VLSI Design
1994, Vol. 1, No. 4, pp. 299–311
Reprints available directly from the publisher
Photocopying permitted by license only

Partial Reset: An Alternative DFT Approach

BEN MATHEW and DANIEL G. SAAB
Center for Reliable and High-Performance Computing, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign

Design for testability (DFT) techniques reduce testing costs at the price of extra hardware. Among the many DFT techniques that have been proposed for this task are full scan, partial scan and hardware reset. In this paper we explore a relatively new DFT method, called partial reset. Reset lines are added to only a subset of the flip-flops and obtain reasonably high coverage. This approach has lower overhead in terms of test application time and hardware area when compared to previous ones. Further enhancement of the controllability is obtained by using multiple reset lines. The configuration of these multiple reset lines is described. This technique has been evaluated on the 1989 ISCAS sequential benchmark circuits and obtained favorable results.

Key Words: *Design for testability; Partial reset; Multiple reset lines; Test application time reduction*

Test generation for sequential circuits is a very expensive and time consuming process. It is known to be NP-complete even for just combinational circuits [1]. To accelerate test generation, improve fault coverage and reduce test application time (TAT), the circuit is modified. The circuit modification process during the design phase is called Design For Testability (DFT) [2], [3]. Many ad hoc and algorithmic DFT techniques have been proposed. Among them are test point insertion [4], built-in self test (BIST) [3], hardware reset and full/partial scan. To choose a DFT technique for a circuit, the strengths and weaknesses of each approach must be evaluated. Characteristics that are taken into consideration include fault coverage obtained, ease of implementation/applicability, test application overhead (time and memory) and hardware overhead. (Hardware overhead is discussed for comparison, usually with respect to a circuit with no DFT techniques added.) A brief review of previous methods is now presented for comparison purposes.

DFT Techniques to Improve Fault Coverage

Hardware reset is a mechanism which alleviates the initialization problem by bringing the sequential machine to a known initial state (usually a state where all state variables are 0). The usefulness of this arises in situations where the sequential test generator (STG) aborts faults when it cannot find a homing or synchronizing sequence [5]. It also reduces the test length if a synchronizing sequence is very long. The hardware overhead includes the logic that resets the flip-flops in addition to the routing of the reset line. This price is paid for higher fault coverage and simplified test generation [6]. TAT is usually reduced when compared to a circuit without DFT.

Full scan is a technique that permits complete controllability and observability [3]. It transforms a sequential circuit virtually to a combinational circuit. This is done by having a scan mode where the multiplexed flip-flops (FFs) are connected in a serial shift-register. Under normal operation, the FFs behave in the ordinary fashion where they all load values in parallel from their respective inputs. In test, or scan mode, any arbitrary value can be shifted, or scanned, into the FFs. Their values can be examined by shifting, or scanning out, the contents. The number of test vectors generated for a given fault coverage is less than for a circuit without scan since the test generation is simplified. However, TAT may be high due to the required scanning in/out of vectors. Furthermore, full scan can have a hardware overhead from 5% to well over 30% depending on the type of scan design used and the ratio of combinational logic to latches [3].

This research was supported by the Semiconductor Research Corporation Contract 92-DP-109.

Partial scan reduces the hardware overhead of full scan by requiring only a subset of the FFs be included in the scan path. For circuits with a large number of FFs, the TATs may be reduced since less scan clocks are required in between regular clocks. Fault coverages approaching those of full scan can sometimes be obtained with partial scan. A myriad methods for selecting the FFs to be scanned have been proposed. Trischler has suggested using testability measures as heuristics for selecting FFs [7]. In [8], the authors generate test vectors and then select FFs that, if scanned, would make all remaining undetected faults detectable. In [9], a minimal set of FFs that break cycles was presented. Lee and Reddy have expanded the cycle cutting method with that of minimizing sequential depth in [10]. The authors in [11] have described an algorithm that is fault-oriented and uses testability measures to rank the FFs in the order of their effect on the fault coverage. The OPUS-2 [12] algorithm expanded the method of selecting the optimal solution from structural analysis with a procedure to select FFs that aided most in detecting faults aborted by a test generator.

Scan designs have several disadvantages. The routing of multiple lines and the more complex FFs contribute to the larger silicon area required. Scan designs will require at least three additional I/O pins unless the PIs and POs are multiplexed. Testing of each individual chip may take longer since the test vectors will have to be scanned in before the faults can be detected. To detect timing faults, the FFs are even more complex [13]. The overall circuit clock speed may have to be reduced due to the additional logic in the scan path FFs.

DFT Techniques to Reduce TAT

As stated above, the time and resources required to test each individual chip is just as important a consideration as fault coverage. The test application times, specifically, should be minimized [14]. Many methods have been suggested towards this end. Williams and Angell suggested that the "shift-register" used in testing can be split into several shorter ones [15] (called *multiple scan chains*). BIST techniques [3] usually have low TATs since no off-chip delays exist. Lee and Shin propose *partial parallel scan* where FF inputs are multiplexed with PIs and POs [14]. The FFs are divided into N scan groups. The k FFs in each scan group are multiplexed to different PIs and POs so that all k FFs in each group can be scanned (in or out) in parallel. So, all FFs can be scanned in, at most, N clock cycles. Pradhan and Saxena have combined an STG and full scan to reduce TATs [16]. This method uses full-scan to initialize the circuit and the STG to activate and propagate the faults without using scan. Thus, the need to scan-in and scan-out vectors between regular clock cycles is eliminated. Gupta et al have developed a system which partitions a hierarchical description of a circuit, generates test for the individual parts and combines the tests for all of the combinational logic [17]. This method has to be used in conjunction to a DFT technique. In general, these TAT reduction DFT techniques have high area overhead.

A method that does not rely on any additional hardware is the one presented by Wang et al [18]. The authors demonstrate that the number of vectors can be reduced during the test generation process by choosing the next target fault based on a fault selection measure. This method can be used in along with our DFT technique to reduce TAT even further (with deterministic STGs).

A New Alternative

Recently it was suggested that a selected subset of the FFs can be reset if a synchronizing sequence is prohibitively long or nonexistent [19]. This technique has been dubbed *partial reset* (PR), and the conventional reset method is hereafter called *full reset* (FR) [20]. The suite of DFT techniques previously available did not provide the tradeoff in fault coverage, TAT and hardware overhead as PR does. We demonstrate that PR is a good addition to the set of DFT techniques allowing more flexibility to designers.

With PR, there is not a unique state to which that the sequential machine returns as with FR [20]. FR, as a consequence, can only be used once in a test vector sequence (usually at the beginning). PR, on the other hand, can be used *within* test sequences to give additional controllability. Direct transfers between two states that differ only in the resettable state variables are possible.

In [20], the authors were the first to show that PR improved fault coverage and reduced test length over FR, using state space information. In this paper we examine the effect of PR on fault coverage and TAT for circuits described at the structural (gate) level using the single node stuck-at fault model. This allows the processing of larger circuits.

The following section describes PR and motivates its use. Next, we describe the FF selection process and reveal how multiple reset lines can be used to augment controllability. Lastly, we present discussion and results obtained using the ISCAS 89 sequential benchmarks [21] and concluding remarks.

PARTIAL RESET

PR's main strength is its simplicity. Only a small subset of the FFs in most sequential circuits cause initialization problems. PR only resets those FFs. Due to this, the PR line can be treated as normal input to the sequential machine, after the circuit is initialized. The small number of FFs selected reduces the interference in state transitions when compared to FR. Thus, PR can be used at the beginning of a test sequence to initialize the circuit, as well as within a sequence to increase controllability.

PR promises lower routing/hardware (than scan designs) and higher fault coverage (than the original circuit) at the price of a slightly more complicated test generator. (The MOSIS Service's CMOSN standard cell family's resettable D-FF is only 14% larger than the normal D-FF, as compared to the scannable D-FF which is 57% larger [22].) Furthermore, PR facilitates relatively faster TATs since it does not require the scanning in of test vectors as with partial or full scan. Thus, the testing proceeds at normal clock speeds.

Pomeranz and Reddy have shown, with experiments on the MCNC synthesis benchmarks using the single transition fault model, that fault coverage either remained the same or increased (for all the circuits for which they generated tests) with PR when compared to FR. Test sequence length and test generation time were usually reduced with PR than with FR. The tradeoff is in the higher test generator complexity, since a homing or synchronizing sequence needs to be generated to initialize the sequential machine without FR capability. This paper will expand on these results.

In a previous work dealing with PR which was based on state transition faults, the reset mechanism is assumed to be fault-free. Our method makes no such assumption and the fault coverages we present include the effect of faults on the reset lines. One of two events occur when the reset line is faulty.

1. If the reset line is stuck-at the inactive value, the FFs affected will not be able to reset. They will behave as one of the unselected FFs (assuming a single stuck-at fault).
2. If the reset line is stuck-at the active value, the affected FFs will be stuck at the reset value.

PR is illustrated using the circuit shown in Figure 1 with the indicated stuck-at fault. This particular fault manifests itself as a single transition fault [23] (darker transition arrow) as shown in the state transition graph (Figure 2). A faulty circuit with this fault will go to state $x_0x_1 = 11$ instead of $x_0x_1 = 01$ when

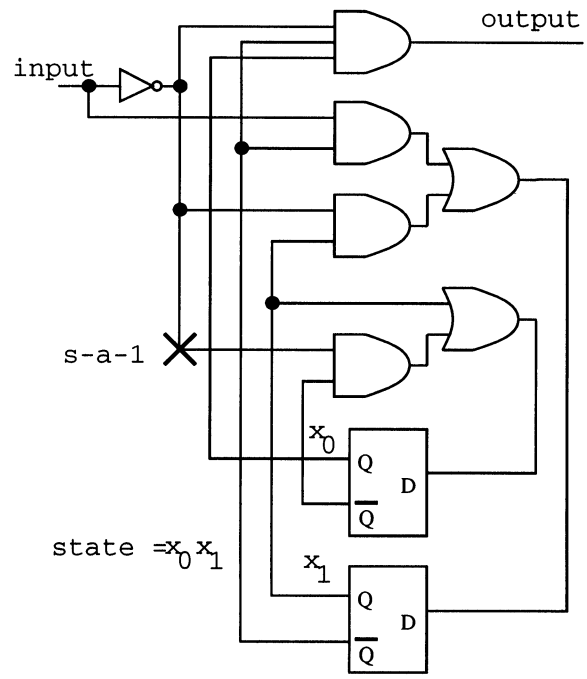

FIGURE 1 Example faulty circuit.

the input of 1 is applied in the state $x_0x_1 = 00$. The arrows are labeled with input/output values.

It is first shown that this fault is untestable without any DFT techniques. All possible transitions from state $x_0x_1 = 00$ (the state from which the faulty transition originates) are shown in Figure 3. The fault is activated by applying the input value 1 causing the machine to arrive at state 01/11, where the notation is good-state/faulty-state. The output produced in this transition is 0 for both circuits. Finally, when either 0 or 1 is applied, both the good and faulty circuits arrive at identical states and produce iden-

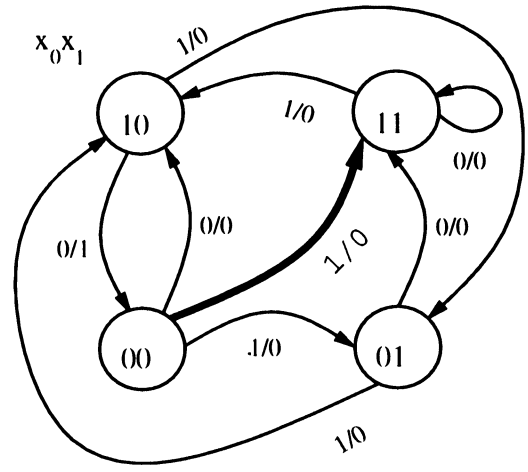

FIGURE 2 State transition graph of example circuit.

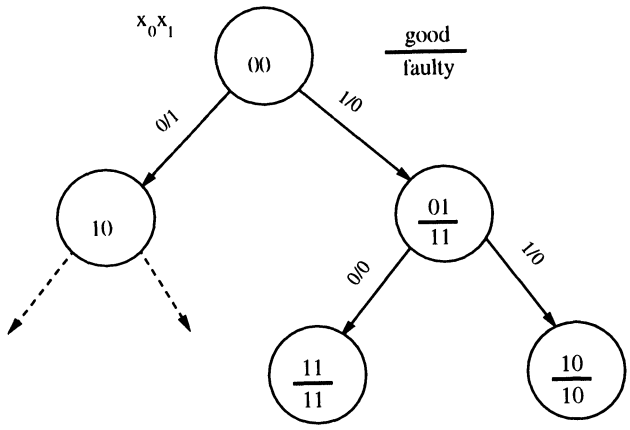

FIGURE 3 Transitions from state 00.

tical output sequences. Note that having FR does not assist in detecting this fault since this is not an initialization problem.

If the FF which is represented by the state variable x_1 is modified so that it resets to 0 when the reset input is activated, the fault becomes detectable. If the reset is applied (along with the normal input), the state transition graph would change as shown in Figure 4. The good and faulty transitions are shown for the state $x_0x_1 = 00$.

The sequence that detects the fault is shown below. Any STG that can initialize both the good and faulty circuits can be used. In the sequel, the notation of $A \xrightarrow[o_g/o_f]{i} B$ indicates that a transition is made from state A to B when the input i is applied producing the output o_g and o_f in the good and faulty circuits respectively [20]. When the good machine is in state A and the faulty one is in B, A/B is used for state representation. The input $0r$ implies that 0 is applied to the circuit's normal input along with reset. When both of the circuits produce the same value, only one

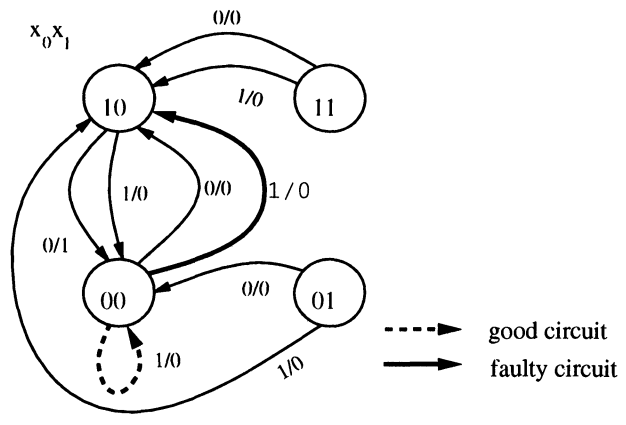

FIGURE 4 State transition graph if reset used.

output value will be shown. The uninitialized value is represented by X. The state machine begins with both state variables uninitialized ($x_0x_1 = XX$). The sequence is

$$XX \xrightarrow[0]{1r} X0 \xrightarrow[0]{1} \frac{01}{X1} \xrightarrow[0]{1}$$

$$10 \xrightarrow[0]{0} 00 \xrightarrow[0]{1r} \frac{00}{10} \xrightarrow[0/1]{0} \frac{10}{00}.$$

Some points require further elaboration. The reset input is activated twice; the first time is to aid in initializing. The good circuit becomes fully specified (initialized) before the faulty one. The states for both circuits become the same after the third input vector. The reset input is utilized the second time to activate the fault. Thus the fault is detected when the PR technique is implemented.

Finally, we note in passing that the observability can possibly be improved in a manner similar to which PR improves controllability. A subset of unobservable FFs can be selected to be probed in some manner, taking into consideration the aliasing problem.

SELECTING FLIP-FLOPS AND RESET VALUES

The first problem in using the PR method is choosing the FFs which should be made resettable for the best results. In the example above, the fault will not be detected if state variable x_0 was made resettable. Therefore, it is important to select the right resettable FFs.

Our FF selection is made based on the circuit structure. A graph called an *s-graph* is constructed to aid in this task. The s-graph is a directed graph $G = (V, E)$ where vertices and edges represent FFs and paths between FFs, respectively. An edge (v_i, v_j) exists in the s-graph if there is a path through combinational logic from the FF represented by v_i to the FF represented by v_j. The main issue in adding testability to most sequential circuits involves the cycles of storage elements and combinational logic [9]. (*Synchronous* sequential circuits are assumed.) Logic values circulate within the sequential machine due to the cycles and reduce the controllability from the primary inputs. If the cycles can be broken, the circuit will become easier to initialize and test. The cycles are "broken" by making one of the FFs in the cycle resettable. The s-graph allows a better view of the circuit's cyclic structure. The problem, then, is to find a minimum set of vertices V' so that $G - V'$ is

acyclic. This is similar to the NP-complete problem known as "feedback vertex set."

A lesser contributor to difficulties in sequential test generation is the sequential depth. PR produced better results for most circuits when it was tailored to tackle the problem of cycles rather than sequential depth.

The problem of breaking cycles has been attacked by researchers concerned with partial scan FF selection. Lee and Reddy have proposed a near-minimal solution [10]. We have adopted their techniques to select FFs that are made resettable. With partial scan design, the selected FFs can be controlled to any desired value independently. (Observability is provided as well.) PR only allows controllability to one value for each FF. The selected FFs cannot be reset to this value independently. The benefit, however, comes in lower hardware requirements (e.g. routing of multiple lines, I/O pins). Note that in highly timing-dependent designs, the FF selection method can be modified so that FFs on the critical path are not selected.

Once the FFs to be reset have been selected, the next task is to select the reset values. Traditionally, the reset value of a FF is assumed to arbitrarily be 0. We propose a design where the selected FFs are allowed to reset to either 0 or 1, whichever will be more beneficial to the testing process. Note that this is not the same as having both *preset* and *clear* inputs, which requires more silicon area. The FF will only reset to *one* of the values in {0, 1}, which is determined *a priori* at design time. In this scheme, the selected FFs do not all have same reset values.

Reset values are selected in one of two ways. The first is based on SCOAP [24]. SCOAP is a testability measure that indicates the degree of controllability and observability of the nodes in a circuit. (Observability values were not used here.) Selected FFs were reset to the value $v \in \{0, 1\}$ where v is the value that had a lower controllability value. In some circuits, the 0- and 1-controllabilities of a node may both be the same, preventing SCOAP from providing any helpful information. This is especially true in highly sequential circuits where many nodes are equally uncontrollable to either value. In such cases, a second method utilizing simulation of random vectors was used. The number of times 0 and 1 occur at the output of the selected FFs are counted. The logic value $\in \{0, 1\}$ having the minimum occurrence count is used as the reset value. A random reset value is chosen if neither of these two methods are helpful. This procedure is outlined in Figure 5.

```
create s-graph
select FFs to be reset
compute SCOAP values
for all reset FFs {
  if SCOAP value helpful
    record resettable value
  else {
    simulate random vectors and count occurrence
    if occurrence count helpful
      record reset value
    else
      select random value
  }
}
```

FIGURE 5 Selecting Reset Values.

MULTIPLE RESET LINES

Up until now, only a single reset line controlling all selected FFs has been considered. To provide more alternatives a method utilizing multiple reset lines is described. If designers have the freedom to add additional PI pins, this technique might improve fault coverage for a small cost. The number of resettable FFs is not increased. Two different schemes utilizing multiple reset lines were investigated below.

Selectable Reset Value

The first approach to increase controllability is to allow selected FFs to change to either 0 *or* 1. This is similar to partial scan in that the FFs can be controlled to both values but is different since the FFs do not have individual control. This technique using two reset lines has been implemented. The function of the reset lines are described in Table I. The effect

TABLE I
Selectable Reset Value

	Predetermined reset value	
r_1r_2	1	0
00	normal	normal
01	set to 1	reset to 0
10	normal	normal
11	reset to 0	set to 1

the reset lines have on a particular FF depends on its assigned reset value. The assigned reset value is only used to separate the selected FFs into groups so that they do not all reset to the same value at the same time.

This potentially doubles the number of transitions from each state when compared to single reset. Note that some FFs will reset to 0 while others reset to 1 depending on the assigned reset value. This method is a simple extension of the single reset technique.

Partitioning FFs

The second method is slightly more complex. It partitions resettable FFs into various sets. Each reset line, which is a PI, will reset the FFs in one of the sets. Fig. 6 shows pictorially how multiple reset lines can be used if the selected FFs are partitioned into two groups. Some heuristics for partitioning the FFs have been developed which produced encouraging results.

How partitioning is helpful is illustrated using the circuit from [25] which has the state transition graph shown in Figure 7. The graph shows a faulty transition due to the stuck-at 0 fault at the node marked "w" in the circuit of Figure 8. The fault can be activated from the state $p_1p_2p_3 = 01X$ (with input = 0), but only 010 is a valid state. The circuit makes the transition $01X \xrightarrow{0/1} \frac{110}{010}$. Since states $p_1p_2p_3 =$ 110 and 010 are equivalent, the fault is not detectable. Assume that the FFs representing p_1 and p_2 are chosen to be made resettable to 1 and 0 respectively. By resetting both FFs simultaneously while applying 0 to the input, both the good and faulty circuits arrive at state 100. Therefore, the single PR technique does not help. If the FFs were given independent control, denoted r_1 and r_2, the following sequence would detect the fault:

$$XXX \xrightarrow[X]{1r_1r_2} 100 \xrightarrow[1]{0} 010 \xrightarrow[1]{0_2} \frac{100}{000} \xrightarrow[1/0]{0} \frac{010}{001}.$$

The number of transitions in the state diagram is increased when additional inputs are added to the sequential machine. This is essentially what reset

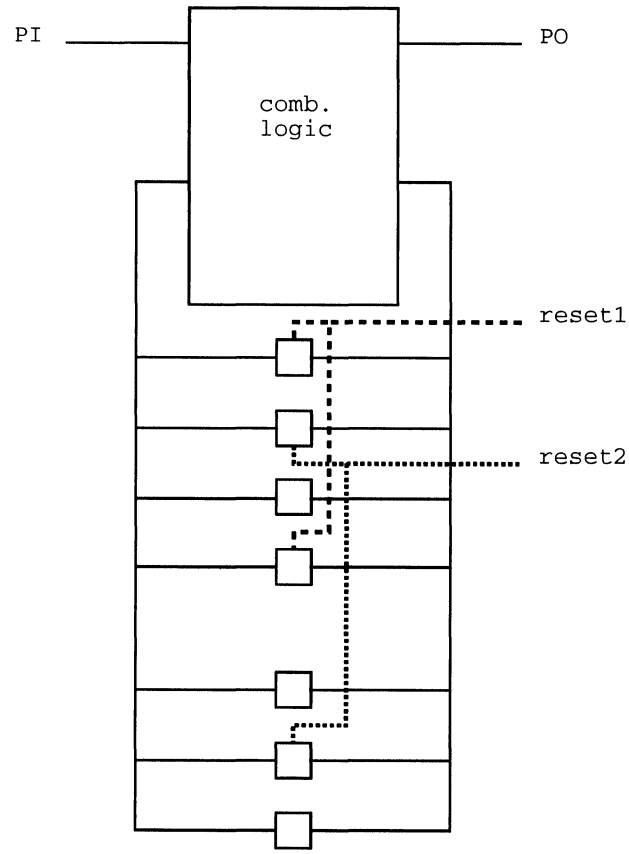

FIGURE 6 Multiple Reset Lines.

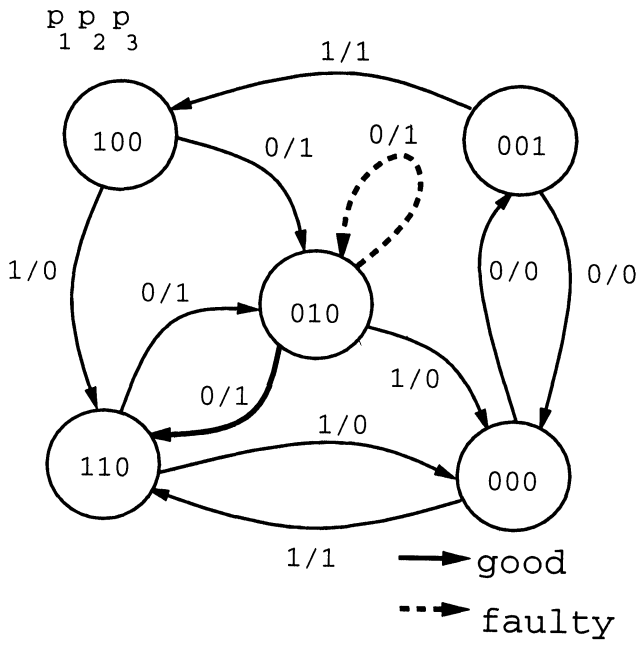

FIGURE 7 Transition Graph of Example Circuit.

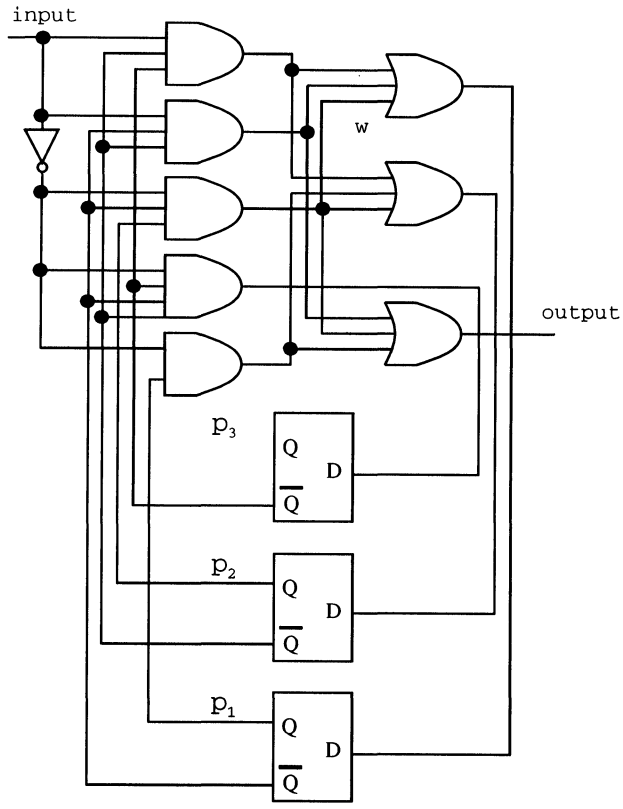

FIGURE 8 Example circuit.

lines do. By partitioning the FFs into separate groups, the machine has more independent control over them. The freedom to reset all the selected FFs simultaneously is also retained. Figure 9 illustrates the maximal number of transitions from a state S for a single-input sequential machine with two reset

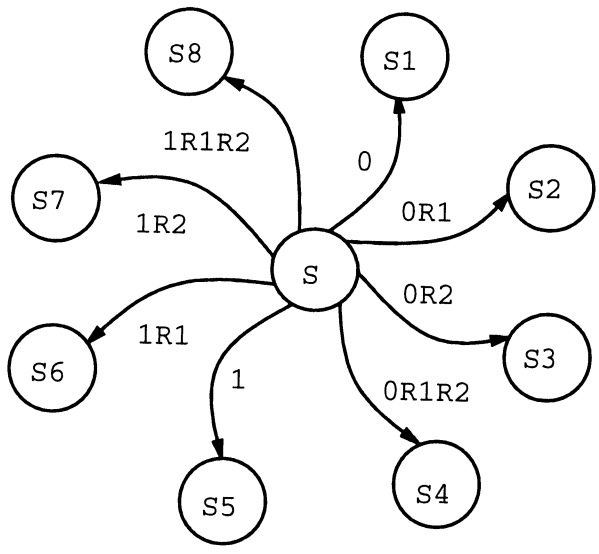

FIGURE 9 Additional Transitions Provided by Reset Lines.

lines. In general, the factor of transitions multiplied from each state is upper bounded by $\Sigma_{i=1}^{l_r} \binom{l_r}{i}$, where l_r is the number of reset lines.

Partitioning Criterion

The selected FFs are put into a new graph, called the *c-graph,* that will be colored so that each color will represent a distinct reset line. A c-graph's nodes are the selected FFs and there is an edge between every node that is in the same strongly-connected component in the s-graph. The c-graph edges will be initially weighted according to the distances between the selected FFs in the s-graph. This graph may be a disconnected set of cliques, where selected FFs in the same strongly connected components in the s-graph are in a clique together.

Next, the c-graph is colored taking into consideration the edge weights. If there is no edge between two nodes, there is no dependence on the colors that may be assigned to these two. Therefore, if two FFs are not in the same strongly-connected component in the s-graph, they can be assigned the same reset line (color) without affecting their independent controllability. The larger the edge weights (distance between selected FFs), the more likely the nodes will be assigned the same color. Distance is calculated as the number of gates between two points in the circuit. Our premise here is that FFs that are closer to each other will benefit more from being controlled by different reset lines since FFs far apart affect different parts of the circuit.

Another heuristic is based on the following observation: two FFs that fan out to the same combinational logic block should be assigned different reset lines for more controllability. Higher controllability of the fanout nodes can be obtained by using independent reset lines than if the FFs were both controlled by the same reset line. The procedure adjusts the appropriate edge weights so the coloring will reflect this heuristic. We found that by randomly perturbing the edge weights slightly, more of the circuits could be successfully partitioned.

Graph coloring is known to be NP-complete and no good polynomial-time approximations exist [26]. Since the problem at hand is simpler and more flexible than general graph coloring, heuristics can be used to get results in a fairly efficient manner. The coloring procedure is shown in Figure 10. Non-zero positive integers represent the colors. (There are as many colors as reset lines.) Starting with the node of highest degree, the procedure assigns each node a random color that has not been used by any of its neighbors that are closer than a threshold edge

```
sort c-graph nodes by degree
set edge threshold
do {
  for each node (in decreasing order of degree) {
    N = set of neighbors of interest (meeting threshold)
    assign random color that neighbors in N do not have
    if no colors available, break out of loop
  }
  if unsuccessful {
    for each node (in decreasing order of degree) {
      N = set of neighbors of interest (meeting threshold)
      assign lowest color that neighbors in N do not have
      if not possible, break out of loop
    }
  }
  if unsuccessful
    reduce threshold
} while unsuccessful
```

FIGURE 10 Graph coloring procedure.

weight. (If there is a problem assigning a random color, the procedure next tries assigning the lowest numbered color that is not used by any neighbors.) This ensures that the neighbors that are closer than a specified threshold will be assigned different colors. If the number of reset lines (colors) allowed does not permit the current graph to be colored, the threshold is reduced and coloring is attempted once more. Effectively, this reduces the number of neighbors that the coloring procedure has to deal with when attempting to assign different colors to the neighbors. The random color selection in the coloring procedure is used to allow the FFs to be partitioned in a somewhat uniform manner. Otherwise, the lowest numbered reset line would get more of the FFs assigned to it. The randomness appears to provide better results.

The overall partitioning method is summarized in Figure 11.

RESULTS

These algorithms were implemented into a C program and we ran them on the ISCAS '89 sequential benchmarks. They process all of the circuits within a few minutes.

```
create c-graph
find distances between FFs
adjust weights according to fanouts
perturb weights randomly
color c-graph
```

FIGURE 11 Summary of Partitioning Method.

The test generator we used in our experiments is CRIS, a new hierarchical (including switch-level) pseudo-random STG that uses genetic algorithms [27]. CRIS has been developed on the Sun 4 platform at the University of Illinois. CRIS generates test vectors quicker than traditional deterministic test generators at the price of slightly lower fault coverage.

Table II describes the benchmark circuits that were used to obtain the results. The "Selected flip-flops" column indicates the number of FFs that were made resettable.

Fault Coverage with Single Reset Line

As a basis for comparison, test vectors for the circuits of Table II without any DFT techniques were generated. The number of faults injected and fault coverages are shown in Table III(a).

Table III(b) contains the number of faults injected and fault coverage obtained using FR. This will assist in separating the initialization effects of PR from the controllability of PR. Recall that FR can only be used for initialization and not within a test sequence.

The selected FFs above were transformed into scan FFs and CRIS was used to generate test vectors for these circuits with partial scan. On the average, about 44% of the FFs in each of the circuits were selected to be made resettable. The fault coverages obtained are displayed in Table III(c). Since no faults were injected on the scan circuitry or scan path, the

TABLE II
Circuit Description

Circuit	Number of gates	Number of flip-flops	Selected flip-flops	Number of inputs	Number of outputs
s298	119	14	1	3	6
s344	160	15	5	9	11
s349	161	15	5	9	11
s382	158	21	9	3	6
s386	159	6	5	7	7
s400	162	21	10	3	6
s510	211	6	5	19	7
s641	379	19	7	35	24
s713	393	19	7	35	23
s820	289	5	4	18	19
s832	287	5	4	18	19
s953	395	29	5	16	23
s1488	653	6	5	8	19
s1494	647	6	5	8	19
s35932	16065	1728	297	35	320

number of faults is the same as for the circuit of Table III(a). Note that the FFs scanned are the same ones that were made resettable to allow a fair comparison.

The results using a single reset line is presented in Table IV. The number of faults injected has increased slightly due to the additional faults of the resetting circuitry. The column labeled "% over no DFT" shows the percentage that the fault coverage increases by adding PR to the original circuit. The percentage that the fault coverage increases for PR circuits over FR is displayed in the column labeled "% over FR." The last column labeled "% under p. scan" shows the percentage that PR's fault coverage is below partial scan.

Most of these circuits yield results fairly close to partial scan. Seven of the circuits with PR produced fault coverages that are within 2.1% of the ones with PR. One circuit (s386) did not seem to improve the coverage much over the circuit with no DFT techniques. In fact, due to the additional faults injected, the fault coverage becomes slightly lower.

Initializable circuits with PR, where the resettable FFs are a large percentage of the total FFs, produce fault coverages further below that of partial scan than other initializable circuits with PR. This is a result of the interference referred to earlier. PR may still help when compared to circuits with no DFT techniques. This indicates that better heuristics are needed to select resettable FFs since the present method seems to help more with partial scan than with PR. We suspect that better results can be obtained by cleverly selecting a subset of these FFs. One possible method would be similar to the technique presented in [28] for identifying FFs that cannot be initialized due to faults that prevent initialization. Another possibility is to combine cycle-breaking with a testability measure, such as SCOAP [24].

TABLE III
Test Generation Results with (a) No DFT Techniques (b) FR and (c) Partial Scan

(a) ckt	Faults inj.	Fault cov.	(b) ckt	Faults inj.	Fault cov.	(c) ckt	Fault cov.
s298	308	86.7	s298	338	86.3	s298	94.0
s344	342	94.4	s344	374	94.7	s344	98.8
s349	350	94.0	s349	382	94.5	s349	98.3
s382	399	77.9	s382	443	78.1	s382	99.0
s386	384	78.9	s386	398	78.1	s386	97.5
s400	421	76.0	s400	465	74.2	s400	97.6
s510	564	0.0	s510	489	84.6	s510	100.0
s641	467	86.5	s641	507	84.4	s641	94.6
s713	581	81.9	s713	621	82.0	s713	88.5
s820	850	48.1	s820	862	46.5	s820	99.5
s832	870	46.7	s832	882	45.5	s832	96.2
s953	1079	9.5	s953	1139	9.7	s953	99.4
s1488	1486	95.3	s1488	1500	93.3	s1488	99.9
s1494	1506	93.3	s1494	1520	90.1	s1494	99.1
s35932	39690	89.3	s35932	42552	79.11	s35932	89.5

TABLE IV
Test Generation Results with PR

ckt	Faults inj.	Fault cov.	% over no DFT	% over FR	% under p. scan
s298	310	91.9	5.2	5.5	2.1
s344	354	98.3	3.9	2.3	0.5
s349	362	98.3	4.3	1.7	0.0
s382	419	90.5	12.6	12.2	8.5
s386	396	77.5	−1.4	−0.6	20.0
s400	441	86.2	10.2	4.2	11.4
s510	576	83.3	83.3	−8.4	16.7
s641	483	93.4	6.9	8.4	1.2
s713	597	86.8	4.9	4.2	1.7
s820	860	52.0	3.9	5.5	47.5
s832	880	52.6	5.9	7.1	43.6
s953	1091	98.7	89.2	19.0	0.7
s1488	1498	96.1	0.8	2.8	3.8
s1494	1518	94.9	1.6	4.8	4.2
s35932	39690	89.5	0.2	9.8	0.0

In circuits that are difficult to initialize, the number of FFs that are made resettable has a big impact. For example, in s510, all the cycles are broken when five of the FFs are selected. If all of the FFs are selected, the fault coverage is further increased, as evidenced by the fault coverage of FR. In general, however, FR produces lower fault coverage than PR. It appears that PR and FR benefit uninitializable circuits the greatest. PR can even aid circuits that already have a FR line since the PR line can be used like a regular PI for extended controllability.

Some circuits were not impacted much by the reset value selection. When the reset values were changed, the fault coverage did not vary significantly. Just the fact that the FF was made resettable increased the testability the most. On the other hand, there were some circuits where the reset value had a big influence. For example, s510 yielded a fault coverage of about 10% when all of the selected FFs were reset to 0. In contrast, the fault coverage rose to over 80% when all of the reset values were changed to 1.

The fault coverage is as expected for most circuits. In general, partial scan is better than PR, which is better than FR, which in turn is better than the circuit with no DFT. Exceptions, on the other hand, do exist. CRIS sometimes produces better results with circuits without DFT when compared to FR. Part of the reason is that CRIS is not fault-oriented. The reset line will be preferred by the STG since it usually has the greatest effect on the circuit's state when compared to other PIs. Typical STG's do not have enough knowledge to determine when the reset line usage would become detrimental to the generated test sequence. The search may be affected adversely due to this and, therefore, produce anomalous results. Note that this behavior may occur in deterministic test generators as well. For example, a circuit with FR may allow the STG to generate a test sequence that is simpler and shorter than a circuit without DFT for a fault f_1. The original circuit may cause the STG to produce a longer test sequence to detect f_1. However, a side effect of this may be that a hard-to-detect fault, f_2, might be detected with the longer sequence. Therefore, the shorter, simpler sequence may miss the faults that are "accidentally" detected by the longer test sequence [29].

TAT Comparison

We now evaluate the TATs for PR, partial scan and full scan. The test lengths for similar coverages are tabulated in Table V. We used the fault coverage of the original circuit (no DFT) as the target fault coverage. (For s510, we used the fault coverage of PR (83.8%), since the original circuit is untestable by 3-value STGs.) Since CRIS is a pseudo-random test generator, the test vector lengths are typically longer than ones obtained from a deterministic fault-oriented test generator. We have calculated the number of clock cycles required to test the circuits. The temporal cost of testing scan design circuits is calculated as follows:

$$t = v + vs + (s - 1), s > 0 \quad \text{and} \quad t > 0$$

where t is the total number of clock cycles required for testing, v is the number of test vectors and s is the number of scanned FFs. The first term v is the number of regular clocks. The second term indicates the number of times that new vectors are scanned into the circuit while scanning out the results from

TABLE V
Test Application Times Comparison

ckt	PR # cyc	p. scan # cyc	p. scan % bel	f. scan # cyc.	f. scan % bel
s298	335	2132	84.29	823	59.30
s344	249	1678	85.16	814	69.41
s349	264	1624	83.74	814	67.57
s382	542	3438	84.23	944	42.58
s386	1725	11350	84.80	453	−280.79
s400	794	3342	76.24	878	9.57
s510	125	4282	97.08	313	60.06
s641	232	2606	91.10	1738	86.65
s713	433	2390	81.88	1698	74.50
s820	1531	1118	−36.94	358	−327.65
s832	1766	983	−79.65	454	−288.98
s953	37	58	36.21	58	36.21
s1488	3160	13907	77.28	1447	−118.38
s1494	4368	11950	63.45	1503	−190.62
s35932	2712	2.009E6	99.86	4.981E6	99.94

the previous time. The last term shows the clock cycles needed to shift out the results of the last test vector. The table shows how much smaller TATs are for PR with respect to the scanned circuits in the "% bel" column. Cases where the scanned circuits had lower TATs are identified with negative "% bel" values.

The results indicate that PR produce much shorter TATs than partial scan, on the average. Only two partial scan circuits produced shorter TATs. Full scan, on the other hand, performed better than partial scan for all except the biggest circuit as far as TATs are concerned. Despite the improved outcome, full scan did better than PR on only 5 of 15 circuits.

In some cases, PR yielded a smaller number of test vectors for the same fault coverage when compared to partial scan. Theoretically, this should not be the case. There are two reasons that these conditions occur. The first is, once again, due to the fact that CRIS is a random fault-independent test generator. The second is that the different search spaces that PR and partial scan have, leads to the existence of search anomalies. A PR circuit has a smaller search space with respect to the PIs. Since it is restricted, it might find a shorter sequence depending on its path in the search space. An STG processing a partial scan circuit may begin to search a part of the search space which would only produce longer test sequences.

We compare our results with those of other DFT techniques used for TAT reduction. PR shortened TATs by 60.78% compared to full scan circuits where there was an improvement. Pradhan and Saxena's method had an average of 29.07% improvement over full scan in the 12 ISCAS '89 circuits for which they reported results. The scheme of Gupta et al reduced test application times of lumped ISCAS '85 combinational circuits by an average of 12.33%. For partial scan circuits where there was an improvement, PR reduces the TAT by 80.9%. This translates to an average of 9.36 times speedup than the partial scan versions of these circuits (not including s35932 circuit which produced a highly divergent speedup of 737). Lee and Shin's method obtained TAT speedups of 11.25 and 8 over that for partial scan for the two circuits which they evaluated (assuming a limited number of PIs and POs). Our results are encouraging, especially since TAT reduction was only a secondary goal.

Fault Coverage with Multiple Reset Lines

The fault coverages obtained for two reset lines are now compared with the ones for circuits with a single reset line. Table VI contains data obtained by using the simple technique of allowing the FFs to be controlled to both 0 and 1. The number of faults injected, the fault coverage and the amount the fault coverage increased over the single reset technique is shown in the table. This technique did not prove beneficial for many circuits. For many of the circuits, the coverage actually appeared to drop below that obtained by using the single reset method. This may be partly due to the heuristics used within CRIS. Only the circuits which had an increase in coverage over single reset are shown.

TABLE VI
Coverage Improvement Using Selectable Reset Value

ckt	Faults inj.	% Fault cov.	% Over 1 reset
s349	384	99.0	0.6
s400	479	86.6	0.4
s713	627	88.7	1.9
s820	878	59.0	7.0
s832	898	54.6	2.0
s1488	1520	97.1	1.0

Table VII contains results for the scheme of dividing the selected FFs into two partitions. The results were much better using the partitioning scheme. Most of the circuits had an increase in fault coverage when compared to the single reset line method. We have noticed that using multiple independent reset lines facilitates detection of faults, even on the reset lines. This is a benefit that single reset cannot guarantee. By giving independence to selected FFs as in the partitioning technique, the fault coverage increases. Multiple reset lines were more helpful in circuits that had lower fault coverage when single reset was used.

TABLE VII
Coverage Improvement Using Partitioning

ckt	Faults inj.	% Fault cov.	% Over 1 reset
s298	309	92.6	0.7
s344	354	99.4	1.1
s349	362	98.9	0.6
s382	421	93.3	2.8
s386	396	87.1	9.7
s400	443	91.0	4.8
s510	576	99.0	7.6
s641	485	94.8	1.4
s713	599	88.8	2.0
s820	862	70.2	18.2
s832	882	68.7	16.1
s953	1091	98.7	0.0
s1488	1498	98.0	1.9
s1494	1518	97.3	2.4
s35932	39692	90.0	0.5

CONCLUSIONS

We have presented a methodology of DFT called partial reset (PR) where a subset of the FFs are made resettable. As a result, PR can be applied *within* a test vector sequence in conjunction to normal inputs which provides an additional degree of controllability. The end result is the boosting of the circuit's fault coverage and the capability to perform at-speed testing. PR reduces the penalties of hardware overhead, long TAT and slower clock speed. A partitioning scheme with multiple reset lines was described, which produced even higher fault coverage. When benchmarked against partial scan, the results were very encouraging. This DFT approach produced good results for both fault-independent STG (like CRIS) as well as fault-targeted STGs.

In the future, we plan to examine the effects that three reset lines have on these circuits. It is conjectured that the law of diminishing returns will take effect as the number of reset lines are increased. The method of FF selection plays a major role in obtaining good fault coverage. If a large subset of the FFs are chosen, they begin to interfere with the operation of the circuit and becomes more like full reset. Multiple reset lines alleviate this problem to a degree. We are currently investigating different FF selection algorithms which rank a set of FFs based on the function of the circuit and not just the structure.

Acknowledgment

We would like to thank D. H. Lee and S. M. Reddy for making their flip-flop selection program available to us.

References

[1] H. Fujiwara and S. Toida, "The complexity of fault detection: An approach to design for testability," in *Proc. 12th Int. Symp. Fault-Tolerant Compt.*, pp. 101–108, June 1982.

[2] T.W. Williams and K.P. Parker, "Design for testability—a survey," *Proc. IEEE,* vol. 71, pp. 98–112, Jan. 1993.

[3] M. Abramovici, M.A. Breuer, and A.D. Friedman, *Digital Systems Testing and Testable Design*. Computer Science Press, Rockville, Md, 1990.

[4] J.P. Hayes, "On modifying logic networks to improve their diagnosability," *IEEE Trans. Computers,* vol. C-23, pp. 56–62, 1974.

[5] I. Pomeranz and S.M. Reddy, "Test generation for synchronous sequential circuits using multiple observation times," in *Proc. Int. Symp. Fault-Tolerant Compt.*, pp. 52–59, 1991.

[6] I. Pomeranz and S.M. Reddy, "Classification of faults in synchronous sequential circuits," Tech. Rep. No. 3-27-1991, Electrical and Comput. Eng. Dept. U. of Iowa, 1991. To appear in IEEE Trans. Comput.

[7] E. Trischler, "Incomplete scan path with an automatic test generator," in *Proc. Int. Test Conf.*, pp. 153–162, 1980.

[8] H.K. Ma, S. Devadas, A.R. Newton, and A. Sangiovanni-Vincentelli, "An incomplete scan design approach to test generation for sequential machines," in *Proc. Int. Test Conf.*, pp. 730–734, 1988.

[9] K.T. Cheng and V.D. Agrawal, "A partial scan method for sequential circuits with feedback," *IEEE Trans. Computers,* vol. 39, pp. 544–548, Apr. 1990.

[10] D.H. Lee and S.M. Reddy, "On determining scan flip-flops in partial-scan designs," in *Proc. Int. Conf. Computer-Aided Design,* pp. 322–325, 1990.

[11] M. Abramovici, J.J. Kulikowski, and R.K. Roy, "The best flip-flops to scan," in *Proc. Int. Test Conf.*, pp. 166–173, 1991.

[12] V. Chickermane and J.H. Patel, "A fault oriented partial scan design approach," in *Proc. Int. Conf. Computer-Aided Design,* pp. 400–403, 1991.

[13] Y.K. Malaiya and R. Narayanaswamy, "Modeling and testing for timing faults in synchronous sequential circuits," *IEEE Design & Test of Computers,* pp. 62–74, Nov. 1984.

[14] S. Lee and K.G. Shin, "Design for test using partial parallel scan," *IEEE Trans. Computers,* vol. 9, pp. 203–211, Feb. 1990.

[15] M.J.Y. Williams and J.B. Angell, "Enhancing testability of large-scale integrated circuits via test points and additional logic," *IEEE Trans. Computers,* vol. C-22, pp. 46–60, Jan. 1973.

[16] D.K. Pradhan and J. Saxena, "A design for testability scheme to reduce test application time in full scan," in *Digest of Papers IEEE VLSI Test Symp.*, pp. 55–60, 1992.

[17] R. Gupta, R. Srinivasan, and M.A. Breuer, "Reorganizing circuits to aid testability," *IEEE Design & Test of Computers,* pp. 49–57, Sept. 1991.

[18] J.-F. Wang, T.-Y. Kuo, P.-C. Chen, and J.-Y. Lee, "Test generation for presettable synchronous sequential circuits," in *Int. Symp. on VLSI Technology, Systems and Applications,* pp. 155–158, 1989.

[19] K.T. Cheng and V.D. Agrawal, "Initializability consideration in sequential machine synthesis," *IEEE Trans. Computers,* vol. 41, pp. 374–379, Mar. 1992.

[20] I. Pomeranz and S.M. Reddy, "On the role of hardware reset in synchronous sequential circuit test generation." to appear in IEEE Trans. Computers, also available as Technical Report No. 10-1-1991, Electrical and Computer Engineering Department, University of Iowa, October 1991.

[21] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *Proc. Int. Symp. Circuits Syst.*, pp. 1929–1934, 1989.

[22] MOSIS Service, Marina del Rey, CA, *CMOSN Cell Notebook: Scalable 2.0 and 1.2 Micron CMOS/Bulk Cell Family,* release 3.0a ed., Oct. 1991.

[23] K.T. Cheng and J.Y. Jou, "Functional test generation for finite state machines," in *Proc. Int. Test Conf.*, pp. 162–168, 1990.

[24] L.H. Goldstein, "Controllability/observability analysis of digital circuits," *IEEE Trans. Computers,* vol. CAS-26, pp. 685–693, Sept. 1979.

[25] S. Devadas, H.-K.T. Ma, A.R. Newton, and A. Sangiovanni-Vincentelli, "Irredundant sequential machines via optimal logic synthesis," *IEEE Trans. Computer-Aided Design,* vol. 9, pp. 8–18, Jan. 1990.

[26] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.

[27] D.G. Saab, Y.G. Saab, and J.A. Abraham, "CRIS: A test cultivation program for sequential VLSI circuits," in *Proc. Int. Conf. Computer-Aided Design,* pp. 216–219, 1992.

[28] M. Abramovici and P.S. Parikh, "Warning: 100% fault coverage may be misleading!!," in *Proc. Int. Test Conf.*, pp. 662–668, 1992.

[29] F. Maamari, Feb. 1993. Private communication.

Biographies

BEN MATHEW received his B.S. in Electrical Engineering from Texas A & M University in 1988 and his M.S. in Electrical and Computer Engineering at the University of Illinois in 1991. From 1989 to 1990, he was a teaching assistant at the Electrical and Computer Engineering Department. Since then, he has been a research assistant in the Coordinated Science Lab at the University of Illinois where he is currently a Ph.D. candidate. Mr. Mathew's research interests include testing and CAD for digital VLSI.

DANIEL G. SAAB has received his B.S. Degree in Computer Engineering, his M.S. Degree in Electrical Engineering, and his Ph.D. in Electrical Engineering from the University of Illinois at Urbana-Champaign in 1982, 1985, and 1988, respectively. From July 1982 to August 1983 he worked as a Computer Aided Design Engineer at Tektronix in Oregon. Currently he is an assistant professor in the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign. He received the 1989 Semiconductor Research Corporation Inventive contribution award. Prof. Saab's research interests include testing and fault simulation, circuit, timing and switch-level simulation of VLSI circuits, parallel processing, and design automation.