Testability Insertion in Behavioral Descriptions

Frank F. Hsu Elizabeth M. Rudnick Janak H. Patel
Center for Reliable & High-Performance Computing
University of Illinois, Urbana, IL

Abstract

A new synthesis-for-testability approach is proposed that uses control points at branch conditions to improve testability. Hard-to-control loops are identified through analysis of the control-data flow graph, and control points are added at the exit conditions of these loops. Test statements are also inserted if necessary to allow hard-to-control variables to be directly controllable from existing primary inputs. Implementation of the proposed techniques using the HLSynth92 and HLSynth95 benchmark circuits results in significant improvements in fault coverage and reductions in test set size and test generation time. Furthermore, the impact on area and performance is minimal, and the ability to do at-speed testing is not affected.

I Introduction

Two different approaches have been used to design testable circuits. The conventional approach has been to obtain a gate-level description of a design, either through synthesis or other means, and then to add design-for-testability (DFT) hardware. With this approach, the DFT techniques are applied near the end of the design cycle. The second approach is to use synthesis-for-testability (SFT) techniques to improve circuit testability prior to logic synthesis. Many studies have been conducted recently on the analysis of testability and the application of SFT schemes early in the design cycle [1]–[10]. Use of testability techniques prior to logic synthesis allows a designer to obtain an easily testable circuit with reasonable overhead [5]. The techniques presented in [6]–[8] utilize the information at the Register Transfer (RT) level to generate easily testable designs. Dey *et al.* [9] presented a method to break data flow loops by exploiting hardware sharing to minimize the usage of scan registers. In [10], Lee *et al.* presented a data path scheduling algorithm for easily testable systems. While all of the proposed SFT techniques have been successful in improving testability, most of them rely upon full scan, partial scan, or built-in-self-test (BIST) to implement the testability enhancements.

Thus, a majority of the current DFT and SFT approaches are actually implemented near the end of the design cycle, when the final netlist or the structural information about the circuit is known. Full scan and partial scan are the most commonly-used techniques in practice. Although the scan approach greatly reduces the difficulties of sequential circuit test generation, it also has many disadvantages. Besides the area overhead required to implement the scan chain and increased time needed for test application, scan-based solutions may have limited capabilities for at-speed test. Scan tests targeted at stuck-at faults cannot be applied at the operational speed of the circuit due to scanning in and out of flip-flop values. Although two-pattern tests targeted at delay faults can be used, they require either a more complex flip-flop design or a functional justification based path delay test generator [11], [12], which might not provide as high of fault coverage. Work done by Maxwell *et al.* [13] has shown that a stuck-at fault test set applied at clock-speed is able to identify more defective chips than a test set having the same fault coverage but applied at a slower speed.

Two SFT techniques that permit at-speed testing have been proposed in the past. The nonscan DFT technique presented in [14] uses RT level structural information to produce testable data paths. This approach utilizes multiplexers added at the gate level to implicitly break feedback loops in the data path and redirect data to improve controllability and observability of logic modules. Similarly, the technique proposed in [15], [16] provides a means of augmenting data flow paths by inserting test statements into the high-level description prior to logic synthesis. These high-level approaches improve the data path testability by adding buses and multiplexers to the circuit; however, high area overhead for routing these buses result if they are added indiscriminately.

We propose a new nonscan SFT technique to improve circuit testability while allowing for at-speed application of tests. The approach is based on an analysis of the controllability of branch conditions in the control-data flow graph (CDFG). The entry and exit

[*] This research was supported in part by the Semiconductor Research Corporation under contract SRC 95-DP-109, in part by DARPA under contract DABT63-95-C-0069, and by Hewlett-Packard under an equipment grant.

conditions of some loops in a CDFG are often hard to control, and hence they may cause difficulties during automatic test generation. We focus our study on these hard-to-control (HTC) loops in the system. A controllability measure is first employed to identify HTC loops in a CDFG. The controllability of the exit condition within each of the HTC loops is then enhanced in order to achieve efficient fault activation and facilitate fault-effect propagation during the test generation process. Improved controllability of the exit condition allows data paths that already exist in the original system to be used. Further analysis of the modified CDFG may reveal that some variables may be hard to control for certain data value ranges. In this case, additional test statements are inserted to allow these variables to be directly controllable from existing primary inputs. Unlike the previous approaches, our approach performs the behavioral modifications at the high level, such that any implementation of this behavior is *inherently testable*. The advantage of this method is that it can use any synthesis tool, since the technique does not require any modification in the synthesis procedures.

We applied our SFT technique to several high-level synthesis benchmarks currently available. The experimental results show that when this approach is used, the circuits generated often require a shorter automatic test pattern generation (ATPG) time and a smaller test set to achieve better fault coverage and ATPG efficiency. While the testability of the circuit is improved, the implementation of this technique requires minimal logic overhead and allows test vectors to be applied at clock-speed. In Section II, we will introduce our testability measure using the CDFG information available from the high-level description. Then the proposed nonscan SFT approach will be discussed in Section III, followed by experimental results in Section IV.

II Testability Measure

A typical control-data flow graph consists of operation nodes, decision nodes, and transition arcs connecting these nodes. As an example, the high-level description and flow graph for the Greatest-Common-Divider (GCD) circuit are shown in Figure 1. The rectangle processing function contains serial operations that are to be executed by the circuit. The diamond shape decision function denotes the decision node with branching conditions. Our *GCD* example contains a *while* loop, as described in the high-level program. The nodes in the *while* loop are connected by bold lines in the corresponding control-data flow graph shown in Figure 1. Before we study the characteristics of the control-data flow graph, some terminologies are defined as follows.

DEFINITION 1: *A node within a control-data flow graph is the* **locus of execution** *for the system if it is currently being executed by the system.*

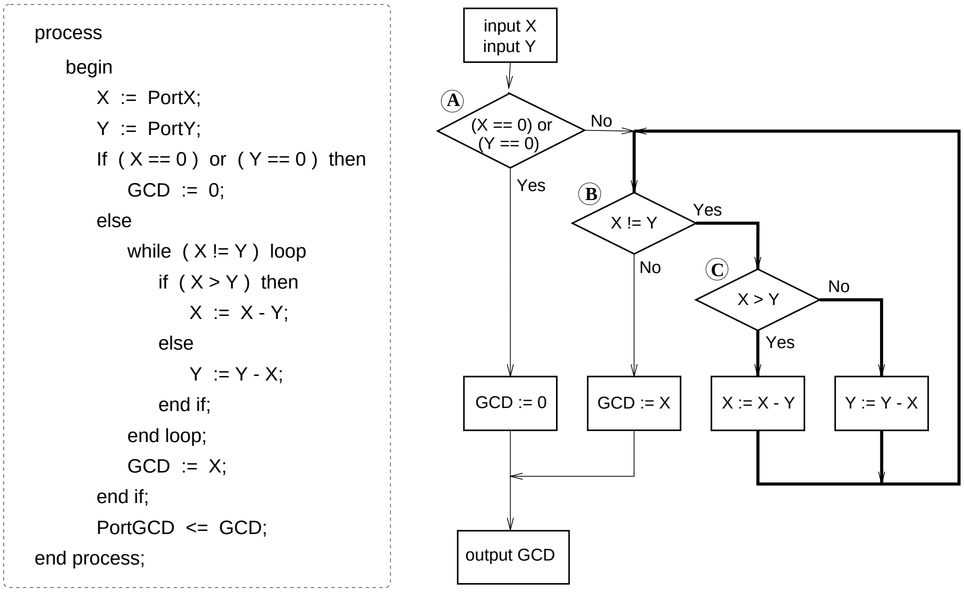
DEFINITION 2: *A decision node within a control-data flow graph is* **K-controllable** *if the direction of the branch taken can be controlled directly or indirectly by the input values* **K** *clock cycles before the* **locus of execution** *reaches the node, where* **K** *is the smallest such integer.*

DEFINITION 3: *A decision node within a control-data flow graph is* **non-controllable** *if the direction of the branch taken cannot be controlled directly or indirectly by the primary inputs within any predetermined number of clock cycles prior to the* **locus of execution** *reaching the node.*

In the remaining text, the term *locus* will mean **locus of execution**. During normal operation, the *GCD* circuit first reads values from the primary inputs. Then, depending on the input values, the system either sends the result to the primary outputs, or spends several iterations within the *while* loop before the result is ready for output. Notice that while the *locus* stays within the loop, the primary inputs are ignored and primary outputs are held constant. Since each iteration is triggered by a rising edge of the system clock, the system becomes uncontrollable and unobservable for several clock cycles until the *locus* exits the *while* loop. The controllability of a loop-exit node can be determined using the proposed testability measure of Definition 2, where a *loop-exit* node is the decision node that controls the exit condition of a loop. In the *GCD* example shown in Figure 1, decision node **A** is 1-controllable, since its decision can be directly controlled by the primary inputs within one clock cycle before the *locus* arrives at node **A**. Decision nodes **B** and **C** are not easily controllable, because the direction of the branch taken cannot be controlled by the primary inputs once the *locus* enters the while loop. Thus these decision nodes are marked as HTC nodes in the control flow.

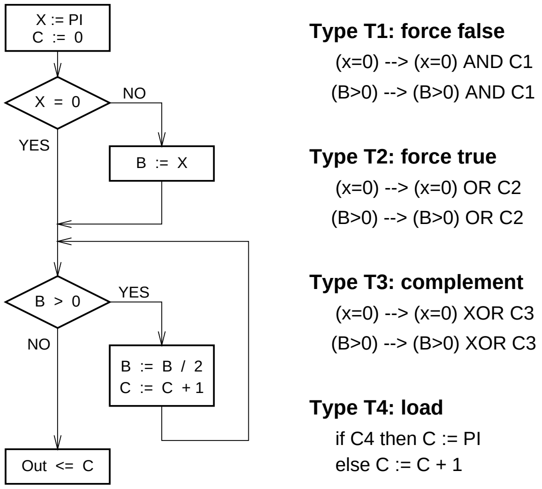
III Testability Insertion

Once the HTC nodes have been identified using the above testability measure, the controllability of these nodes can be augmented. Our SFT technique utilizes one or two extra test input pins to control the outcome of the conditional branches, directly guiding the control flow and indirectly affecting the values of the variables, as shown in Figure 2. We use control points of various types, depending on the situation. Control points of type T1 are **AND**'ed with the original branch condition to allow the condition to be forced *false*. Control points of type T2 are **OR**'ed with the original branch



```
process
    begin
        X  :=  PortX;
        Y  :=  PortY;
        If  ( X == 0 )  or  ( Y == 0 )  then
            GCD  :=  0;
        else
            while  ( X != Y )  loop
                if  ( X > Y )  then
                    X  :=  X - Y;
                else
                    Y  := Y - X;
                end if;
            end loop;
            GCD  :=  X;
        end if;
        PortGCD  <=  GCD;
end process;
```

Figure 1: The high-level description and the control-data flow graph for circuit GCD.



Type T1: force false
(x=0) --> (x=0) AND C1
(B>0) --> (B>0) AND C1

Type T2: force true
(x=0) --> (x=0) OR C2
(B>0) --> (B>0) OR C2

Type T3: complement
(x=0) --> (x=0) XOR C3
(B>0) --> (B>0) XOR C3

Type T4: load
if C4 then C := PI
else C := C + 1

Figure 2: Four types of controllability insertion in the high-level description.

condition to allow the condition to be forced *true*. Control points of type T3 enable the branch condition to be complemented through an **exclusive-OR** function. If hard-to-control variables remain in the circuit after control points of type T1, T2, or T3 have been used, a control point of type T4 is added to enable the variables to be loaded from existing primary inputs. Each test pin can be connected to various decision nodes in the flow graph as long as only one node with augmented controllability is being executed in any clock cycle. The added controllability not only reduces the difficulties of sequential circuit test generation, but it also increases the fault coverage when testing the synthesized circuits. The advantage of controlling the data path through the control of conditional branches is that the area overhead is small and is independent of the width of data registers.

Although the technique can augment any decision nodes in the CDFG, our presentation of this SFT approach will concentrate on improving the controllability of HTC exit nodes of functional loops. The GCD circuit contains a single loop. Decision node **B** of Figure 1 is chosen to demonstrate the advantages of our approach. The *non-controllable* node **B** causes the system to have very low controllability and observability while the *locus* stays within the loop. Implementation of the proposed testability scheme using a control point **C1** of type T1 **AND**'ed with the original loop-exit condition **(X!=Y)** allows the *locus* to exit the *while* loop using the extra control point **C1**. With the added controllability, faults activated within the loop can be quickly propagated to the primary outputs.

Additional improvements in controllability can be made by adding a control point of type T2. Control point **C1** of type T1 is **AND**'ed with the original loop-exit condition **(X!=Y)**; then the result is **OR**'ed with control point **C2** of type T2. While control point **C1** allows the *locus* of the system to escape the *while* loop without completing the computation, control point **C2** enables the *locus* to stay within the loop even after the result has been calculated. The purpose of **C2** is to activate certain faults within the loop when those faults cannot be activated under normal circumstances. Thus, by using two extra test signals to guide the direction of control flow, the test generator is able to create vectors that activate more faults.

Another variation of this approach is to use a single control point, **C3** of type T3, that is **exclusive-OR**'ed with the original branch condition. The use of control point **C3** allows the *locus* of the system to escape the *while* loop early or remain in the loop longer than it normally would. This modified approach is especially useful for reducing the number of test pins when multiple decision nodes are to be controlled independently.

The proposed techniques can be extended for cascaded loops and nested loops. Each loop represents a closely connected group of states in the total state space of the system. During normal operation, the *locus* of the system enters a state space at a specific entry state and exits at a specific termination state. When the *locus* stays within one state space, the functions of other state spaces lie idle. Implementation of the proposed SFT scheme provides extra transitions from the middle of a state space to the initial point of the next state space. Thus, the time required to traverse through all the state spaces is reduced because the extra test signal is able to efficiently pass the *locus* through the loops. As shown in [17], reducing the distance among states of a sequential circuit leads to higher fault coverage, and fewer test vectors are required to test the circuit. Even after the controllability of branch conditions has been improved, some variables may still be hard to control for certain data value ranges. Direct control of these variables is then necessary to make the circuit testable. Rather than adding multiple extra input pins, the variables are loaded from existing primary inputs under control of a single control point, **C4** of type T4. Control point **C4** is implemented using one extra test pin or existing test pins of type T1, T2, or T3.

In summary, the proposed SFT technique utilizes one or two extra test pins to guide the control flow during the testing process. The test pins provide a method for the *locus* to escape from a functional loop or to stay within a loop after the normal function of the loop is finished. As a result, the distances among the states are reduced, increasing the effectiveness of automatic test generation and reducing test application time. Additional test statements are inserted if necessary to allow HTC variables to be directly controllable from existing primary inputs.

IV Experimental Results

The proposed SFT techniques were implemented using the high-level synthesis benchmarks HLSynth92 and HLSynth95. The high-level circuit descriptions are written in VHDL code, and they were translated into a synthesizable subset of the VHDL language in order to evaluate their testability. Then the HTC nodes were identified, and proper modifications to the VHDL code were performed. Finally, gate-level implementations were obtained for the original circuits and circuits with enhanced testability using a commercial logic synthesis system. Characteristics of the benchmark circuits are listed in Table I. The table lists the number and types of functional loops that exist in each high-level description, the number of HTC loops, the number of HTC variables, and the number of conditional branches. Circuit *GCD* calculates the greatest common divisor value of two numbers; *Diffeq* solves differential equations; *Barcode* processes signals received from a barcode reader; and *DHRC* performs differential heat computation.

Several circuit implementations were synthesized for each high-level description. Experimental results are shown in Table II. One implementation uses the original description, and another uses the description with control point C1 of type T1 AND'ed with the branch condition. For all circuits except Barcode, a third implementation uses the description with two control points, C1 of type T1 and C2 of type T2, where C1 is AND'ed with the branch condition and C2 is OR'ed with the result. For some of the circuits, the original description with two control points of type T3 were also used; in this case, the control points were exclusive-OR'ed with the branch conditions for two decision nodes. In two circuits, one extra control point, C4 of type T4, was added to allow the HTC variables to be controlled directly from existing primary inputs.

Table I: Circuit Characteristics

Circuit	Functional Loops	HTC Loops	HTC Variables	Conditional Branches
GCD	1 single	1	0	3
Diffeq	1 single	1	1	1
Barcode	2 nested	2	1	5
DHRC	2 cascaded	2	0	2

Table II: Area and Performance Impact of Proposed SFT Techniques

Circuit	Control Points	Test Pins	Est. Area	Est. Delay	Primitives
GCD	-	0	1156	98	1354
GCD1	T1	1	1159	98	1398
GCD2	T1,T2	2	1146	98	1386
GCD3	T3(2)	2	1135	99	1370
Diffeq	-	0	31,776	93	40,924
Diffeq1	T1	1	30,842	105	39,430
Diffeq2	T1,T2	2	30,843	105	39,438
Barcode	-	0	678	47	713
Barcode1	T1	1	684	46	681
Barcode3	T3(2),T4	2	729	44	744
DHRC	-	0	4619	98	4950
DHRC1	T1	1	4745	99	4919
DHRC2	T1,T2	2	4775	99	4970
DHRC3	T3(2),T4	2	4562	99	4916

Table III: Deterministic ATPG Results of Proposed SFT Techniques

	Control	Test	Faults				Test	Fault	ATPG	Time
Circuit	Points	Pins	Total	Detected	Untestable	Aborted	Vectors	Cov.(%)	Eff.(%)	(min)
GCD	-	0	2145	1557	9	579	202	72.59	73.01	113.6
GCD1	T1	1	2113	1984	8	121	487	93.89	94.27	34.0
GCD2	T1,T2	2	2118	2068	6	44	792	97.64	97.92	17.6
GCD3	T3(2)	2	2254	2212	1	41	795	98.14	98.18	31.1
Diffeq	-	0	57,486	57,398	6	82	949	99.85	99.86	45.4
Diffeq1	T1	1	54,814	54,805	0	9	854	99.98	99.98	31.0
Diffeq2	T1,T2	2	54,819	54,807	0	12	754	99.98	99.98	30.9
Barcode	-	0	1037	613	16	408	1147	59.11	60.66	70.2
Barcode1	T1	1	999	738	19	242	2250	73.87	75.78	47.0
Barcode3	T3(2),T4	2	1306	1126	2	178	866	86.22	86.37	34.9
DHRC	-	0	8217	7272	68	877	488	88.50	89.33	163.6
DHRC1	T1	1	8327	7669	114	544	704	92.10	93.47	116.1
DHRC2	T1,T2	2	8260	7697	146	417	1091	93.18	94.95	93.1
DHRC3	T3(2),T4	2	8371	7792	100	479	1389	93.08	94.28	126.6

For the Barcode circuit, the test pins used to implement the T3 functions were also used for the T4 functions; thus, the number of test pins required was two. During logic synthesis, the optimization directive was set to minimize circuit area, and the timing constraint (clock cycles) was set to a fixed constant across different variations of a given design. Estimates of the area and delay were provided by the synthesis tool. The small variations among the sizes and delays are caused by the randomness of the circuit optimization process. As shown in Table II, the proposed SFT techniques produce almost no area overhead, while the circuit delays are kept within a reasonable range of the target speed.

The testability enhancements were evaluated using a commercial ATPG system that uses a deterministic, fault-oriented algorithm. Results are given in Table III. The faults modeled are stuck-at faults on the inputs and outputs of macro-cells, i.e., pin faults. Fault coverage is the percentage of total faults detected, and ATPG efficiency is the percentage of total faults either detected or identified as untestable. ATPG times are shown for an HP 9000 J200 workstation with 256 MB RAM. The proposed technique is able to increase the fault coverage and fault efficiency of the synthesized circuits by increasing the effectiveness of the test generation process. The number of aborted faults is greatly reduced by the augmented controllability in the circuits, and the time required to perform automatic test generation is often reduced.

The progression of testability improvement is shown by the results for the *GCD* circuits in Table III. The testability in general for *GCD* is very low, and many faults are aborted during test generation. By adding a control point of type T1, we are able to increase the fault coverage from 72.6% to 93.9%, and the time for test generation drops by 70%. Adding a control point of type T2 further increases the fault coverage to 97.6%, and the test generation time drops by 84%. Adding controllability to the “**if (X > Y)**” decision node in addition to the *while* loop using two test pins of type T3 results in further improvements in fault coverage. In summary, the testability of *GCD* is improved by the proposed scheme, and the impact on circuit area and performance is negligible.

Unlike *GCD*, which has low fault coverage initially, *Diffeq* is highly testable even without implementing any testability enhancement scheme. However, the proposed SFT approach can still be applied to further improve its testability. As shown in Table III, adding the extra control points enables the test generator to produce smaller test sets that achieve higher fault coverage and are generated within a shorter period of time. Even though the performance is lowered for the testability-enhanced *Diffeq* circuits synthesized in this experiment, the implementation of the proposed technique requires virtually no area overhead. Thus, designers may take further optimization steps to improve the performance of the circuit without significant impact on circuit area.

While *GCD* and *Diffeq* each contain a single loop only, *Barcode* contains two nested loops. The inner loop makes at least 255 iterations each time it is invoked. With one test pin of type T1 controlling the exit condition of the inner loop, the system is able to return the *locus* to the outer loop prior to completion of the iterations. As shown in Table III, the fault coverage is improved from 59% to 74%, and the number of aborted faults is greatly reduced. Adding a control point of type T2 allowing the *locus* to stay in the inner loop after the 255 iterations would not be helpful, since reaching the end of 255 iterations is already difficult for a gate-level ATPG. However, using two control points of type T3 to control two different decision nodes through exclusive-OR functions is effective. Furthermore, Barcode contains an embedded loop counter, and embedded counters are notorious for making any circuit hard to test at the gate level. Therefore, it may be necessary to make the loop counter directly controllable to get a high fault coverage. Thus, a control point of type T4 was added to load the counter directly from existing primary inputs. The two test pins added as type-T3 control points were also used as type-T4 control points to minimize the number of test pins. Significant improvements in fault coverage resulted, at the cost of an estimated 7.5% increase in area.

Unlike *Barcode*, which has two nested loops, *DHRC* has two cascaded loops. The first loop initializes the memories in the design; then the second loop performs the computations. The high-level description of *DHRC* is a modified version of the *DHRC* benchmark, because the original description is incomplete. The problem size has been reduced so that logic synthesis can be done within a reasonable period of time. The design without testability enhancement is fairly testable. Addition of control points of types T1 and T2 results in fewer aborted faults, shorter test generation time, and higher fault coverage for the deterministic ATPG. Use of two control points of type T3 (which required only one test pin) combined with a control point of type T4 (to load the counter directly from existing primary inputs) also resulted in higher fault coverage.

V Conclusions

We have presented an SFT approach to add controllability to HTC loops in a system. The HTC loops are identified using a new high-level testability measure that evaluates the controllability of conditional branches in a high-level circuit description. Implementation of this approach does not utilize any scan design, and modifications are done at the high level only. Logic synthesis can be performed by any high-level synthesis system. Experimental results on several high-level synthesis benchmarks show that when this approach is used, the circuits generated often require shorter ATPG times to produce test sets that achieve better fault coverage and fault efficiency. The test vectors can be applied at clock-speed, and testability is improved at the cost of one or two extra input pins, while area and performance overheads are minimal.

References

[1] L. J. Avra and E. J. McCluskey, "High-level synthesis of testable designs: An overview of university systems," *Proc. Test Synthesis Seminar, Int. Test Conf.*, TS Paper 1.1, 1994.

[2] S. Chiu and C. A. Papachristou, "A partial scan cost estimation method at the system level," *Proc. IEEE Int. Conf. Computer Design*, pp. 146–150, 1993.

[3] T. C. Lee, N. K. Jha, and W. H. Wolf, "A conditional resource sharing method for behavioral synthesis of highly testable data paths," *Proc. Int. Test Conf.*, pp. 744–753, 1993.

[4] M. Potkonjak, S. Dey, and R. K. Roy, "Behavioral synthesis of area-efficient testable designs using interaction between hardware sharing and partial scan," *IEEE Trans. Computer-Aided Design*, vol. 14, no. 9, pp. 1141–1154, Sept. 1995.

[5] T. Thomas, P. Vishakantantaiah, and J. A. Abraham, "Impact of behavioral modifications for testability," *Proc. IEEE VLSI Test Symp.*, pp. 427–432, 1994.

[6] V. Chickermane, J. Lee, and J. H. Patel, "A comparative study of design for testability methods using high-level and gate-level descriptions," *Proc. Int. Conf. Computer-Aided Design*, pp. 620–624, 1992.

[7] S. Bhattacharya, F. Brglez, and S. Dey, "Transformations and resynthesis for testability of RT-level control-data path specifications," *IEEE Trans. VLSI Systems*, vol. 1, no. 3, pp. 304–318, Sept. 1993.

[8] H. Harmanani and C. Papachristou, "An improved method for RTL synthesis with testability tradeoffs," *Proc. Int. Conf. Computer-Aided Design*, pp. 30–35, 1993.

[9] S. Dey, M. Potkonjak, and R. Roy, "Exploiting hardware sharing in high-level synthesis for partial scan optimization," *Proc. Int. Conf. Computer-Aided Design*, pp. 20–25, 1993.

[10] T. C. Lee, W. H. Wolf, and N. K. Jha, "Behavioral synthesis for easy testability in data paths scheduling," *Proc. Int. Conf. Computer-Aided Design*, pp. 616–619, 1992.

[11] B. Underwood, W. Law, S. Kang, and H. Konuk, "Fastpath: A path-delay test generator for standard scan designs," *Proc. Int. Test Conf.*, pp. 154–163, 1994.

[12] P. Varma, "On path delay testing in a standard scan environment," *Proc. Int. Test Conf.*, pp. 164–173, 1994.

[13] P. C. Maxwell, R. C. Aitken, V. Johansen, and I. Chiang, "The effect of different test sets on quality level prediction: when is 80% better than 90%?," *Proc. Int. Test Conf.*, pp. 358–364, 1991.

[14] S. Dey and M. Potkonjak, "Non-scan design-for-testability of RT-level data paths," *Proc. Int. Conf. Computer-Aided Design*, pp. 640–645, 1994.

[15] C.-H. Chen, C. Wu, and D. G. Saab, "BETA: Behavioral testability analysis," *Proc. Int. Conf. Computer-Aided Design*, pp. 202–205, 1991.

[16] C. Chen, T. Karnik, and D. G. Saab, "Structural and behavioral synthesis for testability techniques," *IEEE Trans. Computer-Aided Design*, vol. 13, no. 6, pp. 777–785, June 1994.

[17] F. Hsu and J. H. Patel, "A distance reduction approach to design for testability," *Proc. IEEE VLSI Test Symp.*, pp. 158–163, 1995.