer, "BALLAST: A Methodology for Partial Scan Design," *Fault Tolerant Computing Symposium*, pp. 118-125, June 1989.

[6] A. Kunzmann and H.-J. Wunderlich, "An Analytical Approach to the Partial Scan Problem," *Journal of Electronic Testing*, pp. 163-174, May 1990.

[7] K.-T. Cheng and V.D. Agrawal, "A Partial Scan Method for Sequential Circuits with Feedback", *IEEE Trans. on computers*, vol. 39, pp. 544-548, April 1990.

[8] D. H. Lee and S. M. Reddy, "On Determining Scan Flip-Flops in Partial Scan Design", *Proc. of Intl. Conf. on Computer-Aided Design*, pp. 322-325, Nov. 1990.

[9] V. Chickermane and J.H. Patel, "An Optimization Based Approach to the Partial Scan Design Problem," *Proc. of Intl. Test Conf.*, pp. 377-386, September 1990.

[10] S. T. Chakradhar, A. Balakrishnan, and V.D. Agrawal, "An Exact Algorithm for Selecting Partial Scan Flip-Flops", *Proc. of Design Automation Conf.*, pp. 81-86, 1994.

[11] T. Orenstein, Z. Kohavi, and I. Pomeranz, "Implicit Computation of Minimum-Cost Feedback-Vertex Sets for Cycle Breaking in Directed Graphs," *Journal of Electronic Testing*, pp. 71-82, August/October 1995.

[12] S. E. Tai and D. Bhattacharya, "A Three Stage Partial Scan Design Method to Easy ATPG," Journal of Electronic Testing, pp. 95-104, August/October 1995.

[13] D. Xiang, S. Venkataraman, W.K. Fuchs, J.H. Patel, "Partial Scan Design Based on Circuit State Information", *Proc. of Design Automation Conf.*, pp. 807-812, 1996.

[14] D. Xiang and J. H. Patel, "A Global Algorithm for the Partial Scan Design Problem Using Circuit State Information," *Proc. of Intl. Test Conf.*, pp. 548-557, 1996.

[15] A. Balakrishnan and S. T. Chakradhar, "Peripheral Partition and Tree Decomposition for Partial Scan," *Intl. Conf. on VLSI Design*, pp. 181-186, January, 1998.

[16] V. D. Agrawal, K. T. Cheng, D. D. Johnson, and T. Lin, "Designing Circuits with Partial Scan," *IEEE Design and Test of Computers*, vol. 5, pp. 8-15, April 1988.

[17] V. Chickermane and J.H. Patel, "A Fault Oriented Partial Scan Design Approach," *Proc. of Intl. Conf. on CAD*, pp. 400-403, November 1991.

[18] I. Park, D. S. Ha, and G. Sim, "A New Method for Partial Scan Design Based on Propagation and Justification Requirements of Faults," *Proc. of Intl. Test Conf.*, pp. 413-422, 1995.

[19] V. Boppana and W. K. Fuchs, "Partial Scan Based on State Transition Modeling," *Proc. of Intl. Test Conf.*, pp. 538-547, October 1996.

[20] M. S. Hsiao, G. S. Saund, E. M. Rudnick, and J. H. Patel, "Partial Scan Selection Based on Dynamic Reachability and Observability Information," *Intl. Conf. on VLSI Design*, pp. 174-180, January, 1998.

[21] X. Lin, I. Pomeranz, and S. M. Reddy, "On Finding Undetectable and Redundant Faults in Synchronous Sequential Circuits", *Intl. Conf. on Computer Design*, pp. 498-505, October 1998.

[22] X. Lin, I. Pomeranz, and S. M. Reddy, "MIX: A Test Generation System for Synchronous Sequential circuits", *Proc. 11th Intl. Conf. on VLSI Design*, pp. 456-463, January 1998.

[23] I. Pomeranz and S.M. Reddy, "Classification of Faults in Synchronous Sequential Circuits," IEEE Trans. on Computers, Sept. 1993, pp. 1066-1077.