

Timing-Driven Partial Scan

BESIDES LOW AREA overhead, one key advantage of partial scan design over other design-for-testability (DFT) techniques is low (or zero) performance degradation if the designers select scan flip-flops carefully. However, no existing technique formally addresses this issue. (See the box, Full and partial scan, for background on existing techniques.) We propose a partial scan approach that aims at minimizing both area overhead and performance degradation caused by the test logic.

Because meeting performance objectives is one of the most difficult and time-consuming design tasks—especially in high-performance ASIC design—designers should guide scan flip-flop selection to meet the given performance requirement. If testability concerns make performance requirement violation inevitable, designers should guide selection to reduce the potential extra area caused by the subsequent performance optimization for target speed.

In this article, we enhance the cycle-breaking method introduced in Cheng and Agrawal.¹ Given a target speed and an initial design that meets the target, our algorithm selects a minimum set of scan flip-flops that 1) breaks all sequential cycles and 2) does not violate the performance requirement after the scan logic is added. If such a set does

JING-YANG JOU
National Chiao Tung
University, Taiwan
KWANG-TING (TIM) CHENG
University of California,
Santa Barbara

This partial scan approach reduces area overhead and performance degradation caused by test logic. Given an initial design that meets a target speed, the authors' algorithm selects a set of scan flip-flops that allows the circuit to meet performance requirements after the scan logic is added. If no such set exists, the algorithm selects a set that minimizes the total area increase caused by the scan logic and the subsequent performance optimization the circuit requires to meet target speed.

not exist, the algorithm finds a set of scan flip-flops that 1) breaks all sequential cycles and 2) minimizes the total area increase caused by the scan logic and the subsequent performance optimization.

In addition, for circuits synthesized by automatic synthesis tools, we suggest a new design flow, through which we

select and insert partial scan logic after area optimization but before performance optimization. This produces designs with less area increase than the traditional design flow, which considers testability and adds test logic after performance optimization.

This work represents an important concept in design for testability: during the synthesis phase, the designer must simultaneously consider the three major design parameters—performance, area, and testability—to obtain the right partial scan solution.

Algorithm

The inputs to our algorithm include the target speed, netlist, and each flip-flop's slack reported from a static timing analyzer; the algorithm's output is a set of flip-flops for scan. We define the slack of a flip-flop as the slack of the flip-flop's data input terminal. The slack of a signal is the difference between the required time from the performance specification and the real arrival time. If the design does not meet the performance requirement, some flip-flops have negative slacks.

Figure 1 shows the area overhead and performance degradation a circuit incurs when the designer inserts partial scan logic. If the resulting circuit does not meet the target speed, the designer can trade additional area for speed by

Full and partial scan

In the scan design technique, a well-known method of design for testability (DFT), the designer chains all flip-flops in the circuits into one or more shift registers. Because all the circuit's registers can be controlled and observed directly from the primary inputs and outputs, the designer can partition the sequential circuit into a purely combinational circuit and a shift register that can be tested separately. This technique substantially reduces test generation cost and complexity.²

Many companies have been developing and using comprehensive scan-based automatic test pattern generation systems for more than a decade. NEC's Scan-Path design system³ and IBM's Level-Sensitive Scan Design (LSSD) system⁴ are the most well-known examples. As these systems continue to evolve, other groups are introducing similar systems.^{2,5} Major differences among these systems come from different clocking methodologies and choices of scannable flip-flops. These choices lead to a variety of design rules and area/performance overheads.

In partial scan, the scan chain includes only some of the flip-flops, reducing area overhead. Designers can also minimize the timing penalty by excluding flip-flops on critical paths from the scan register. Due to its low overhead, the partial scan design methodology is becoming popular as a major DFT technique for sequential circuits.

Designers have applied a variety of techniques to automate selection of the flip-flops for the partial scan path. In Trischler's approach,⁶ the designer uses testability analysis to select scan flip-flops and then a sequential circuit test generator to obtain the test vectors. Agrawal et al.⁷ use functional vectors and a combinational circuit test generator to select scan flip-flops. This method generates test vectors only for the faults not detected by the functional vectors.

Another approach shows that purely structural analysis of a graph associated with the circuit is also effective for scan flip-flop selection.⁸ Parikh and Abramovici⁹ rank all flip-flops based on a sensitivity analysis that determines the circuit's improvement in testability as a result of scanning a flip-flop. With this method, designers can also estimate the number of scan flip-flops needed to reach good fault coverage.

Recently, Cheng and Agrawal proposed a partial scan method in which flip-flop selection aims at breaking up the circuit's cyclic structure.¹ The resulting circuit remains sequential in test mode, and the designer can easily handle it with a sequential circuit test generator. This approach produces high fault coverage with relatively low area overhead. Gupta, Gupta, and Breuer⁸ and Kunzmann and Wunderlich¹⁰ also use this cycle-breaking concept.

using various timing optimization techniques¹¹ (see Figure 2).

If the designer selects flip-flop f for scan, the circuit incurs extra delay for all combinational paths ending at f . The amount of extra delay is roughly equal to the delay of a 2-to-1 multiplexer, denoted by d_{mux} . Notice that the actual ex-

tra delays various scan flip-flops incur are slightly different due to different loads they drive. However, the differences are quite small, and we consider them negligible.

The circuit does not incur performance degradation if the designer selects a flip-flop for scan with slack

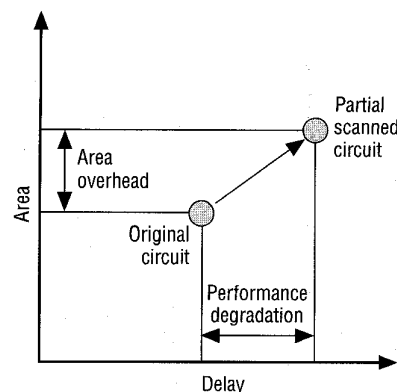


Figure 1. Partial scan overhead.

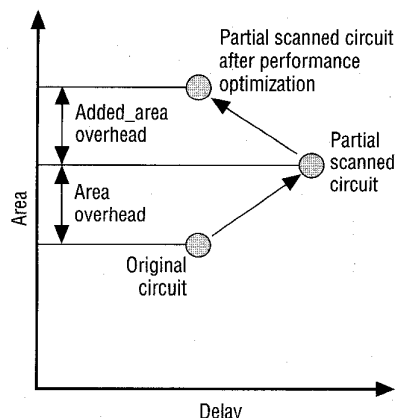


Figure 2. Total area overhead of partial scan.

greater than d_{mux} . However, if the designer selects a flip-flop with slack less than d_{mux} , either the circuit's performance will degrade or performance optimization will require extra area overhead to meet the target speed. The more flip-flops with negative slacks the designer chooses for scan, the more area the performance optimizer adds to make the circuit meet the target speed.

A circuit's area and delay have a trade-off relationship; circuitry can either trade area for speed or speed for area. However, it is extremely difficult to predict the exact amount of area overhead that will reduce extra delay. Our practical experience suggests that

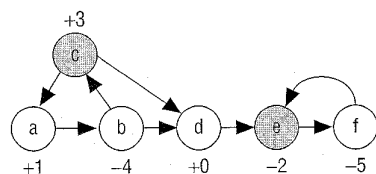


Figure 3. Example of a circuit graph.

```

begin
  clean up
  graph reduction
  while (graph is not completely
    reduced) do
    begin
      heuristic node selection
      graph reduction
    end
  end
end

```

Figure 4. Lee-Reddy algorithm.

the product of area and delay is approximately a constant,¹¹ that is, $area \times delay = c$.

If, given target delay D , flip-flop f has negative slack s_f , we express the additional area introduced by the performance optimizer to meet the target delay as

$$Added_area = c\{[1/D] - [1/(D - s_f)]\}$$

If the designer selects this negative-slack flip-flop for scan, its slack decreases by d_{mux} . We then express the difference in area added to meet the performance requirement between selecting and not selecting f for scan as

$$d(D, s_f) = c\{[1/(D - s_f)] - [1/(D - s_f + d_{mux})]\}$$

Similarly, if the slack of flip-flop f is nonnegative but less than d_{mux} , we express the difference in area added to meet the performance requirement between selecting and not selecting f for scan as

$$e(D, s_f) = c\{[1/D] - [1/(D - s_f + d_{mux})]\}$$

Given a netlist, set of flip-flops S , and target delay D , we define the area overhead function (AOF) for scanning the flip-flops in S as

$$AOF(S) = a \cdot |S| + \sum_{f \in S} u(s_f)$$

Here, a is a constant equal to the value of the area difference between a scan flip-flop and a nonscan flip-flop, and

$$u(x) = \begin{cases} d(D, x) & \text{if } x < 0 \\ e(D, x) & \text{if } 0 \leq x < d_{mux} \\ 0 & \text{if } x \geq d_{mux} \end{cases}$$

The first term of the $AOF(S)$ equation represents the area overhead of replacing the flip-flops in S by their scan versions. The second term reflects the additional area overhead required to achieve the target speed if S is scanned. To minimize AOF, we must simultaneously minimize the number of flip-flops in the scan set and, if possible, avoid selecting flip-flops with negative slacks. Notice that we know or can easily derive all constants in the previous expression. To derive c , we simply take the product of the area and delay of the original netlist.

We model a sequential circuit as a weighted directed graph $G = (V, E, W)$, where vertex v_i represents a flip-flop, and edge (v_i, v_j) exists when there is a combinational path from flip-flop i to flip-flop j . Each vertex v_i is associated with weight w_i , the slack of corresponding flip-flop i . Figure 3 shows the graph of a six-flip-flop circuit. The character inside the vertex denotes the vertex name, and the number next to the vertex denotes the slack of the corresponding flip-flop.

For test generation, cycles of length greater than 1 pose a greater problem than cycles of length 1 or self-loops.¹ In most designs, a significant percentage of flip-flops have self-loops. Thus, scanning flip-flops to break all sequential cycles including self-loops results in a

design close to a full scan circuit. Therefore, we consider it reasonable to leave self-loops unbroken, and we adopt this heuristic in our algorithm.

There are many ways to break all the cycles in the graph in Figure 3, but some choices incur less performance degradation than others. Choosing vertices c and e to break all cycles results in a testable design with minimum performance degradation, because vertices c and e have greater slacks than the other candidates.

Given a weighted directed graph, our algorithm finds a set of vertices that 1) breaks all cycles except self-loops and 2) provides all with positive weights. If such a set does not exist, our algorithm finds a set that breaks all cycles and minimizes the area overhead function.

We state this problem formally as follows: Given weighted directed graph G , find a set of vertices V' such that $G - V'$ is acyclic and $AOF(V')$ is minimum.

This problem differs from (and is harder than) the feedback vertex set problem that other partial scan methods solve. Because the feedback vertex set problem is an NP-complete problem, we cannot find an efficient, exact algorithm for our problem. However, we can derive a good heuristic algorithm from the feedback vertex set problem, which has been studied extensively. As an example, we describe an algorithm that we modified from the Lee-Reddy algorithm.¹²

Like the Lee-Reddy algorithm, our algorithm consists of two major steps: graph reduction and heuristic node selection. Figure 4 outlines this iterated algorithm.

In the clean-up step, we simply remove all self-loops to reduce the area overhead as explained earlier for the subsequent cycle-breaking task.

In the graph reduction step, we reduce the graph while preserving all the properties to find a feedback vertex set with minimal AOF. The graph reduction step has five operations; the first three

are exactly the same as those given in Lee and Reddy,¹² while the last two take the nodes' weights into account.

1. Source operation. Given node v with in_degree 0, remove v and all incident edges. (The in_degree of a node is the number of fan-ins to that node.) This is necessary because selecting source nodes does not contribute to cycle-breaking tasks.
2. Sink operation. Given node v with out_degree 0, remove v and all incident edges (same reason as operation 1). (The out_degree of a node is the number of fan-outs from that node.)
3. Self-loop operation. Given node v with self-loop, include v in the feedback vertex set, and remove v and all edges incident to v . Any node that ends up with a self-loop during the iteration process is the only node eligible for breaking this self-loop.
4. Conditional unit in operation. Given node v with in_degree 1 and no self-loop, if the weight of its unique predecessor v' is greater than d_{mux} , merge v into v' as a single node. Node v' can break any cycles node v can break, and from the timing point of view v' is always the better candidate.
5. Conditional unit out operation. Given node v with out_degree 1 and no self-loop, if the weight of its unique successor v' is greater than d_{mux} , merge v into v' as a single node (same reason as operation 4).

If these operations reduce the graph completely, the selected feedback vertex set is an optimal solution to our problem. Otherwise, we apply heuristics to select a node from those remaining, add it to the set, and remove it and its incident edges from the graph. We iteratively apply the graph reduction and heuristic node selection steps until the graph is empty.

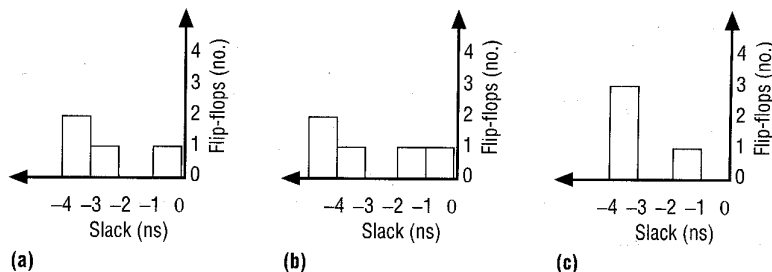


Figure 5. The slacks of s1423 with target delay 88 ns. Original circuit (a), cycle-broken circuit (b), and timing-driven cycle-broken circuit (c).

In the heuristic selection step, we choose a node with 1) slack greater than d_{mux} and 2) a maximum product of in_degree and out_degree . In general, removing a node of large fan-ins and fan-outs will break more cycles. If no such node exists (all remaining nodes have slacks less than d_{mux}), we select a node with a maximum value of $(\text{in_degree})(\text{out_degree}) + w(\text{slack})$, where w is a constant. If we assign w a large value (greater than the product of the maximum in_degree and maximum out_degree of the graph), the algorithm selects a node with maximum slack. On the other hand, if we assign a zero value to w , the selection criterion is the same as the one suggested in Lee and Reddy.¹² In our current implementation, we assign w the product of the average in_degree and out_degree of the original circuit graph.

Our algorithm selects a very different set of flip-flops from that selected by the Lee-Reddy algorithm if any of their selected flip-flops has negative slack. For example, circuit s1423 has a total of 74 flip-flops. If we implement the circuit with the AT&T 1.2- μm , CMOS standard-cell library and the target delay is 88 ns, four flip-flops have negative slacks. (In comparing the algorithms, we are concerned only with negative-slack flip-flops.)

Figure 5a shows the bar chart of s1423's flip-flop count versus slack. Figure 5b is the chart for the partial scan circuit with scan flip-flops selected by

the Lee-Reddy algorithm. This algorithm selects not only the original four negative-slack flip-flops, but another flip-flop whose slack becomes negative when selected. This scan set increases the longest delay of the circuit by about 1.3 ns.

The bar chart in Figure 5c shows flip-flop count versus slack for the partial scan circuit with scan flip-flops selected by our algorithm. Our algorithm selects only two of the original negative-slack flip-flops for scan, and the longest delay remains the same as that of the nonscan circuit. However, the Lee-Reddy algorithm selects only 22 flip-flops for scan, while our algorithm selects 29.

Experimental results

We used several ISCAS89 sequential benchmark circuits¹³ to compare our partial scan method with the existing methods. We sampled two pure FSMs—s386 and s510—that have a small number of flip-flops but complex structures. We also chose five large designs with a larger number of flip-flops (ranging from 29 to 531).

First, we ran a technology-mapping program to map the technology-independent netlists onto AT&T 1.2- μm , CMOS standard-cell library. Then, we used AT&T's performance optimizer LATTIS¹¹ to reduce the critical path delay by 50% for each circuit. We call the resulting circuits "reference circuits." Because partial scan with cycle-break-

Table 1. Comparison of TD_Pscan and Lee-Reddy algorithm.

Circuit	Reference circuits		Lee-Reddy		TD_Pscan	
	Area (grids)	Critical path delay (ns)	Area overhead (%)	Delay overhead (%)	Area overhead (%)	Delay overhead (%)
s386	368	13.00	8.2	8.5	8.2	1.5
s510	675	17.80	4.4	7.3	4.4	6.7
s953	1,232	20.80	2.4	5.8	2.4	1.4
s1423	2,428	42.90	5.4	2.6	7.4	2.1
s5378	4,689	25.77	1.2	0.0	1.2	0.0
s9234	5,684	28.19	0.4	0.0	0.4	0.0
s15850	14,653	42.70	3.8	1.2	4.7	0.0

Table 2. Comparison of TD_Pscan and Lee-Reddy with subsequent timing optimization.

Circuit	Reference circuits		Lee-Reddy		TD_Pscan	
	Area (grids)	Critical path delay (ns)	Area (grids)	Total area overhead (%)	Area (grids)	Total area overhead (%)
s386	368	13.00	470	27.70	441	18.90
s510	675	17.80	719	6.50	707	4.80
s953	1,232	20.80	1,510	22.60	1,302	5.90
s1423	2,428	42.90	2,616	7.70	2,649	9.10
s5378	4,689	25.77	4,745	1.20	4,745	1.20
s9234	5,684	28.19	5,707	0.40	5,707	0.40
s15850	14,653	42.70	15,830	8.00	15,343	4.70

ing achieves very high fault coverage,¹ our experiments concentrated only on comparing area and delay overheads.

Our first experiment compared our timing-driven flip-flop selection algorithm with the Lee-Reddy algorithm. We implemented our algorithm in a program called TD_Pscan. In this experiment, we did not optimize performance after scan flip-flop selection and insertion. We used the reference circuit's delay as the target delay for the timing-driven scan flip-flop selection.

In Table 1, columns 2 and 3 list the areas and critical path delays of the reference circuits; columns 4 and 5 list the area and delay overheads of the partial scan circuits produced by the Lee-Reddy algorithm; and columns 6 and 7 list the area and delay overheads of the circuits produced by TD_Pscan.

For s386, s510, and s953, TD_Pscan selected the same number of scan flip-flops as the Lee-Reddy algorithm, but generated designs with substantially less performance degradation. For s1423, TD_Pscan selected more scan flip-flops, but resulted in lower performance degradation. For s5378 and s9234, all the scan flip-flops selected by the Lee-Reddy algorithm happened to be in noncritical paths. Thus, TD_Pscan selected exactly the same sets of scan flip-flops. For s15850, TD_Pscan selected more flip-flops (corresponding to 4.7% area overhead) than the Lee-Reddy algorithm (3.8% area overhead). However, our method caused no performance degradation, while the Lee-Reddy algorithm introduced 1.2%. It is worth noting that the weight w introduced in our algorithm's heuristic node

selection step can be adjusted to trade area for speed in scan design circuits.

In our second experiment, right after scan flip-flop selection we optimized the circuits' timing to meet the original target performance. Table 2 shows experimental results comparing our approach with the existing approach, which does not use timing information for scan flip-flop selection.

For each circuit, we chose the reference circuit's delay as the target delay. We used the Lee-Reddy algorithm on each reference circuit to select scan flip-flops. For five out of seven cases, the target speeds were not satisfied after we added the scan logic. We then used LATTIS to optimize the partial scan circuit to meet the original target speed. Column 4 in Table 2 shows the areas of the final circuits.

Then, we used TD_Pscan to select scan flip-flops for the same seven circuits. Column 6 lists the areas of the circuits this procedure produced. For both partial scan approaches, total area overhead (columns 5 and 7) represents the true area overheads including both overhead due to the scan logic and additional overhead for meeting the performance requirement.

Table 1 shows that for circuit s953, the Lee-Reddy algorithm selected scan flip-flops causing 2.4% area overhead and 5.8% performance degradation. However, to meet the target critical path delay of 20.8 ns, we must convert the 5.8% performance overhead into an additional 20.2% area overhead ($22.6\% - 2.4\%$), as Table 2 indicates. Clearly, approaches that do not consider the extra area overhead that compensates for performance degradation caused by scan logic might not represent the whole story on area overhead.

Another interesting example is s15850. Our approach selects more scan flip-flops before performance optimization and shows a higher area overhead as compared with the Lee-Reddy approach. But, after performance optimization, our approach's final area overhead is smaller than that of the old approach. In other words, the additional flip-flops eventually pay off when the circuit meets target speed.

For s1423, our approach selects more scan flip-flops. In this case, the final area overhead of our approach remains higher than that of the Lee-Reddy approach (see Table 2). This is because our flip-flop selection algorithm is heuristic and does not guarantee the optimal solution to the problem. Fine-tuning of parameter w in the heuristic selection step may improve the results.

The results of timing-driven partial scan depend greatly on the particular timing optimizer used. Timing optimization technology is very diversified, and there are no golden algorithms that always produce the best results. Thus,

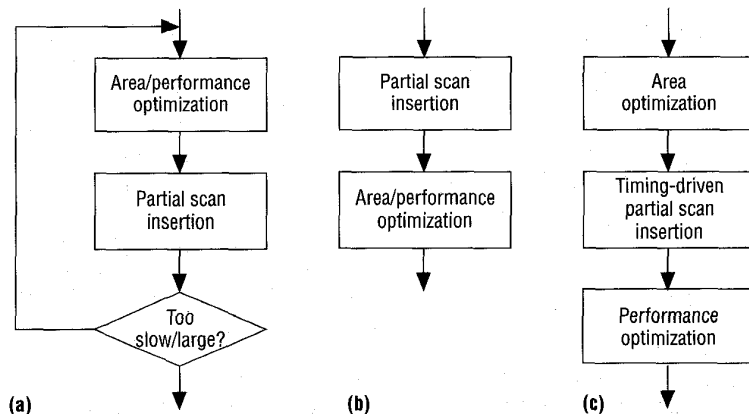


Figure 6. Traditional design flows 1 (a) and 2 (b), and design flow 3, our suggestion (c).

we do not intend our results to be used for benchmarking. However, our results do show that carefully selected flip-flops on the less critical paths generally result in lower area overhead for partial scan insertion.

Choosing a design flow

We have demonstrated that designers can use timing information effectively to reduce area overhead for partial scan insertion. However, a circuit's trade-off relationship between area and delay can only be well quantified when the circuit is in reasonably good shape. In other words, we cannot effectively use timing information to estimate the added area that will reduce delay unless the circuit is area optimized to some extent. Here, we discuss how three design flows meet area, delay, and testability requirements for circuit designs. We show that one of them is superior to the others in producing a smaller, faster, testable design.

Design flow 1. In the traditional design flow (Figure 6a), testability improvement occurs after the designs have met their area and performance specifications. One obvious drawback of this design flow is that test logic usually degrades performance, rendering the previous optimization effort waste-

ful. Another disadvantage is that functional logic and test logic are optimized separately, especially for timing. Even if we try to add the test logic in a way that least degrades performance, the performance requirement may still be violated, and performance optimization must be done again for the circuit to meet target speed.

Design flow 2. In another typical design flow (Figure 6b), designers insert testability before optimizing both area and timing. It is difficult to predict the final area overhead for the particular set of flip-flops selected for scan in the unoptimized initial circuit. Therefore, the advantage of the timing-driven partial scan insertion cannot be exploited.

Design flow 3. We propose a better design flow, illustrated in Figure 6c. After performing area optimization, we use the flip-flops' slack information and the circuit's target speed to guide scan flip-flop selection. Judicious use of the slack information contributes directly to the subsequent performance optimization, in which we trade area for delay to achieve the original circuit speed.

There are two advantages to our design flow: it avoids the iteration of performance optimization, and it produces designs whose area-delay trade-off

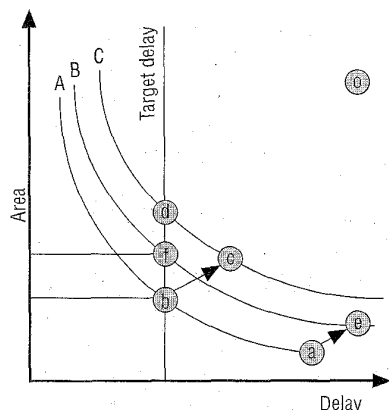


Figure 7. Typical area-delay trade-off curves.

curves are generally superior to those of designs produced by traditional flows.

Comparison. We can use area-delay trade-off curves to elaborate the differences in area overhead incurred through the different design flows. In Figure 7, curve A shows the area-delay trade-off curve for a design without partial scan. Design flows 3 and 1 (Figures 6c and 6a) yield designs with trade-off curves B and C.

The vertical line in Figure 7 represents the design's target speed; point o represents the initial design. Design flow 1 first optimizes the design for area, moving the design to point a. It then speeds up the design, moving the design to point b on curve A to meet the performance requirement. Partial scan logic, added later, causes area overhead and usually degrades performance, moving the design to point c on curve C. Since the target speed is no longer satisfied, the designer must again conduct performance optimization, producing the design at point d.

If we follow design flow 3, we first optimize the initial circuit for area. This changes the design from o to a. In the next step, we insert timing-driven partial scan, changing the design from a to e. The performance optimizer then pro-

duces the final design, corresponding to point f on curve B.

Figure 7 clearly shows the difference in area between the designs these two approaches produce. The specific results shown in Table 2 further support this general observation. It is also worth noting that the higher the target speed, the larger the difference in area between designs produced by these two approaches.

TWO NEW CONCEPTS emerge in this article. First, our timing-driven partial scan scheme uses timing information to guide scan flip-flop selection. Our algorithm selects scan flip-flops that break all cycles (except self-loops) and minimize area overhead, including area the performance optimizer adds to resolve performance degradation due to test logic. Second, by our design flow, we select and insert the partial scan flip-flops before performance optimization. This flow not only avoids unnecessary repetition of the timing optimization, but also results in an area-delay trade-off curve superior to the one resulting from the traditional design flow.

Further enhancement of the algorithm to produce constraint-driven partial scan is an interesting topic. The goal is to integrate the objectives of optimizing area, performance, testability, and power into a unified set of design algorithms. 

Acknowledgments

This work was done when both authors were with AT&T Bell Laboratories, Murray Hill, New Jersey.

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. V.D. Agrawal, S.K. Jain, and D.M. Singer, "Automation in Design for Testability," *Proc. Custom Integrated Circuit Conf.*, IEEE, Piscataway, N.J., 1984, pp. 159-163.
3. S. Funatsu, W. Wakatsuki, and T. Arima, "Test Generation System in Japan," *Proc. ACM/IEEE Design Automation Conf.*, IEEE, 1975, pp. 114-122.
4. E. Eichelberger and T. Williams, "A Logic Design Structure for LSI Testability," *Proc. ACM/IEEE Design Automation Conf.*, IEEE, 1977, pp. 462-468.
5. P.P. Fasang et al., "Automated Design for Testability of Semicustom Integrated Circuits," *Proc. Int'l Test Conf.*, IEEE Computer Society Press, Los Alamitos, Calif., 1985, pp. 558-564.
6. E. Trischler, "Incomplete Scan Path with an Automatic Test Generation Methodology," *Proc. Int'l Test Conf.*, IEEE CS Press, 1980, pp. 153-162.
7. V.D. Agrawal, et al., "Designing Circuits with Partial Scan," *IEEE Design & Test of Computers*, Vol. 5, No. 2, Apr. 1988, pp. 8-15.
8. 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.
9. P.S. Parikh and M. Abramovici, "A Cost-Based Approach to Partial Scan," *Proc. ACM/IEEE Design Automation Conf.*, IEEE, 1993, pp. 255-259.
10. A. Kunzmann and H.-J. Wunderlich, "An Analytical Approach to the Partial Scan Problem," *J. Electronic Testing: Theory and Application (JETTA)*, May 1990, pp. 163-174.
11. J.P. Fishburn, "LATTIS: An Iterative Speedup Heuristic for Mapped Logic," *Proc. ACM/IEEE Design Automation Conf.*, IEEE, 1992, pp. 488-491.
12. D.H. Lee and S.M. Reddy, "On Determining Scan Flip-Flops in Partial Scan Designs," *Proc. Int'l Conf. Computer-Aided Design*, IEEE CS Press, 1990, pp. 322-325.
13. F. Brglez, D. Bryan, and K. Kozminski, "Combinational Profiles of Sequential

Benchmark Circuits," *Proc. Int'l Symp. Circuits and Systems*, IEEE, 1989, pp. 1929-1934.



Jing-Yang Jou is an associate professor in the Department of Electronic Engineering at National Chiao Tung University, Taiwan. He has worked in the GTE Laboratories and in the Design Automation Lab of AT&T Bell Laboratories. His research interests include behavioral and logic synthesis, synthesis for testability, computer architecture, parallel processing, and fault-tolerant computing.

Jou received a BS in electrical engineering from National Taiwan University, and MS and PhD degrees in computer science from the University of Illinois at Urbana-Champaign. He is a member of Tau Beta Pi and the IEEE, and the recipient of the Distinguished Article award of the IEEE International Conference on Computer-Aided Design, 1990. He is a member of the IEEE Circuits and Systems Society.



Kwang-Ting (Tim) Cheng is an associate professor of electrical and computer engineering at the University of California, Santa Barbara, after several years at AT&T Bell Laboratories. His current research interests

include VLSI testing, synthesis, and verification. Cheng holds a BS in electrical engineering from National Taiwan University and a PhD in electrical engineering and computer sciences from the University of California, Berkeley. He coauthored *Unified Methods for VLSI Simulation and Test Generation*, holds three US patents, and received a Best Paper award at the 1994 Design Automation Conference. He serves on the editorial boards of *IEEE Design & Test of Computers*, *IEEE Transactions on Computer-Aided Design*, and the *Journal of Electronic Testing: Theory and Applications*. Cheng is a member of the IEEE and the IEEE Computer Society.

Direct questions concerning this article to Jing-Yang Jou, Department of Electronics Engineering, National Chiao Tung University, 1001 Ta Hsueh Rd., Engineering Building 4, Room 539, Hsinchu, Taiwan, R.O.C.; jyjou@bestmap.ee.nctu.edu.tw.

**Late magazines?
No magazines?
Membership status
problems?
No answers to your
complaints?**

Let your Computer Society
Ombudsman cut through the
red tape for you.

Ombudsman
IEEE Computer Society
10662 Los Vaqueros Circle
PO Box 3014
Los Alamitos, CA 90720-1314
e-mail: membership@computer.org
fax: (714) 821-4641



IEEE COMPUTER SOCIETY

Moving?

Please notify us four weeks in advance.

Name (Please print)

New Address

City

State/Country

Zip

Mail to:
IEEE Computer Society
Circulation Department
PO Box 3014
10662 Los Vaqueros Circle
Los Alamitos, CA 90720-1314

- List new address above.
- This notice of address change will apply to all IEEE publications to which you subscribe.
- If you have a question about your subscription, place label here and clip this form to your letter.

**ATTACH
LABEL
HERE**