Single-Clock Partial Scan

KWANG-TING (TIM) CHENG
University of California,
Santa Barbara

Existing partial-scan designs use a separate scan clock to simplify scan flip-flop selection and test generation methods. Such designs require multiple clock trees and create clock-signal routing problems that, in general, require tight control of clock skew. The author examines using the system clock for the scan operation and includes experimental results based on ISCAS89 benchmark circuits.

SEQUENTIAL CIRCUIT TEST generation is emerging as one of the new frontiers in CAD. Recent advances in this area have reduced the need for high-overhead design-for-testability techniques, such as full scan. These advances have resulted in several commercial sequential test generators. Such tools work together with partial scan to successfully generate high-coverage test sequences for certain large designs.

Partial scan usually relies on a scan clock that is separate from the system clock. Two-clock partial scan offers several advantages (see the box), but causes a difficult routing problem: routing two separate, small-skew clock trees. Also, the need to apply a scan sequence before *each* vector produced by the test generator creates two other disadvantages of using a separate scan clock: 1) it requires a longer test sequence, and 2) the vectors are not applied at speed, from the viewpoint of the circuit's combinational portion.

We want to avoid the routing problem by using the system clock for the scan operation (Figure 1), yet retain the advantages of the two-clock technique. We must therefore solve the problems single-clock partial scan causes in test generation and scan flip-flop selection, as well as optimize scan flip-flop ordering to recoup the fault coverage lost as a result of this technique.

Basic problem

When we apply a (system) clock pulse to a single-clock partial-scan circuit, both scan and nonscan flip-flops update their states. The next states of the scan flip-flops can be derived two ways. 1) With the circuit in test mode, the next state is simply a single bit shift of the scan register (scan shifting).[1] 2) In functional mode, functional logic produces the next state (functional justification). Even if we employ both methods, the flip-flops can only reach a limited number of next states from their current states. Some faults that are testable when we treat scan flip-flops as fully controllable and observable are now untestable.

Consider the circuit in Figure 2 that has four flip-flops, A, B, C, and D. The primary outputs are just the outputs of flip-flops. Suppose flip-flops A and B are scan flip-flops and C and D are nonscan flip-flops. Let the scan register's order be $S_{in} \rightarrow B \rightarrow A \rightarrow S_{out}$, where S_{in} and S_{out} are scan input and output signals. Activation of target fault C stuck-at-0 needs all three inputs of the AND gate feeding C to be 1. This requires A and D to be 1 during the previous clock cycle. It can be shown that neither scan shifting nor functional justification

0740-7475/95/$04.00 © 1995 IEEE

Two-clock partial scan

A separate scan clock allows us to freeze nonscan flip-flops during scan and substantially increases the controllability and observability of scan flip-flops in a partial-scan circuit. By assuming that scan flip-flops are completely controllable and observable at every clock cycle, we can use standard test generators. Two-clock partial scan treats the scan flip-flop outputs and inputs as primary inputs and outputs for test generation. After generating the test sequence, we add scan sequences before and after each vector to initialize the scan flip-flops to the desired states.

Partial scan circuits typically generate the separate scan clock internally by gating the scan-enable signal with the system clock (Figure A), rather than having a clock provided externally.

In addition, recently proposed partial-scan techniques[1,2] based on the two-clock model automatically select and scan memory elements that cause problems for test generators. The scan operation breaks most cycles, creating a feedback-free circuit. This is a distinct benefit because the computational complexity of test generation for a feedback-free sequential circuit is low, similar to that of a combinational circuit.[1,3]

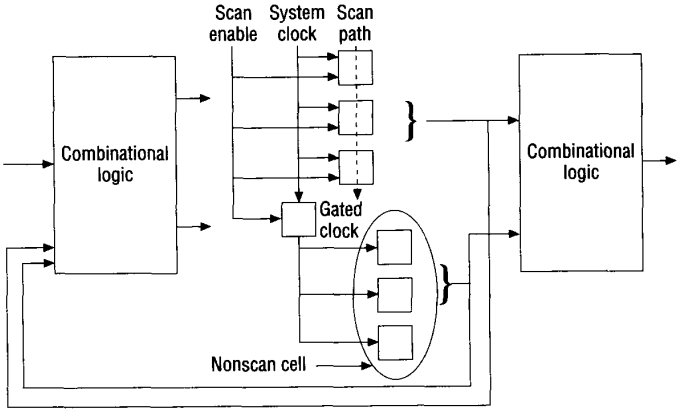

Figure A. *Two-clock partial scan using a gated clock.*

References

1. K.-T. Cheng and V.D. Agrawal, "A Partial Scan Method for Sequential Circuits with Feedback," *IEEE Trans. Computers*, Vol. 39, No. 4, Apr. 1990, pp. 544-548.
2. R. Gupta, R. Gupta, and M.A. Breuer, "The Ballast Methodology for Structured Partial Scan Design," *IEEE Trans. Computers*, Vol. 39, No. 4, Apr. 1990, pp. 538-544.
3. A. Miczo, *Digital Logic Testing and Simulation*, Harper & Row, New York, 1986.

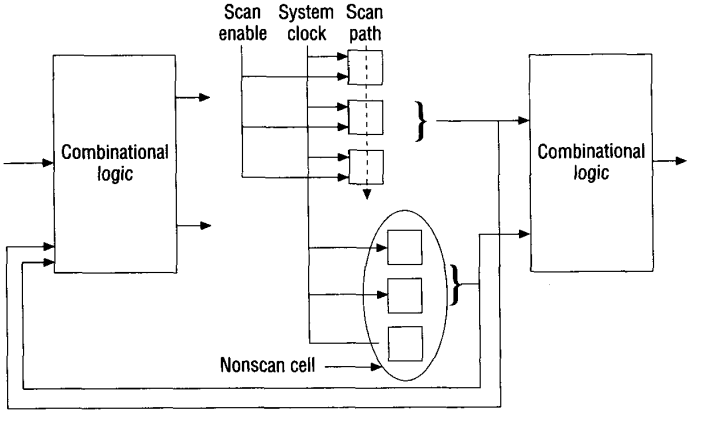

Figure 1. *Partial scan using the system clock for the scan operation.*

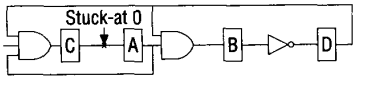

Figure 2. *Example circuit with a fault that becomes untestable for single-clock partial scan.*

can justify this state (A = 1, D = 1).

For example, in scan shifting, B and A should each be 1 because B precedes A in the scan chain. However, for D to be 1, B should be zero. Therefore, for a single-clock design with scan ordering $S_{in} \rightarrow B \rightarrow A \rightarrow S_{out}$, the fault is untestable. However, this fault is testable if we treat A and B as primary inputs, as for a two-clock design. This example illustrates why fault coverage for a single-clock partial-scan circuit may be lower than that of a two-clock circuit. Scan shifting can also generate tests for delay faults in full-scan circuits.[1,2]

Theoretically, we can mix scan shift-

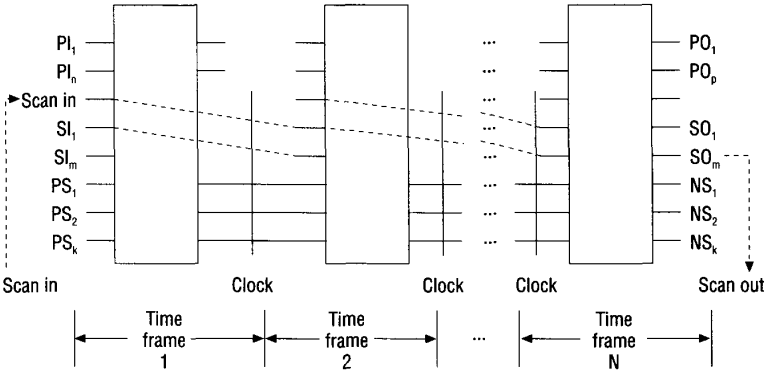

Figure 3. *Model of test generation for partial-scan circuits without a separate scan clock.*

ing and functional justification in a test sequence; some time frames can use scan shifting and others, functional justification. However, using both methods for all time frames would make test generation for large designs too complex. Functional justification is much more expensive than scan shifting in large designs because it justifies the states of both scan and nonscan flip-flops (backwards) through complex functional logic. Our experimental implementation, therefore, only incorporates scan shifting.

Our modified test generation method inserts scan sequences before the first vector and after the last vector in each test sequence generated for a target fault. As our experimental results prove, eliminating scan sequences between test vectors substantially reduces the total length of the test set. Also, we can apply each subsequence at speed (because there are no scan vectors in between) and thus uncover delay faults.

Test generation

This study modified the Back algorithm,[3] which employs time frame-expansion and reverse time processing (RTP) techniques[4] to generate tests for single-clock partial-scan circuits. It assigns a D or $\bar{D}$ to a primary output, and justification proceeds backwards in the time frame model. Testability measures, like controllability and drivability, guide the backward search. The backward justification process requires some modification to account for not freezing the nonscan flip-flop states during scan operation. As mentioned earlier, This investigation assumes that only scan shifting (that is, using test mode) produces the next states of the scan flip-flops.

Figure 3 shows a test generation model under this assumption. There are n primary inputs (PI_1, ..., PI_n), p primary outputs (PO_1, ..., PO_p), m scan flip-flops, and k nonscan flip-flops. The data outputs and inputs of the scan flip-flops are pseudo inputs (SI_1, ..., SI_m) and outputs (SO_1, ..., SO_m). The data outputs and inputs of the nonscan flip-flops are the present state (PS_1, ..., PS_k) and next state (NS_1, ..., NS_k) lines. Test generation starts from the last time frame to time frame N. It is assumed that all primary outputs and pseudo outputs are fully observable and next-state lines are not (directly) observable. The dashed lines indicate single bit-shift operations. In backward justification, if the algorithm assigns a value to SI_i in time frame x, the value needs further justification. The algorithm will assign the same value to SI_{i-1} in time frame $x-1$ (as shown by the dashed lines), assuming scan flip-flop $i-1$ precedes scan flip-flop i in the scan register. Test generation is completed when the present state lines specify no required state.

A test sequence obtained in such a manner assures that 1) a single bit shift of the states at time frame $i-1$ produces scan flip-flop states at time frame i, and 2) functional logic produces the nonscan flip-flop states. The completion criterion for the automatic test pattern generation process (no state requirement at PSs) assures that the test sequence will be valid for any power-up state.

This model generates a test sequence that has a first vector with two parts, a primary input vector and initial state. The rest of the sequence contains only primary input vectors. A scan sequence added at the beginning of the test sequence will place the circuit in its required initial state. The remaining vectors do not need the additional scan sequences because a single bit shift of the scan chain produces the desired states.

Usually, we perform fault simulation after successfully generating a test sequence. Fault simulation also requires minor modifications. Suppose the newly generated test sequence has N vectors. In fault simulation, we must consider the pseudo outputs (SOs) unobservable at time frames 1 to $N-1$. At those time frames, the circuit remains in test mode and fault effects cannot be latched into the scan flip-flops, even if they appear at the flip-flops' data inputs. Therefore, we cannot consider faults that produce fault effects at SOs in time frames 1 to $N-1$ to be detected at the SOs.

Using such a test generation scheme, however, has several other advantages. First, the scheme implicitly compacts the test sequences. The existing test generation scheme treats the pseudo inputs as primary inputs. Additional constraints imposed at pseudo inputs (shown as dashed lines in Figure 3) eliminate the

need for scan sequences between time frames. Benchmark circuit results show that with the extra constraints on pseudo inputs most faults remain testable, and we can usually generate a test using about the same number of time frames. Thus, eliminating scan sequences between time frames significantly reduces total test length.

Secondly, because we don't apply any scan sequences between test vectors, the vectors are actually applied at speed and thus such vectors can detect delay faults. Incorporating other techniques[5] can automatically generate tests for transition faults in partial-scan circuits.

Selecting scan flip-flops

Cycle-breaking remains an effective heuristic for flip-flop selection in single-clock partial-scan designs. To justify this contention, this study analyzed the sequential graphs (S-graph) for partial-scan circuits. The S-graph of a synchronous sequential circuit is directed graph G={V,E}, where vertex v_i represents flip-flop i. A directed edge from v_i to v_j implies a combinational path between flip-flops i and j. Explicitly representing only memory elements and dependencies between memory elements, this graph shows us the signal flow between flip-flops.

We can construct the S-graph for single-clock partial-scan circuit G_{PS_1c} from the S-graph for nonscan circuit G in two steps. (Here we assume that only scan shifting produces the scan flip-flop next states.) First, if we select flip-flop i as a scan element, we remove the incoming edges of vertex i in G. Removal of these edges reflects the fact that, for scan shifting, the values in the scan flip-flops are not from functional logic. Second, if flip-flop j precedes i, we add an edge to G_{PS_1c} from vertex j to i. The added edge represents the signal flow of the scan register's single bit shift.

Figure 4 illustrates the changes from G to G_{PS_1c}, where A and B are scan flip-flops and A precedes B in the scan chain. Different orderings of the scan register result in different versions of G_{PS_1c}.

This study formulates the flip-flop selection problem as follows. Given G, select a set of vertices and a scan-register ordering such that, after the modification just described, G_{PS_1c} is acyclic. First, we can solve this problem without accounting for the ordering. That is, if vertex set V and scan ordering R result in an acyclic S-graph, then vertex set V with any ordering R′ will result in an acyclic S-graph, too. We can easily prove this statement by showing that the added edges (for scan shifting) will never be part of any cycle. Therefore, given a set of vertices for scan, regardless of how we add these edges (that is, regardless of scan ordering), we will not create or remove any cycle. Based on this observation, we can simplify the flip-flop selection problem as follows: Given G, select a set of vertices such that after removing all incoming edges to these vertices, resulting S-graph G_{PS_1c} is acyclic.

The vertex selection problem is not a standard graph theory problem. However, a solution to a known graph problem, the feedback vertex set problem, is also a solution to our problem. Given a graph G(V,E), the feedback vertex set problem is to select a set of vertices V′ such that the removal of V′ and its associated incoming I(V′) and outgoing O(V′) edges, results in an acyclic graph. We can express this as G′ = [V–V′,E–I(V′)–O(V′)]. We can also easily show that if G′ is acyclic, then G″ = [V,E–I(V′)] is acyclic, too.

Ordering scan flip-flops

During test mode in scan shifting, the system obtains a scan flip-flop's state from its predecessor in the scan chain. Therefore, ordering flip-flops differently in the scan register may result in different sets of reachable states and, thus, different fault coverages. This study's ordering maximizes fault coverage. Scan flip-flop ordering may also affect other cost functions, such as routing area. This discussion does not consider other cost functions, although our formulation could easily incorporate them as constraints.

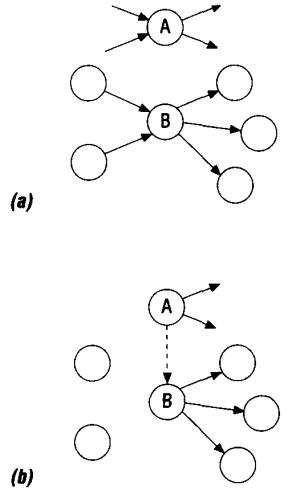

Figure 4. *S-graphs for a nonscan sequential circuit* ***(a)*** *and a single-clock partial-scan circuit* ***(b)****, in which A and B are scan flip-flops, and A precedes B in the scan chain.*

Consider Figure 2's example again. To break the global feedback, we choose A and B as scan flip-flops, giving us two possible scan orderings: $S_{in}\rightarrow A\rightarrow B\rightarrow S_{out}$ or $S_{in}\rightarrow B\rightarrow A\rightarrow S_{out}$. As shown earlier, fault C stuck-at-0 is untestable with the scan ordering $S_{in}\rightarrow B\rightarrow A\rightarrow S_{out}$. However, with the other scan ordering, the fault becomes testable. This study examined the induced S-graph for $S_{in}\rightarrow B\rightarrow A\rightarrow S_{out}$ (shown in Figure 5, next page, where edge B→A is added for the given scan ordering) that makes C stuck-at-0 untestable. This scan ordering causes two equal-length paths of length 2 between flip-flops B and C. Justifying a value at C backwards through these two paths leads to a conflict at B's output. In general, equal-length paths between any two vertices in an S-graph may lead to

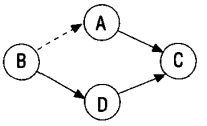

Figure 5. S-graph for the circuit in Figure 2 with scan order B→A.

conflicts that make faults untestable. On the other hand, the ordering S_{in}→A→B→S_{out} has no equal-length paths between any two vertices in the induced S-graph. Based on this information, we can specify the following heuristic to optimize fault coverage.

Heuristic 1. Find a scan register ordering such that the induced S-graph has the minimum number of equal-length paths.

Not all equal-length, reconvergent paths would actually cause a conflict in backward justification. Even if a conflict occurs, a fault may remain testable if we can justify the required values for fault detection through alternative fan-in routes. Equal-length paths are a problem when 1) they indeed lead to a conflict (in backward justification), and 2) there are no alternate fan-in routes from the paths.

However, checking conditions 1 and 2 for an equal-length path pair is similar to performing test generation for a fault. It is too computationally expensive to check the impact of all equal-length paths. Heuristic 1 simply tries to minimize the number of equal-length, reconvergent paths. It makes an implicit and conservative assumption that all such paths cause a conflict and therefore, should be avoided, if possible, through scan flip-flops ordering.

Exploring paths of all lengths between two vertices in a directed graph is computationally expensive. However, for long, equal-length paths, conflicting requirements are less likely to occur because other fan-ins along the path exist that could resolve the conflict. This leads to a second heuristic.

Heuristic 2. It is sufficient to consider only equal-length, short paths between any two vertices in the S-graph to predict conflicts.

These two heuristics help determine an efficient ordering of scan flip-flops maximizing fault coverage as follows. Take as given a sequential graph G(V,E) and a set of k scan flip-flops S ⊆ V. Find a total ordering of the set S, $s_1 > s_2 > s_3 ... > s_k$ (where s_1 is the first element in the scan register and s_k the last), that minimizes the following cost function for the induced directed graph G[V,E–I(S)+J],

$$C = \sum_{\forall[(I,J)\in V]} \frac{P(i,j)-1}{L(i,j)}$$

I(S) is the set of incoming edges to set S and J ≡[(s_i,s_{i+1}) | i = 1, ..., k–1], which reflects the data flow associated with the scan register. $P(i,j)$ is the number of distinct shortest paths between i and j, and $L(i,j)$ is the length of the shortest path between i and j.

In other words, we seek a scan ordering that minimizes the cost of the sequential graph it induces. In the cost

Table 1. Fault coverage and test efficiency for two-clock and single-clock circuits.

Circuit	No. flip-flops	No. nonscan flip-flops	Fault coverage (%) Two clock	Fault coverage (%) Single clock*	Fault coverage (%) Difference	Test efficiency (%) Two clock	Test efficiency (%) Single clock*	CPU time (s) Two clock	CPU time (s) Single clock*
s344	15	7	97.1	97.1	0.0	98.8	98.8	21	21
s382	21	12	99.5	98.5	1.0	100.0	100.0	2	3
s400	21	12	98.1	96.7	1.4	100.0	100.0	3	3
s444	21	12	96.0	91.8	4.2	98.1	95.8	99	236
s641	19	13	94.4	94.4	0.0	100.0	100.0	5	5
s713	19	12	88.5	88.5	0.0	100.0	100.0	19	18
s953	29	24	100.0	97.2	2.8	100.0	100.0	13	63
s1423	74	52	80.3	78.9	1.4	81.2	79.7	5,340	5,520
s5378	149	119	93.7	92.7	1.0	99.4	98.8	720	1,080
s9234.1	211	99	93.3	90.5	2.8	99.7	99.3	1,620	2,820
s13207.1	638	334	97.5	97.4	0.1	99.2	99.1	2,880	3,300
s15850.1	534	344	93.9	92.7	1.2	96.9	95.8	7,440	9,240
s35932	1,728	1,422	89.8	89.8	0.0	99.9	99.9	12,780	13,680

**Uses scan shifting only. Scan register ordering follows the order of the flip-flops in the netlist file (default order) and could be considered a random order.*

Table 2. Test lengths for single- and two-clock circuits (fault coverages not necessarily equal).

Circuit	No. generated vectors		Single-chain sequence Test length (clock ticks)			Five-chain sequence Test length (clock ticks)		
	Two clock	Single clock (subsequences)	Two clock	Single clock*	Ratio (two÷single)	Two clock	Single clock*	Ratio (two÷single)
s344	59	102 (32)	540	366	1.48	-	-	-
s382	95	117 (35)	960	441	2.18	-	-	-
s400	121	124 (33)	1,220	430	2.84	-	-	-
s444	160	118 (27)	1,610	370	4.35	-	-	-
s641	108	106 (72)	763	544	1.40	-	-	-
s713	105	98 (68)	848	582	1.46	-	-	-
s953	218	234 (116)	1,090	819	1.33	-	-	-
s1423	237	290 (77)	5,474	2,006	2.73	-	-	-
s5378	1,403	1,268 (376)	43,524	12,578	3.46	9,282	3,530	2.63
s9234.1	1,889	1,510 (645)	213,570	73,862	2.89	45,360	16,368	2.77
s13207.1	2,173	2,494 (807)	663,070	248,126	2.67	134,788	51,782	2.60
s15850.1	4,861	4,472 (767)	928,451	150,392	6.17	189,618	33,656	5.63
s35932	509	2,414 (257)	156,570	81,362	1.92	32,130	18,410	1.75

** Uses scan shifting only. The ordering of scan register is the default order.*

function, $P(i,j)$ represents the number of possible conflicts as indicated in Heuristic 1. $L(i,j)$ weights down long equal-length paths, which might not result in conflicts as indicated in Heuristic 2.

We restrict our attention to equal-length shortest paths for two reasons: 1) Such paths are more likely to cause conflicts for value justification than long ones, and 2) evaluating the number of long paths between vertices is computationally difficult, and the problem formulation, even if it is solvable, requires an evaluation that is unverifiable in polynomial time. On the other hand, we can determine the number of distinct shortest paths between two vertices of a directed graph in polynomial time using a modified form of Dijkstra's algorithm.[6]

We may also modify the cost function to include the routing factor. If we can estimate the cost of routing for each ordered pair of adjacent flip-flops, we can simply combine it with the cost function through a weighting factor.

A simulated annealing optimization loop found an efficient order to the formulated problem. The "moves" in the simulated annealing loop consist of swapping positions of two elements in the current order.

Experimental results

This study modified sequential test generator STG3[3] to incorporate scan shifting in the test generation process. Table 1 lists test generation results for ISCAS89 (International Symposium on Circuits and Systems 1989) benchmark circuits. A procedure based on the cycle-breaking criterion and other heuristics[7] selects scan flip-flops and assumes that the scan chain flip-flop order follows the original netlist file order (default order). Test efficiency is the number of detected faults divided by the total faults minus the identified untestable faults. The table also lists test generation results for two-clock partial-scan circuits. Fault coverage reduction varies from 0% to 4.2%.

Table 2 compares the test lengths of single- and two-clock partial-scan designs. Columns 2 and 3 shows the number of vectors produced by the test generators (not including the scan sequences) for each design. The number of generated subsequences appears in parentheses in column 3. A subsequence is the sequence generated for a target fault using the model shown in Figure 3. We need to add a scan sequence before and after each subsequence. Therefore, the number in parentheses plus one equals the number of scan sequences to be inserted in the final test set.

The total length of the final test set including the scan sequences (that is, the total number of clock ticks) depends on the number of scan chains. Therefore, the table includes information for single- and five-chain designs. This comparison did not consider circuits with less than 30 scan flip-flops. In all cases, the total clock ticks for the single-clock design is significantly shorter than the two-clock design's. Therefore, incorporating scan shifting in test generation seems to be a good strategy for test compaction.

Because the fault coverages for two- and single-clock designs in Table 1 are slightly different, directly comparing the

Table 3. *Test lengths for single- and two-clock circuits with equal fault coverages.*

Circuit	Additional vectors generated by hybrid strategy	Single-chain sequence: Test length (clock ticks) Two clock	Hybrid*	Ratio (two÷hybrid)	Five-chain sequence: Test length (clock ticks) Two clock	Hybrid*	Ratio (two÷hybrid)
s5378	121	43,524	16,360	2.66	9,282	4,384	2.12
s9234.1	195	213,570	95,897	2.23	45,360	21,072	2.15
s13207.1	40	663,070	260,326	2.55	134,788	54,324	2.48
s15850.1	1,540	928,451	444,532	2.09	189,618	93,755	2.02
s35932	0	156,570	81,362	1.92	32,130	18,410	1.75

**Uses scan shifting first. If a fault is reported as undetectable, the unmodified STG3 is used to generate additional vectors.*

Table 4. *Results for default and optimized ordering of the scan register in single-clock circuits. (Two-clock results included for comparison.)*

Circuit	Fault coverage (%): Two clock	Single clock: Default	Single clock: Optimized	Test efficiency (%): Two clock	Single clock: Default	Single clock: Optimized	CPU time (s): Two clock	Single clock: Default	Single clock: Optimized
s344	97.1	97.1	97.1	98.8	98.8	98.8	21	21	21
s382	99.5	98.5	99.3	100.0	100.0	100.0	2	3	3
s400	98.1	96.7	97.9	100.0	100.0	100.0	3	3	3
s444	96.0	91.8	94.3	98.1	95.8	96.1	99	236	244
s641	94.4	94.4	94.4	100.0	100.0	100.0	5	5	5
s713	88.5	88.5	88.5	100.0	100.0	100.0	19	18	18
s953	100.0	97.2	97.2	100.0	100.0	100.0	13	63	64
s1423	80.3	78.9	79.0	81.2	79.7	79.9	5,340	5,520	5,400
s5378	93.7	92.7	92.7	99.4	98.8	98.8	720	1,080	1,080
s9234.1	93.3	90.5	93.1	99.7	99.3	99.5	1,620	2,820	2,520
s13207.1	97.5	97.4	97.4	99.2	99.1	99.1	2,880	3,300	3,300
s15850.1	93.9	92.7	93.1	96.9	95.8	96.2	7,440	9,240	8,220
s35932	89.8	89.8	89.8	99.9	99.9	99.9	12,780	13,680	12,840

test sequence lengths may be inappropriate. A fair comparison required another experiment. First, the modified test generator (with scan shifting) performed test generation. If it reported any untestable fault, the experiment then called the unmodified STG3 (which treats the pseudo inputs as primary inputs). The unmodified STG3 generated additional vectors for those single-clock untestable, two-clock testable, faults. The entire "hybrid" test sequence generated this way can achieve the same fault coverages as those listed in Table 1 for the two-clock design.

Table 3 compares the length of the hybrid sequence with that generated by the unmodified STG3 for a two-clock design. Again, the comparisons only consider circuits with more than 30 scan flip-flops. For a single scan chain, the sequence length generated by STG3 is 92-166% longer than that generated by the hybrid strategy. With five scan chains, the difference is 75 to 148%.

Table 4 compares results for our fault-coverage optimized order and default order. This gives us a trivial upper bound on the fault coverage for any ordering. These results show that our heuristic for scan ordering does improve the fault coverage over that of the default order. For all these examples, our scan ordering algorithm takes only seconds to run.

SINGLE-CLOCK PARTIAL SCAN is a practical and cost-effective DFT technique. Investigating the incorporation of functional justification to improve fault coverages for such designs is worth pursuing. D&T

Acknowledgments

I am grateful to Ashok Vittal and Saikat Bhattacharya for their contributions to formulating and implementing the scan ordering algorithm.

References

1. K.-T. Cheng, S. Devadas, and K. Keutzer, "Delay-Fault Test Generation and Synthesis for Testability Under a Standard Scan Design Methodology," *IEEE Trans. CAD*, Vol. 12, No. 8, Aug. 1993, pp. 1217-1231.
2. J. Savir, "Skewed-Load Transition Test: Part I, Calculus," *Proc. Int'l Test Conf.*, IEEE Computer Society Press, Los Alamitos, Calif., 1992, pp. 705-713.
3. W.T. Cheng, "The BACK Algorithm for Sequential Test Generation," *Proc. Int'l Conf. Computer Design*, IEEE CS Press, 1988, pp. 66-69.
4. R.A. Marlett, "EBT: A Comprehensive Test Generation Technique for Highly Sequential Circuits," *Proc. Design Automation Conf.*, IEEE CS Press, 1978, pp. 335-339.
5. K.-T. Cheng, "Transition Fault Testing For Sequential Circuits," *IEEE Trans. CAD*, Vol. 12, No. 12, Dec. 1993, pp. 1971-1983.
6. E.W. Dijkstra, "A Note on Two Problems in Connection With Graphs," *Numerische Mathematik [Numerical Mathematics]*, 1959, pp. 269-271.
7. S. Bhawmik et al., "Pascant: A Partial Scan and Test Generation System," *Proc. IEEE Custom Integrated Circuits Conf.*, IEEE Electron Devices Society, San Diego, Calif., 1991, pp. 17.3.1-17.3.4.

Kwang-Ting (Tim) Cheng is currently an associate professor of electrical and computer engineering at the University of California, Santa Barbara, after several years at AT&T Bell Laboratories in Murray Hill, New Jersey. His current research interests include VLSI testing, synthesis, and verification. He is coauthor of *Unified Methods for VLSI Simulation and Test Generation* (Kluwer Academic Publishers, 1988), holds three US patents, and received a Best Paper award at the 1994 Design Automation Conference. Cheng holds a BS degree in electrical engineering from National Taiwan University and a PhD in electrical engineering and computer science from the University of California, Berkeley. He serves on the editorial boards of *IEEE Transactions on Computer-Aided Design*, *IEEE Design and Test of Computers*, and the *Journal of Electronic Testing: Theory and Applications*. Cheng is a member of the IEEE.

Address questions or comments about this article to the author at the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106-9560; timcheng@ece.ucsb.edu.