other applications including SOI, MOS and Bipolar breakdown, insulated gate bipolar transistor (IGBT), etc.

In addition to greatly improving grid quality which solved the observed convergence problems, the new grid generation algorithm allows a drastic reduction in grid size without sacrificing solution accuracy. Since there is a superlinear dependence of CPU time on grid size (approximately $O(N^{1.5-1.75})$ in 2-D), the resulting speed up can be significant even in less critical cases where convergence problems are not observed with the larger original grid. In the shown example with the original grid size of 3700 and grid size after remesh of 1300, a speed up of more than 3–4 times per Newton iteration would be expected (total CPU time may vary if automatic biasing is utilized).

The discussed application example is an electrothermal breakdown simulation of a power MOS device, which could not be simulated using the original grid. The suggested remeshing strategy achieved an improvement in grid quality by 3–5 orders of magnitude and allowed an entire breakdown current-voltage curve to be generated in under 0.5 h on a workstation. Similar observations were made for an off-state avalanche breakdown of the same device.

Since the original advancing front mesh generation algorithm was first demonstrated in the work reported here, a more general and robust version of it has been implemented as pdMesh, a tool from PDF Solutions [10], [11]. Subsequently, it was also used by Technology Modeling Associates, Inc. (TMA) in the so-called adaptive boundary conforming (ABC) mesh [12] and most recently by Integrated Systems Engineering (ISE) in their MDRAW-ISE product [13].

ACKNOWLEDGMENT

The author would like to thank S. Verdonckt-Vandebroeck from Xerox Research who provided the initial motivation for this work as well as the power MOSFET structure where the convergence problems were more severe than usual. Many valuable discussions on this subject with colleagues at TMA are also greatly appreciated, in particular with J. G. Rollins and V. Boksha.

REFERENCES

[1] *TSUPREM-4 User's Manual*, Technology Modeling Associates, Palo Alto, CA, 1995.
[2] S. Selberherr, *Analysis and Simulation of Semiconductor Devices*. New York: Springer-Verlag, 1984.
[3] G. Strang and G. J. Fix, *An Analysis of the Finite Element Method*. Englewood Cliffs, NJ: Prentice-Hall, 1973.
[4] *PLTMG User's Guide 7.0*, R. E. Bank, Soc. Ind. Appl. Math. (SIAM), 1994.
[5] M. R. Pinto, "Comprehensive semiconductor device simulation for ULSI," Ph.D. dissertation, Stanford University, CA, 1990.
[6] S. Kumashiro and I. Yokota, *A Triangular Mesh Generation Method Suitable for the Analysis of Complex MOS Device Structures*, NUPAD V, Honolulu, 1994.
[7] V. Axelrad and R. Klein, "Electrothermal simulation of an IGBT," in ISPSD-92, Tokyo, Japan, 1992.
[8] *MEDICI User's Manual*, Technology Modeling Associates, Palo Alto, CA, 1995.
[9] U. Ascher, P. A. Markowich, C. Schmeiser, H. Steinrueck and R. Weiss, "Conditioning of the steady-state semiconductor device problem," Computer Sci., Univ. British Columbia, Vancouver, Can., Tech. Rep. 86-12, 1986.
[10] V. Axelrad, "pdMesh White Paper," PDF Solutions, San Jose, CA, Dec. 1996.
[11] V. Axelrad, G. Long, and P. Kuepper, *Elimination of Meshing Noise in Statistical TCAD*, in Int. Workshop Semiconductor Metrology (IWSM), Kyoto, Japan, June 1997.
[12] V. Moroz, S. Motzny, and K. Lilja, *A Boundary Conforming Mesh Generation Algorithm for Simulation of Devices with Complex Geometry*, in Simulation Semiconductor Devices Processes (SISDEP), Boston, Sept. 1997.
[13] Integrated Systems Engineering (ISI), "ISI Newsletter 97," Integrated Systems Eng. (ISE AG), Switzerland, Dec. 1997.

A Controller Redesign Technique to Enhance Testability of Controller-Data Path Circuits

Sujit Dey, Vijay Gangaram, and Miodrag Potkonjak

***Abstract*— We study the effect of the controller on the testability of sequential circuits composed of controllers and data paths. We show that even when all the loops of the circuit have been broken by using scan flip-flops (FF's) and the control and data path parts are individually 100%testable, the composite circuit may not be easily testable by gate-level sequential automatic test pattern generation (ATPG). Analysis shows that a primary problem in test pattern generation of combined controller-data path circuits is the correlation of control signals due to implications imposed by the controller specification. A design-for-testability (DFT) technique is developed to redesign the controller such that the implications which may produce conflicts during test pattern generation are eliminated. The DFT technique involves adding extra control vectors to the controller. Experimental results show the ability of the controller DFT technique to produce highly testable controller-data path circuits, with nominal hardware overhead.**

***Index Terms*— Data path, design for testability, controller, high-level synthesis, high-level testability.**

I. INTRODUCTION

Several existing scan-based design-for-testability (DFT) techniques use heuristics based on the topology of a circuit like breaking all loops, except self-loops, and reduction of sequential depth as ways to make sequential automatic test pattern generation (ATPG) of circuits easy [1]–[3]. However, it has been observed that for many circuits, even when all loops are broken using scan flip-flops (FF's) and the sequential depth is low, the circuit remains difficult for sequential ATPG. This paper introduces a new DFT technique to supplement topology-based DFT techniques like loop-breaking and sequential depth reduction. The proposed technique uses high-level

Manuscript received February 23, 1996. This paper was recommended by Associate Editor S. Reddy.

S. Dey was with C & C Research Laboratories, NEC, Princeton, NJ 08540 USA. He is now with the Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093 USA (e-mail: dey@ece.ucsd.edu).

V. Gangaram was with C & C Research Laboratories, NEC, Princeton, NJ 08540 USA. He is now with Design Technologies of Intel Corporation, Folsom, CA 95630 USA.

M. Potkonjak was with C & C Research Laboratories, NEC, Princeton, NJ 08540 USA. He is now with the Computer Science Department, University of California, Los Angeles, CA 90095 USA.

Publisher Item Identifier S 0278-0070(98)04511-4.

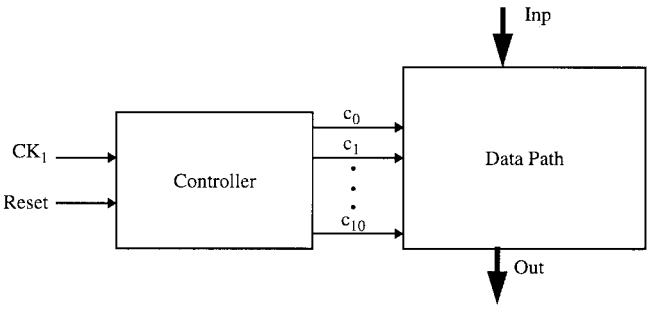


Fig. 1. Block diagram of an example circuit IIR filter showing controller-data path parts.

design information, both topological and functional. In particular, the effect of the controller on the data path of the circuit is investigated, and a controller redesign technique is proposed to make a sequential circuit consisting of controller-data path easily testable.

A. High-Level DFT Techniques

Recently, several high level design and synthesis approaches have been proposed to generate easily testable data paths for both built-in-self-test (BIST)-based testing methodology [4]–[7] and ATPG methods [8]–[14]. These techniques either modify the behavioral description of a design to improve the testability of the resulting circuit [8], [11], [13] or consider testability as one of the design objectives during the behavioral synthesis process [9], [10], [12].

All the existing BIST-based as well as ATPG-based high level synthesis/design for testability techniques consider the testability of the data paths only. Either the existing techniques ignore the controller or assume that the control signals coming to the data path are independently controllable. An exception is Genesis [15], but that system focuses on hierarchical test generation and not gate-level sequential ATPG, which is the focus of this work. In actuality, the control signals to the data path come from the controller and they are not independently controllable. In actuality, the control signals to the data path come from the controller and they are not independently controllable.

Several DFT techniques have been developed which use RT-level design information [16]–[19]. While the first two techniques focus exclusively on data path circuits, the third technique focuses on making controller-data path circuits testable for combinational ATPG using full scan. A register-transfer (RT)-level DFT technique which considers the presence of the controller along with the data path and which targets sequential ATPG was presented in [19]. It was shown that the use of RT-level information to select scan FF's results in significantly better performance when compared to techniques limited to gate-level information only. The technique addressed two types of circuits. For instruction-set processors, scan selection techniques were developed for both controller and data path. However, a structural description of the controller (its implementation) is required for selecting scan FF's in the controller. For data-dominated circuits like DSP filters, the technique assumed that the data path does not have any controller. This assumption is valid when hardware sharing and time sharing are not performed while synthesizing the data path. However, for various applications, data paths are synthesized using hardware/time sharing, and hence have controllers implementing the schedule of the operations of the design. A comprehensive survey of existing behavioral and RT-level test synthesis techniques can be found in [20].

The idea of augmenting the controller to enhance the testability of a circuit has been researched before [21], [22]. However, no controller DFT technique has been proposed before to enhance the testability of controller-data path designs based on an investigation into the effect of the controller on the testability of the data path.

B. The Proposed Approach

Even when a gate-level DFT technique like [1]–[3], or a high-level DFT technique like [19], makes both the controller and the data path individually highly testable, a high test efficiency may not be achieved by sequential ATPG for the combined controller-data path circuit. For instance, when a fully testable data path generated by a behavioral test synthesis system (BETS) [12] interacts with its corresponding controller, which is also made fully testable by using scan, the testability of the combined circuit may deteriorate significantly. In this paper, we address the problem of making a circuit composed of a controller and a data path easy-to-test for sequential ATPG, unlike the existing high-level techniques which focus exclusively on data paths. Also, unlike the technique presented in [19], our DFT technique does not involve the use of scan and is applicable to application-specific data paths which have controllers.

We study the effect of the controller on the testability of the data path. We observe that because of the correlation of the control signals due to implications imposed by the controller specification, the complete circuit may not have high fault coverage even when both the controller and data path are individually highly testable. We develop techniques to identify the control signal implications which may create conflicts during sequential ATPG. A technique is developed to redesign the controller such that the identified implications are eliminated. The technique involves adding a few extra control vectors to the existing control vectors which are outputs of the controller. The controller DFT technique has been applied to several controller-data path circuits. Experimental results show the ability of the proposed DFT technique to produce highly testable controller-data path circuits with only marginal area overhead.

II. Motivation for Controller-Based DFT

Fig. 1 shows the block diagram of a typical circuit composed of a controller and a data path. The circuit implements an IIR filter, synthesized from its high level specification using a behavioral test synthesis tool BETS [12]. The data path implements the operations of the design, while the controller implements the schedule of the operations. In the case of the example design, the controller has two inputs, Reset and CK1, while it has 11 outputs, the control signals $\{c_0, \cdots, c_{10}\}$. The control signals are inputs to the data path and are used to control the various units of the data path. The data path also has an n-bit data input and n-bit data output.

The data path, shown in Fig. 2, has an RT level structural specification consisting of execution units (like adders, multipliers, ALU's, transfer units), registers, multiplexors, and interconnects. Each interconnect shown in the data path is n-bit wide, where n is the word size of the design.

The VHDL specification of the functionality of the corresponding controller is shown in Fig. 3. For convenience, the state transition graph (STG) of the controller is shown in Fig. 4(a). Note that the transitions from each state to state s_0 on the reset input are not shown in Fig. 4(a) for clarity. The control vectors $\{\mathrm{CV}_0, \cdots, \mathrm{CV}_5\}$ representing the outputs of the controller are expressed in terms of the output (control) signals $\{c_0, \cdots, c_{10}\}$, as shown in Fig. 4(b). Note that the controller has a functional specification, as opposed to the structural specification of the data path, since the controller has not yet been implemented. Throughout the paper, unless otherwise stated, the data path and controller of the fourth order IIR filter shown in Figs. 2–4 are used to illustrate the new DFT technique.

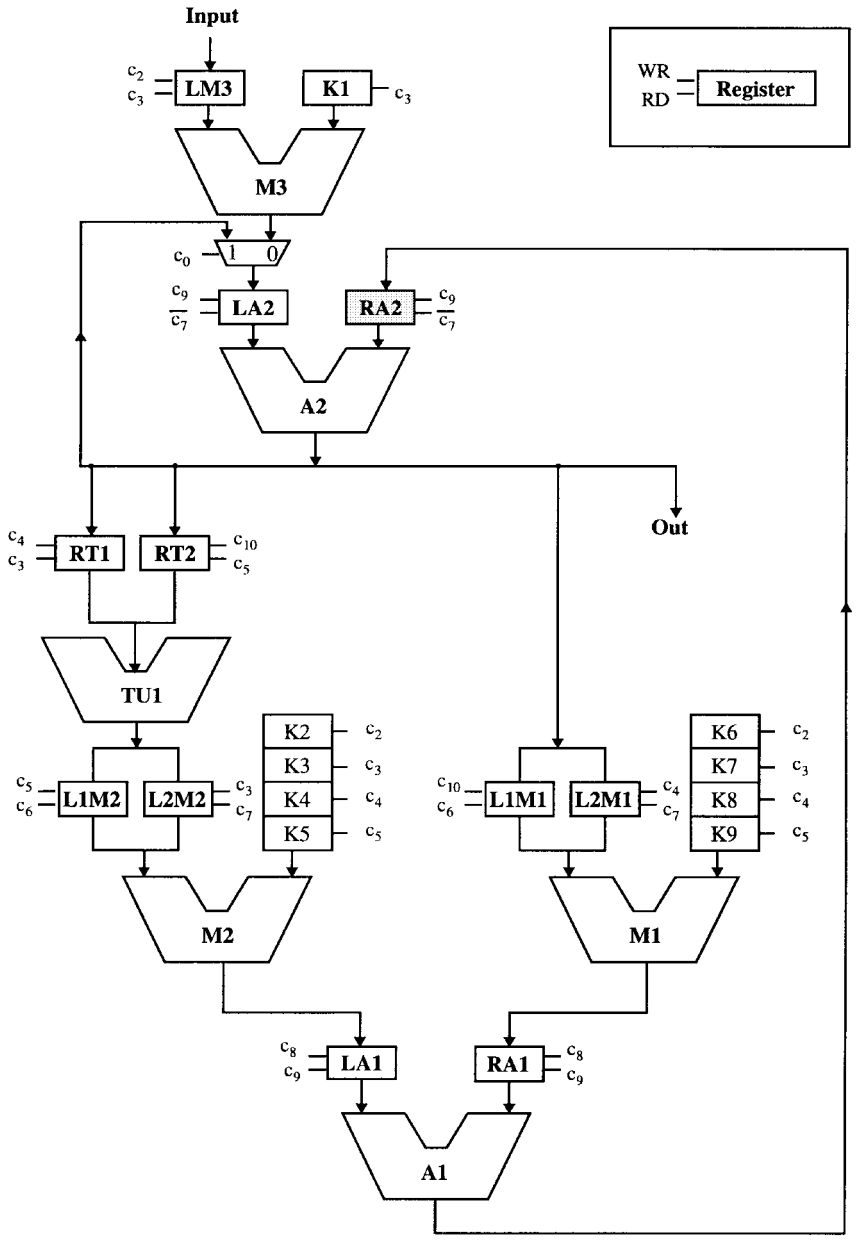


Fig. 2. Data path of example circuit IIR filter.

Without the use of any DFT, the original data path and controller, individually and combined, are not easily testable, as shown by the rows under *Original* in Table II. The data path has been synthesized using a behavioral synthesis-for-testability tool BETS [12] such that all loops, except self-loops, pass through the register RA2 and scanning RA2 breaks all loops. Assuming full and independent controllability of the control signals, as is done by the existing high level synthesis-for-testability techniques including BETS [12], the data path with the partial scan register RA2 is easily testable, as indicated by the row *Scan DFT - Datapath* in Table II. The sequential ATPG program HITEC [23] can achieve almost 95% fault coverage and almost 100% test efficiency on the scan data path. Similarly, after scanning all the FF's of the controller, the controller is highly testable, with both the fault coverage as well as the test efficiency being 100%. Note that fault coverage gives the percentage of faults for which a test could be found, while test efficiency gives the percentage of faults either for which a test could be found or which could be identified as redundant (that is, the percentage of faults not aborted).

However, in real life, the control signals to the data path are not independently controllable, as is assumed by DFT techniques to synthesize testable data paths. When the controller and the data path are actually put together, the independence assumption of the control signals is no more valid, and the fault coverage and test efficiency achieved for the full circuit are only about 77% and 84%, respectively, as shown in the row *Scan DFT - Contr + DP* in Table II. Note that the low fault coverage and test efficiency of the composite controller-data path circuit is in spite of all the nonself loops in the circuit broken by scanning register RA2 and the controller FF's.

A. Conflicts Due to Implication of Control Signals

When the controller in Fig. 4 is considered together with the data path in Fig. 2, conflicts may arise in justification and propagation of values during test pattern generation. These conflicts stem from the correlation of the control signals imposed by the controller specification. Note that when the data path is considered independently,

```
ARCHITECTURE behavior OF controller IS
BEGIN
   PROCESS(CK1,RESET)
    VARIABLE CurrentState : INTEGER := 0;
    VARIABLE NextState : INTEGER := 0;
    CONSTANT CV0 : std_logic_vector := ('1','0','1','0','0','0','0','1','1','0','0');
    CONSTANT CV1 : std_logic_vector := ('0','0','0','1','0','0','0','1','1','1','0');
    CONSTANT CV2 : std_logic_vector := ('1','0','0','0','1','0','1','0','1','1','0');
    CONSTANT CV3 : std_logic_vector := ('1','0','0','0','0','1','1','0','1','1','0');
    CONSTANT CV4 : std_logic_vector := ('1','0','0','0','0','0','0','0','0','1','1');
    CONSTANT CV5 : std_logic_vector := ('1','1','0','0','0','0','0','0','0','0','0');
   BEGIN
      IF RESET = '1' THEN
         CurrentState := 0;
      ELSIF CK1 = '1' THEN
         CurrentState := NextState;
      END IF;
      CASE CurrentState IS
         WHEN 0 => ControlOut <= CV0;
         WHEN 1 => ControlOut <= CV1;
         WHEN 2 => ControlOut <= CV2;
         WHEN 3 => ControlOut <= CV3;
         WHEN 4 => ControlOut <= CV4;
         WHEN 5 => ControlOut <= CV5;
         WHEN OTHERS => ControlOut <= CV0;
      END CASE;
      CASE CurrentState IS
         WHEN 0 => NextState := 1;
         WHEN 1 => NextState := 2;
         WHEN 2 => NextState := 3;
         WHEN 3 => NextState := 4;
         WHEN 4 => NextState := 5;
         WHEN 5 => NextState := 0;
         WHEN OTHERS => NextState := 0;
      END CASE;
      IF RESET = '1' THEN
         NextState := 0;
      END IF;
   END PROCESS;
END behavior;
```

Fig. 3. VHDL specification of the controller of IIR filter data path.

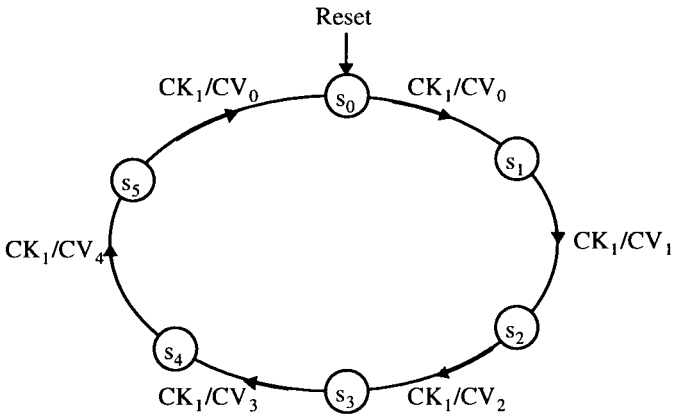
TABLE I
CONTROL SIGNAL CONFLICT DURING JUSTIFICATION

TF	Desired Action	Required Control Signal
0	LM3 ← x (Inp)	$(c_2 = 1)$
1	LA2 ← x, RA2 ← 0, LM3 ← y (Inp)	$(c_3 = 1, c_0 = 0, c_9 = 1), (c_2 = 1)$
2	Out(A2) ← x, LA2 ← y, RA2 ← 0	$(\overline{c_7} = 1), (c_0 = 0, c_9 = 1)$

these conflicts do not arise due to the independence assumption of the control signals, leading to easy testability of the data path.

Let us consider justification of a value at the output of the execution unit A1. Let us say that in the next time frame, two different values will be required to be justified at the two registers LA1 and RA1. There exists a reconvergence from the output of A2 to the output of A1, leading to a possible conflict. However, since the paths from Out(A2) to LA1 pass through two registers, while the paths from Out(A2) to RA1 pass through one register, the fanout conflict can be avoided by requiring that in two successive time frames, two different values x and y are generated at the output of A2. Let us also assume that the justification objective is chosen such that $x > y$ and the constant $k_1 = 1$.

Table I shows the desired actions (justification objectives) and the required control signals needed to fulfill the desired actions to justify two different values x and y, $x > y$, in two successive time frames at the output of the adder unit A2. The time frames associated with the desired actions are also shown in the column *TF*. In time frame 0, the register LM3 has to be loaded with value x from primary input Inp, requiring that the WRITE signal of register LM3 $c_2 = 1$. In the next time frame 1, register LA2 has to be loaded with x and scan register RA2 loaded with 0, requiring that the READ signal of register LM3, $c_3 = 1$, the control signal to the multiplexor $c_0 = 0$, and the WRITE signal to the registers LA2 and RA2, $c_9 = 1$. At the same time, register LM3 has to be loaded with y from Inp, requiring that the READ signal $c_2 = 1$. In next time frame 2, the output of A2 is made x by reading the registers LA2 and RA2, which store x and 0, respectively, and adding them. This requires that the READ signal of register LA2 $\overline{c_7} = 1$. Simultaneously, register LA2 has to be loaded with y (from LM3) and register RA2 with 0, requiring that the multiplexor signal and the LA2 WRITE signals $c_0 = 0$ and $c_9 = 1$.



(a)

	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c8	c_9	c_{10}
CV_0	1	0	1	0	0	0	0	1	1	0	0
CV_1	0	0	0	1	0	0	0	1	1	1	0
CV_2	1	0	0	0	1	0	1	0	1	1	0
CV_3	1	0	0	0	0	1	1	0	1	1	0
CV_4	1	0	0	0	0	0	0	0	0	1	1
CV_5	1	1	0	0	0	0	0	0	0	0	0

(b)

Fig. 4. Controller of the IIR filter: (a) state transition graph and (b) control vectors composed of control signals.

However, due to the design requirements, the controller has been specified and synthesized such that whenever the control signal $c_7 = 0$, $c_0 = 1$, as seen from the control vectors specification in Fig. 4(b). Consequently, $\overline{c_7} = 1$ and $c_0 = 0$ cannot be satisfied simultaneously in the same clock cycle, leading to a justification conflict. The result is that in time frame 2, the multiplexor control signal $c_0 = 1$; hence, register LA2 cannot be loaded with the desired value y, but instead has to be loaded with the output of A2 x. Consequently, in time frame 3, the value y, $y < x$ cannot be produced at the output of A2. Instead, Out(A2) will be z, where $z = x + RA2$, hence $z > x$.

The implication $(\overline{c_7} = 1) \Rightarrow (c_0 = 1)$ denoted by $\overline{c_7} \Rightarrow c_0$ imposes a correlation between the control signals c_7 and c_0, instead of the assumption of independent controllability of control signals assumed while doing ATPG for the data path alone. In general, two control signals (c_i, c_j) are said to be *correlated* if there exists an implication $c_i \Rightarrow c_j$ in the controller specification. As illustrated in this section, the control signal implications and the consequent correlations between corresponding control signals are responsible in producing conflicts during test pattern generation, leading to poor fault coverage and test efficiency even when scan DFT has broken all loops in the circuit and the individual controller and data path parts are highly testable.

B. Redesigning the Controller to Improve Testability

The correlations imposed by the implications of the control signals can be broken by adding extra control vectors to the controller. To preserve the behavior of the circuit in the functional mode, the extra control vectors should be enabled only in the test mode. Fig. 5 shows the VHDL specification of the redesigned controller derived from the original controller specification in Fig. 3. Due to space constraints, only parts of the controller which have been modified are being shown in Fig. 5. Two new control vectors, TCV_0 and TCV_1, have been added. The output functions of the controller, when in state 0 and 1, have been changed. In state 0, if $Test = '1'$, the output of the controller is the vector TCV_0, else it remains CV_0. Similar change has been effected in state 1. The state transition graph and a table of control vectors in terms of the control signals are shown in Fig. 6(a) and (b), respectively. Note that the "Test" pin is set to 0 in the functional mode, hence the controller modification preserves the functionality of the circuit.

Adding the two test control vectors eliminates a number of control signal implications that would most likely produce conflicts during test pattern generation. For instance, the control vector TCV_0 has been designed such that it breaks the implication $\overline{c_7} \Rightarrow c_0$, besides breaking other implications that will be discussed later. The implication is broken by having $c_7 = 0$ and $c_0 = 0$ simultaneously in the same vector. When the redesigned controller is used instead of the original controller, the control signal conflict shown by Table I in Section II-A is avoided. This is because now $(\overline{c_7} = 1)$ and $(c_0 = 0)$ can be simultaneously satisfied as required by the justification sequence in time frame 2.

After the controller DFT, again all the FF's of the controller are scanned. The data path remains the same, with register RA2 scanned. The composite controller-data path circuit becomes significantly more testable than the composite circuit with the same scan FF's but no controller DFT. The last row *Contr + DP* in Table II shows the results of running HITEC on the circuit containing the controller with controller DFT. Very high fault coverage (93.6%) and test efficiency (99.7%) can be achieved for the circuit derived by controller DFT + scan DFT, as opposed to 77.2% fault coverage and 83.7% test efficiency achieved for the controller-data path circuit derived by just scan DFT. Note that the only difference between the two circuits is the controller DFT of adding two extra test control vectors. The controller DFT increases the number of cells from 3448 to 3502, which represents a marginal area overhead of 1.57%.

In the next two sections, we describe the two phases of the controller-based DFT technique. The first phase identifies which control signal implications/correlations should be eliminated. The second phase derives a minimal set of control vectors to eliminate the identified implications/correlations. The control vectors are added to the controller so that they can be activated only in the test mode, thus preserving the functionality of the design.

III. Identifying Which Control Signal Implications to Eliminate

As explained in the previous section, control signal implications may cause conflicts during test pattern generation, even in the absence of topological features which have been traditionally identified to be problematic for ATPG, like reconvergences, loops, and sequential depth. In this section, we first show how control signal implications can be extracted from the given controller specification. Since there are typically a large number of such implications, trying to eliminate all of them can be very costly. We show how to identify a small subset of the implications, termed inhibiting implications, that need to be eliminated to facilitate easy sequential ATPG.

A. Control Signal Implications

Given the controller specification, we first form the control vector table shown in Fig. 4(b). Next, we extract all the implications of each control signal c_i and $\overline{c_i}$. Consider the controller specification and the corresponding control vector table shown in Figs. 3 and 4(b),

```
ARCHITECTURE behavior OF controller IS
BEGIN
   PROCESS(CK1,RESET,Test)
    VARIABLE CurrentState : INTEGER := 0;
    VARIABLE NextState : INTEGER := 0;
    CONSTANT CV0  : std_logic_vector := ('1','0','1','0','0','0','0','1','1','0','0');
    CONSTANT CV1  : std_logic_vector := ('0','0','0','1','0','0','0','1','1','1','0');
    CONSTANT CV2  : std_logic_vector := ('1','0','0','0','1','0','1','0','1','1','0');
    CONSTANT CV3  : std_logic_vector := ('1','0','0','0','0','1','1','0','1','1','0');
    CONSTANT CV4  : std_logic_vector := ('1','0','0','0','0','0','0','0','0','1','1');
    CONSTANT CV5  : std_logic_vector := ('1','1','0','0','0','0','0','0','0','0','0');


CONSTANT TCV0 : std_logic_vector := ('0','0','1','1','0','0','1','0','1','1','0');
    CONSTANT TCV1 : std_logic_vector := ('1','0','0','1','1','1','1','1','1','1','1');

   BEGIN
      IF RESET = '1' THEN
         CurrentState := 0;
      ELSIF CK1 = '1' THEN
         CurrentState := NextState;
      END IF;
      CASE CurrentState IS

         WHEN 0 => IF Test = '1' THEN
                      ControlOut <= TCV0;
                   ELSE
                      ControlOut <= CV0;
         WHEN 1 => IF Test = '1' THEN
                      ControlOut <= TCV1;
                   ELSE
                      ControlOut <= CV1;

         WHEN 2 => ControlOut <= CV2;
         WHEN 3 => ControlOut <= CV3;
         WHEN 4 => ControlOut <= CV4;
         WHEN 5 => ControlOut <= CV5;
         WHEN OTHERS => ControlOut <= CV0;
      END CASE;
      .
      .
      .
```

Fig. 5. VHDL specification of the controller after DFT.

respectively. Let us find the implications of the control signal c_7 and $\overline{c_7}$. $(c_7 = 0)$ implies $(c_0 = 1)$, $(c_2 = 0)$, and $(c_3 = 0)$. Hence, $\overline{c_7} \Rightarrow c_0, \overline{c}_2, \overline{c}_3$. Similarly, $c_7 \Rightarrow \overline{c}_1, \overline{c}_4, \overline{c}_5, \overline{c}_6, c_8, \overline{c}_{10}$.

The implications of the other control signals are as follows:

$$\overline{c}_0 \Rightarrow \overline{c}_1, \overline{c}_2, c_3, \overline{c}_4, \overline{c}_5, \overline{c}_6, c_7, c_8, c_9, \overline{c}_{10}$$
$$c_2 \Rightarrow c_0, \overline{c}_1, \overline{c}_3, \overline{c}_4, \overline{c}_5, \overline{c}_6, c_7, c_8, \overline{c}_9, \overline{c}_{10}$$
$$c_3 \Rightarrow \overline{c}_0, \overline{c}_1, \overline{c}_2, \overline{c}_4, \overline{c}_5, \overline{c}_6, \overline{c}_7, \overline{c}_8, \overline{c}_9, \overline{c}_{10}$$
$$c_4 \Rightarrow c_0, \overline{c}_1, \overline{c}_2, \overline{c}_3, \overline{c}_5, c_6, \overline{c}_7, c_8, c_9, \overline{c}_{10}$$
$$c_5 \Rightarrow c_0, \overline{c}_1, \overline{c}_2, \overline{c}_3, \overline{c}_4, c_6, \overline{c}_7, c_8, c_9, \overline{c}_{10}$$
$$c_6 \Rightarrow c_0, \overline{c}_1, \overline{c}_2, \overline{c}_3, \overline{c}_7, c_8, c_9, \overline{c}_{10}$$
$$c_8 \Rightarrow \overline{c}_1, \overline{c}_{10}$$
$$c_9 \Rightarrow \overline{c}_1, \overline{c}_2$$
$$c_{10} \Rightarrow c_0, \overline{c}_1, \overline{c}_2, \overline{c}_3, \overline{c}_4, \overline{c}_5, \overline{c}_6, \overline{c}_7, \overline{c}_8, c_9.$$

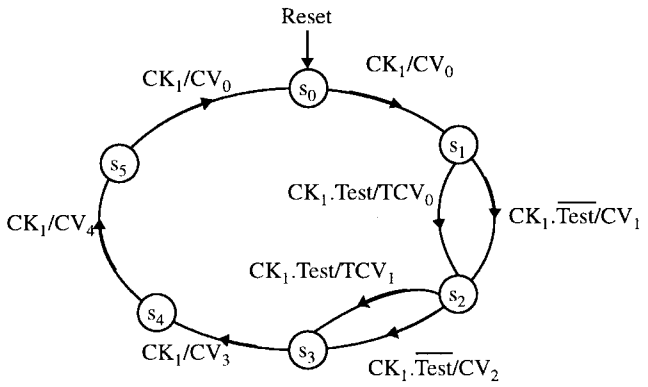
The implications of each control signal are stored as an implication graph, shown in Fig. 7(b) for the control signal $\overline{c_7}$.

B. Topological Relations of Control Signals in Data Path

Since the number of control signal implications is usually very large, a large number of control vectors may have to be added to eliminate all the implications, making the controller very expensive as well as difficult to test. Using structural information about how control signals control the different modules in the data path, we derive information regarding which pairs of control signals can affect the flow of data in k time frames for an user specified k. We will eventually consider eliminating implications between those pairs of control signals which affect data flow directly over k time frames, and hence can lead to conflicts during ATPG.

Each control signal c_i controls the flow of data through one or more modules in the data path like a multiplexor and/or a register. We say that control signal c_j is *k time frame module dependent* on control signal c_i if a module controlled by c_j has a path of no more than k time frames from a module controlled by c_i. For each control signal c_i, we construct a k-time frame module dependency graph as follows. Initially, the graph has only one node c_i. A node c_j and a directed edge from node c_i to node c_j are added to the graph if control signal c_j is k time-frame module dependent on control signal c_i. This is determined by traversing the data path from all modules controlled by c_i and including the corresponding control signals of all modules reached in k time frames.

The 2-time frame module dependency graph for the control signal $\overline{c_7}$ is shown in Fig. 7(a). For each edge $(\overline{c_7}, c_j)$ in the graph, there is



(a)

	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
CV_0	1	0	1	0	0	0	0	1	1	0	0
CV_1	0	0	0	1	0	0	0	1	1	1	0
CV_2	1	0	0	0	1	0	1	0	1	1	0
CV_3	1	0	0	0	0	1	1	0	1	1	0
CV_4	1	0	0	0	0	0	0	0	0	1	1
CV_5	1	1	0	0	0	0	0	0	0	0	0
TCV_0	0	0	1	1	0	0	1	0	1	1	0
TCV_1	1	0	0	1	1	1	1	1	1	1	1

(b)

Fig. 6. The controller ***after DFT***: (a) state transition graph and (b) control vectors.

a path not going across more than two time frames from the register controlled by $\overline{c_7}$ to a module controlled by signal c_j. For instance, there is a path of 1 time frame from LA2 controlled by $\overline{c_7}$ and RT1 controlled by c_4, hence the edge $(\overline{c_7}, c_4)$ in Fig. 7(a).

C. Forming the Dependency-Implication Graphs

For a control signal c_i given its k-time frame module dependency graph and its control implication graph, a new graph, the *dependency-implication graph*, is formed which consist of those edges which are common to both the k-time-frame module dependency graph and the control implications graph of signal c_i. The formation of the dependency-implication graph for control signal $\overline{c_7}$ is shown in Fig. 7(c). Each edge in the dependency-implication graph (c_i, c_j) correspond to an implication $c_i \Rightarrow c_j$ which exists due to the controller specification, and may affect data flow in the circuit since the module controlled by c_j is dependent on module controlled by c_i. Hence, each implication in the dependency-implication graph can potentially cause a conflict during ATPG, and hence is a candidate for removal by the DFT technique described later. The dependency-implication graphs of the control signals of the controller in Fig. 4 are shown in Fig. 8.

D. Identifying Inhibiting Implications to Eliminate

Even after the dependency-implication graphs have been formed for each control signal, there are many implications to possibly break. However, some of the implications in the dependency-implication graphs should remain as eliminating them will not help in justification/propagation of values. As an example, consider the implication $(c_3 \Rightarrow \overline{c}_0)$ in the dependency-implication graph of c_3 in Fig. 8. The consequence of the implication on the data path is that when in a particular clock cycle, the value of register LM3 is read, the result of the multiplication in M3 is passed on through the multiplexor to be stored in LA2. This implication will not lead to any conflict during the justification/propagation of values, and hence is not needed to be broken to reduce conflicts during ATPG.

On the other hand, some implications may potentially cause conflicts during ATPG, like the implication $(\overline{c_7} \Rightarrow c_0)$ shown in Section II-A. We use a heuristic to identify implications that may cause conflicts. We say an implication is an *inhibiting implication* if it prevents a desired action from taking place. We choose to eliminate only the inhibiting implications in the dependency-implication graphs.

Consider the dependency-implication graph shown in Fig. 8. The implication $(\overline{c_7} \Rightarrow c_0)$ is inhibiting because it prevents data from M3 to go through the multiplexor to LA2 in the same time frame that data is read from LA2. Similarly, the implication $(c_2 \Rightarrow \overline{c}_9)$ prevents data to be written to register LA2 in the same time frame that data is being written to LM3. The implication $(c_3 \Rightarrow \overline{c}_6)$ prevents data from being read from register L1M2 in the same time frame as data is read from register RT1. The above implications which restrict/prevent data flow are termed inhibiting implications. The inhibiting implications for the example controller-data path circuit of the IIR filter are shown marked in the dependency-implication graphs in Fig. 8.

The restrictions imposed by the inhibiting implications are due to the design functionality. However, they will reduce concurrency during ATPG and may lead to conflicts. Hence, each inhibiting implication is selected for removal during the next step of the DFT technique.

IV. Eliminating Control Implications by Adding Test Control Vectors

The second phase of the proposed DFT technique is to add control vectors to the controller specification such that the inhibiting implications identified in the first phase are eliminated. In this section, we show how to derive such control vectors, termed test control vectors (TCV's), as they will be generated by the controller only in the test mode. We provide a technique to merge the test control vectors to minimize the number of test control vectors needed so as to minimize the DFT overhead. We outline the process of augmenting the controller with the test control vectors such that the functionality of the controller, and hence the whole design is preserved.

A. Creating Test Control Vectors

For each control signal which has inhibiting implications that need to be eliminated, a test control vector is created to eliminate the inhibiting implications. An implication $(c_i = x) \Rightarrow (c_j = y)$, $x, y \in \{0, 1\}$ can be eliminated by adding a new control vector which has $c_i = x$ and $c_j = \overline{y}$. In general, for all the inhibiting implications of control signal c_i, $\{(c_i = x) \Rightarrow (c_j = y), \cdots, (c_k = z)\}$, a test control vector is formed with $c_i = x$ and each implied signal $c_j, \cdots, c_k$ having the complement of the value implied, that is, $(c_j = \overline{y}), \cdots, (c_k = \overline{z})$.

Consider the inhibiting implications from control signal c_2, denoted by dashed edges in Fig. 8. The implications $c_2 \Rightarrow c_0, \overline{c}_3, \overline{c}_9$, due to the existing control vectors, can be eliminated by adding a new control vector T_1: $c_2 = 1$, $c_0 = 0$, $c_3 = 1$, $c_9 = 1$ to the controller. Similarly, a new test control vector T_2: $c_3 = 1$, $c_6 = 1$, $c_7 = 0$ will eliminate the inhibiting implications from c_3. Fig. 9 lists the test control vectors $T_1, \cdots, T_6$ created to break the inhibiting implications marked in Fig. 8 for each control signal.

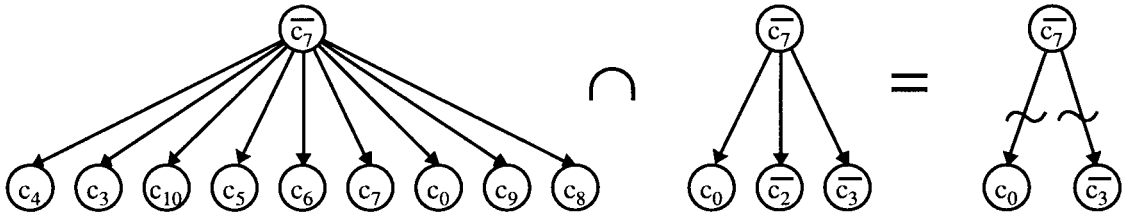


Fig. 7. Derivation of dependency-implication graph for control signal $\overline{c_7}$: (a) 2-time-frame module dependency graph for $\overline{c_7}$, (b) control signal implications of $\overline{c_7}$, and (c) dependency–implication graph for $\overline{c_7}$.

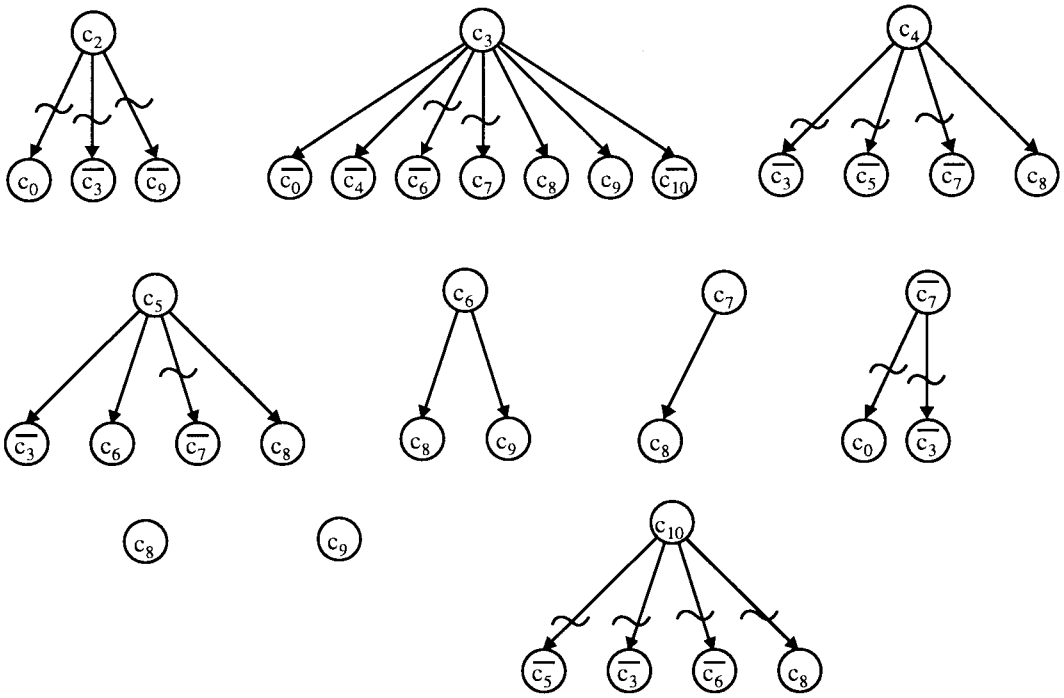


Fig. 8. Dependency-implication graphs for the control signals in Fig. 4. A dashed edge denotes an inhibiting implication.

To Break Implications from	TCV	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c8	c_9	c_{10}
c_2	T_1	0		1	1						1	
c_3	T_2				1			1	0			
c_4	T_3				1	1			1			
c_5	T_4						1		1			
$\overline{c_7}$	T_5	0			1				0			
c_{10}	T_6				1			1		1		1
$c_2, c_3, \overline{c_7}$	TCV_0	0	0	1	1	0	0	1	0	1	1	0
c_4, c_5, c_{10}	TCV_1	1	0	0	1	1	1	1	1	1	1	1

Fig. 9. Test control vectors to break inhibiting implications shown in Fig. 8.

B. Minimizing Number of Test Control Vectors

Two test control vectors are defined to be *compatible* if and only if, for each control signal c_k that is defined in both the control vectors, the value of c_k is identical in both the control vectors. A pair of compatible control vectors can be merged into a single vector while still eliminating the implications that each control vector was created to eliminate. For example, the vectors T_1 and T_2 in Fig. 9 are compatible because the only control signal existing in both T_1 and T_2, c_3 has the same value 1 in both the vectors. Similarly, the test control vector T_5 is compatible with both T_1 and T_2, and hence all three vectors can be merged to form a single test control vector TCV_0 shown in Fig. 9. Note that TCV_0 eliminates all the implications from c_2, c_3, $\overline{c}_7$ that the original TCV's T_1, T_2, and T_5 eliminate.

The problem of merging the test control vectors to produce a minimal number of test control vectors (akin to the test vectors compaction problem in ATPG) can be solved by mapping the problem to the well-known problem of graph clique partitioning. We begin by defining the clique partitioning problem. A graph is complete if and only if every pair of its nodes has an edge connecting them. A clique of a graph G is a complete subgraph of G. The problem of clique partitioning is to partition a graph into a minimal number of cliques such that each node belongs to exactly one clique. The clique partitioning problem is NP-Complete for partitioning into three or more cliques [24].

Given a set of TCV's, we construct a compatibility graph where each node corresponds to a test control vector, and there is an edge (T_i, T_j) if the two vectors T_i and T_j are compatible. A solution to the minimal clique partitioning problem of the compatibility graph gives a solution to the problem of finding a minimal set of test control vectors. Partitioning the compatibility graph into a minimal number of cliques $\{K_i\}$ and merging the test control vectors corresponding to nodes of clique K_i to form the test control vector TCV_i, results in a minimal set of test control vectors $\{\text{TCV}_i\}$ with the number of test control vectors equal to the number of cliques. Note that the number of TCV's to be minimized is bounded by the number of control signals, which is in the order of the number of modules (registers, multiplexors, execution units) in the data path. Hence, the number of nodes in the compatibility graph is usually small, and any heuristic algorithm like [25] or [26] can be used to solve the clique partitioning problem efficiently. In particular, we use the algorithm in [26] which has a running time complexity of $O(n^2)$ for a compatibility graph with n nodes and guarantees an optimal solution with a very high probability.

Fig. 10(a) shows the compatibility graph for the set of test control vectors $\{T_1, \cdots, T_6\}$ listed in Fig. 9. A possible solution to the minimal clique partitioning problem is illustrated in Fig. 10(b) which shows two cliques $K_0 = \{T_1, T_2, T_5\}$ and $K_1 = \{T_3, T_4, T_6\}$ covering all the nodes of the graph. Hence, the six TCV's $\{T_1, \cdots, T_6\}$ can be merged into only two TCV's: $\text{TCV}_0 = (T_1 \cup T_2 \cup T_5)$ and $\text{TCV}_1 = (T_3 \cup T_4 \cup T_6)$, as shown in Fig. 9.

C. Adding Control Vectors to Controller

After deriving the new test control vectors to eliminate the inhibiting implications, and hence the undesirable control signal correlations, the new vectors are added to the controller specification in a way that they are generated only in the test mode, hence preserving the behavior of the circuit. This is achieved by using a new "Test" pin which is set to zero in the functional mode. Also, to minimize area overhead, a test control vector TCV_i is added to a state with control vector CV_j such that TCV_i and CV_j have the minimum distance, that is the minimum number of control signals which have a value x in TCV_i and a different value $\overline{x}$ in CV_j. This would ensure the minimal change in minterms of the output function of any control signal c_k after the addition of the new test control vectors.

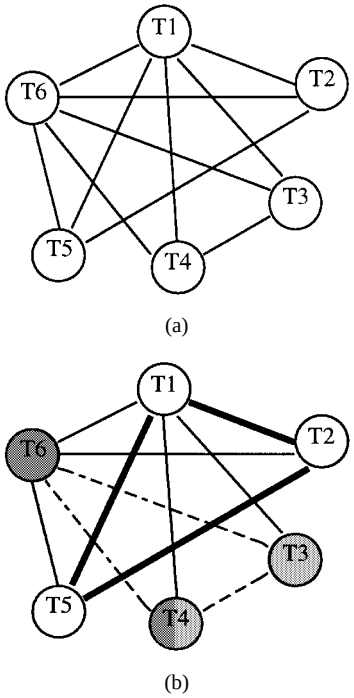


Fig. 10. (a) Compatibility graph of the test control vectors and (b) a two-clique partition.

Fig. 5 shows the VHDL specification of the redesigned controller derived from the original controller specification in Fig. 3. The two test control vectors TCV_0 and TCV_1 derived in the previous section are added as follows. TCV_0 has a minimum distance with CV_1 (= 3), and hence TCV_0 is added to state 1 with CV_1. If $Test = 1$, the output of the controller is the new vector TCV_0 else it remains CV_1. The test control vector TCV_1 is similarly added to state 2, since its distance with vector CV_2 is minimum (=4). The state transition graph and a table of control vectors are shown in Fig. 6(a) and (b), respectively. Since the "Test" pin is set to zero in the functional mode, the functionality of the circuit is preserved.

V. Steps of the DFT Algorithm

We summarize the steps of the proposed controller-based DFT algorithm. Each step has been described in detail in Sections III and IV.

A. Phase 1: Identify Inhibiting Implications

Input: Functional specification of controller and structural specification of data path.

Output: Set of inhibiting control signal implications.

1) Form the control vector table;
2) For each control signal {
3) Identify the control signal implications;
4) Form the k-time-frame module dependency graph, for user specified k;
5) Form the dependency–implication graph, DIG;
6) For each implication in DIG, identify if it is an inhibiting implication; }

TABLE II
4IIR.8: EFFECT OF SCAN-BASED AND CONTROLLER-BASED DFT SCHEMES

Circuit	DFT	Area	Faults		FC%	TE%	Vec	Tgen (secs)
			Total	Abt				
Original								
Controller	None	186	279	0	75.98	100.00	59	52.3
Datapath	None	3300	4572	1176	69.26	74.27	179	26589.2
Contr + DP	None	3448	4825	4257	4.83	*11.77*	94	>25.5 hrs
Scan DFT								
Controller	6 scan/6 FFs	186	279	0	100.00	100.00	68	0.7
Datapath	8 scan/88 FFs	3300	4572	12	94.72	99.73	399	1207.8
Contr + DP	14 scan/94 FFs	3448	4805	785	77.19	*83.66*	210	17917.0
Scan DFT + Controller DFT								
Contr + DP	14 scan/94 FFs + 2 TCV	3502	4846	14	93.60	*99.71*	504	933.8

TABLE III
4IIR.14: EFFECT OF SCAN-BASED AND CONTROLLER-BASED DFT SCHEMES

Circuit	DFT	Area	Faults		FC%	TE%	Vec	Tgen (secs)
			Total	Abt				
Original								
Controller	None	186	279	0	75.98	100.00	59	52.3
Datapath	None	6912	9767	910	86.76	90.68	271	22460.8
Contr + DP	None	7099	9996	8273	3.38	*8.88*	75	28760.1
Scan DFT								
Controller	6 scan/6 FFs	186	279	0	100.00	100.00	68	0.7
Datapath	14 scan/154 FFs	6912	9767	220	93.76	97.75	463	8433.8
Contr + DP	20 scan/160 FFs	7099	9996	1222	82.28	*87.78*	349	31890.9
Scan DFT + Controller DFT								
Contr + DP	20 scan/160 FFs + 2 TCV	7109	10027	42	94.43	*99.58*	807	3263.1

B. Phase 2: Eliminate Inhibiting Implications

Input: Set of inhibiting control signal implications.

Output: Modified functional specification of controller, with test control vectors added.

1) For each control signal{
2) Create a test control vector to eliminate all the inhibiting implications; }
3) Create compatibility graph of all test control vectors;
4) Partition the compatibility graph into a minimal number of cliques $\{K_i\}$;
5) Form test control vector TCV_i by merging the test control vectors corresponding to nodes of clique K_i;
6) Add the minimal set of test control vectors $\{\text{TCV}_i\}$ to the controller;
7) Add a test pin "Test" to the controller;
8) For each TCV from the set $\{\text{TCV}_i\}$ {
9) Identify the control vector CV_i with which TCV_i has minimum distance;
10) Modify controller such that if Test=1, controller outputs TCV_i, else CV_i; }

The proposed DFT technique identifies and eliminates inhibiting implications between pairs of control signals. In general, further improvements in the testability of a control-data path circuit may be achieved by identifying and eliminating simultaneous correlations between more than two signals. However, considering the simultaneous correlation of three or more signals can be computationally very expensive. As shown by the experimental results in the next section, considering only pairs of control signals is sufficient to achieve very high testability for most control-data path circuits.

VI. EXPERIMENTAL RESULTS

We applied the proposed controller-based DFT technique on several controller-data path designs, synthesized using the behavioral test synthesis system BETS [12] from behavioral descriptions [27]. In this section, we report the results obtained for the following circuits implementing: 1) a fourth order IIR filter with a word size of 8 bits (4IIR.8); 2) a fourth order IIR filter with a word size of 14 bits (4IIR.14); 3) a speech filter of word size 12 bits (Speech.12); and 4) a wave digital filter of word size 8 bits (WDF.8). The designs generated by BETS consist of a structural VHDL specification of the data path and a functional VHDL specification of the controller, as shown in Fig. 2 and 3, respectively. The proposed DFT technique modifies the functional VHDL specification of the controller; the data path specification remains unchanged. The VHDL specifications are synthesized to gate-level circuits using OASIS [28]; one-hot encoding is used to synthesize the controller.

The results of applying scan-based DFT and the proposed controller-based DFT techniques on the different design examples are reported in the Tables II–V. The Tables show the different kind of circuits for each design (column *Circuit*), the DFT technique used to derive the circuits (column *DFT*), and the area in number of transistor pairs (column *Area*) after technology mapping using the SIS technology mapper [29] and the *lib2.genlib* standard cell library [30]. For the experiments, the scan chains have not been added to the circuits, and hence the area reported for the original and scan circuits are the same.

The rows *Controller*, *Datapath*, and *Contr + DP* refer to the controller circuit, the data path circuit, and the combined controller-data path circuit for each design. The rows under **Original** refer to the original design produced by the behavioral test synthesis system BETS [12], but without using the scan registers recommended by

TABLE IV
SPEECH.12: EFFECT OF SCAN-BASED AND CONTROLLER-BASED DFT SCHEMES

Circuit	DFT	Area	Faults		FC%	TE%	Vec	Tgen (secs)
			Total	Abt				
Original								
Controller	None	88	175	0	74.28	100.00	34	10.0
Datapath	None	5294	7443	1468	76.15	80.27	75	5216.2
Contr + DP	None	5379	7535	6810	3.03	*10.25*	57	15932.3
Scan DFT								
Controller	6 scan/6 FFs	88	175	0	100.00	100.00	38	0.4
Datapath	12 scan/132 FFs	5294	7443	342	91.28	95.48	200	1014.2
Contr + DP	18 scan/138 FFs	5379	7535	782	83.79	*89.62*	332	2622.0
Scan DFT + Controller DFT								
Contr + DP	18 scan/138 FFs + 2 TCV	5441	7610	103	93.36	*98.65*	402	2271.8

TABLE V
WDF.8: EFFECT OF SCAN-BASED AND CONTROLLER-BASED DFT SCHEMES

Circuit	DFT	Area	Faults		FC%	TE%	Vec	Tgen (secs)
			Total	Abt				
Original								
Controller	None	63	135	0	71.85	100.00	46	6.6
Datapath	None	3586	4689	15	94.62	99.68	578	218.2
Contr + DP	None	3638	4802	393	85.90	*91.81*	951	1571
Scan DFT								
Controller	7 scan/7 FFs	63	135	0	100.00	100.00	42	0.2
Datapath	None	3586	4689	15	94,62	99.68	578	218.2
Contr + DP	7 scan/119 FFs	3638	4802	308	87.69	*93.54*	780	1569.9
Scan DFT + Controller DFT								
Contr + DP	7 scan/119 FFs + 2 TCV	3673	4849	69	93.81	*98.57*	1577	705.4

BETS in the data path. The rows under **Scan DFT** show the same controller, data path, and controller-data path circuits with scan FF's used as follows. All the FF's of the controller are scanned. For the data path circuit, the scan FF's correspond to the registers selected by BETS as scan registers, and are the minimum number of FF's needed to break all loops, except self-loops, in the data path circuit by any gate-level loop-breaking partial scan tool, like [1]–[3]. The scan Control+DP circuit is obtained by just putting together the full scan controller and the partial scan data path, and hence does not contain any loops besides self-loops. The scan FF's and the total number of FF's used by the three circuits are indicated in the column DFT. For instance, for the design example 4IIR.8, the DFT column shows that all six FF's of the controller are scanned, while eight out of the 88 FF's used in the data path are scanned. The scan Contr+DP circuit has a total of 14 scan FF's out of a total of 94 FF's.

The application of the proposed controller-based technique on the scan Contr+DP circuit is shown under **Scan DFT + Controller DFT**. The *Contr + DP* row shows the same scan data path, with the controller redesigned by adding test control vectors, synthesized using OASIS [28] in exactly the same way as the original controller and scanning all the FF's of the controller. Note that the controller redesign technique does not increase the number of states and hence the number of FF's of the controller. The scan FF's used by the redesigned Contr+DP circuit is the same as the scan Contr+DP circuit. The only difference is the test control vectors added to the redesigned controller. For the IIR.8 example, the column DFT shows that the number of scan FF's remains the same at 14, and two test control vectors are added to the controller, as shown in Fig. 6.

The tables show the result of running the sequential test pattern generator HITEC [23] on the circuits in column *Circuit* for each design example. The total number of faults (*Total*) and the number of faults aborted (*Abt*) by HITEC are shown. Column *FC%* shows the percentage fault coverage, which is the percentage of faults for which a test could be found. Column *TE%* gives the percentage test efficiency, which is the percentage of faults either for which a test could be found or which could be identified as untestable (that is, the percentage of faults not aborted). The number of test vectors needed (*Vec*) and the test generation time taken (*Tgen*) in central processing unit secs on a Sparc10 are also reported.

The tables show that for most circuits, the original controller and data path circuits, separately and combined, are not testable. Consider the 4IIR.8 design example. The original data path has 69.3% fault coverage and 74.3% test efficiency. The original controller, while having a 100% test efficiency, has only a 76% fault coverage. The original controller-data path circuit is very untestable, having only 4.8% fault coverage and 11.8% test efficiency. Even for WDF.8 where the original controller and data paths are highly testable due to an absence of loops in the circuits, the test efficiency of the combined Contr+DP circuit goes down to 91.8%.

The use of scan DFT can make the controller and data path circuits highly testable when considered separately. However, for the combined controller-data path circuit, the test efficiency deteriorates significantly from that which can be achieved for the controller and data path individually. For example, for the IIR.8 example, the full scan controller has 100% fault coverage as well as 100% test efficiency. The data path using eight scan FF's (to break all loops) has close to 95% fault coverage and 100% test efficiency. However, the combined controller-data path circuit has only 77.2% fault coverage and 83.7% test efficiency.

The experimental results show the effectiveness of the controller-based DFT technique to significantly improve the testability of the scan controller-data path circuits. When the controller is re-

designed by adding the test control vectors by the proposed technique, followed by full scan of the control FF's, the redesigned controller-data path circuit becomes significantly more testable than the scan Contr+DP circuit. For the IIR.8 example, HITEC could achieve 93.6% fault coverage and close to 100% test efficiency for the redesigned controller-data path circuit (last row of Table II), up from 77.2% and 83.7%, respectively. Similarly, the controller DFT technique could enhance the fault coverage of other scan Cntr+DP circuits: from 82.3% to 94.4% for 4IIR.14, from 83.8% to 93.4% for Speech.12, and from 87.7% to 93.8% for WDF.8. There was a corresponding improvement in test efficiency: from 87.8% to 99.6% for 4IIR.14, from 89.6% to 98.7% for Speech.12, and from 93.5% to 98.6% for WDF.8.

The area overhead required by the proposed controller DFT technique is very low. The new technique does not have routing overhead associated with scan clock and scan chain or routing inputs/outputs to/from test points associated with other scan and nonscan DFT schemes. Experimental results show that the active area overhead due to adding the test control vectors, measured by the increase in number of cells, is very low. For the IIR.8 example, the controller DFT increases the number of cells from 3448 to 3502, which represents a marginal area overhead of 1.57%. The average area overhead for the circuits reported is only 1.03%.

VII. Conclusions

We have presented a new controller-based DFT technique to facilitate ATPG of sequential circuits consisting of controller and data path specifications. The proposed DFT technique uses both the functional specification of the controller and the structural specification of the data path to derive a few test control vectors which are added to the controller to offset control signal correlations that can cause conflicts during ATPG. We have demonstrated application of the technique, with nominal area overhead, to significantly improve the testability of controller-data path circuits. Note that though the proposed controller redesign technique was applied to designs with scan-based DFT, it can be equally effectively applied to designs with nonscan DFT schemes.

The proposed technique exploits the hierarchy information of the RT-level circuits, namely the controller and data path hierarchies, and their interface. In the future, we plan to extend the technique to be applicable to general hierarchical designs, consisting of multiple blocks for multiple processes, each block representing a controller-data path pair.

References

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

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

[3] V. Chickermane and J. H. Patel, "An optimization based approach to the partial scan design problem," in *Proc. Int. Test Conf.,* Sept. 1990, pp. 377–386.

[4] S. S. K. Chiu and C. Papachristou, "A built-in self-testing approach for minimizing hardware overhead," in *Proc. Int. Conf. Computer Design,* 1991.

[5] L. Avra, "Allocation and assignment in high-level synthesis for self-testable data paths," in *Proc. Int. Test Conf.,* 1991.

[6] I. Harris and A. Orailoğlu, "Microarchitectural synthesis of VLSI designs with high test concurrency," in *Proc. Design Automation Conf.,* June 1994, pp. 206–211.

[7] N. Mukherjee, M. Kassab, J. Rajski, and J. Tsyzer, "Arithmetic built-in self test for high-level synthesis," in *Proc. 13th IEEE VLSI Test Symp.,* Apr. 1995.

[8] C.-H. Chen, T. Karnik, and D. G. Saab, "Structural and behavioral synthesis for testability techniques," *IEEE Trans. Computer-Aided Design,* vol. 13, pp. 777–785, June 1994.

[9] A. Majumdar, R. Jain, and K. Saluja, "Incorporating testability considerations in high-level synthesis," *J. Electr. Testing: Theory Applicat.,* pp. 43–55, Feb. 1994.

[10] T. C. Lee, N. K. Jha, and W. H. Wolf, "Behavioral synthesis of highly testable data paths under nonscan and partial scan environments," in *Proc. Design Automation Conf.,* 1993, pp. 292–297.

[11] S. Dey and M. Potkonjak, "Transforming behavioral specifications to facilitate synthesis of testable designs," in *Proc. Int. Test Conf.,* Oct. 1994.

[12] M. Potkonjak, S. Dey, and R. Roy, "Behavioral synthesis of area-efficient testable designs using interaction between hardware sharing and partial scan," *IEEE Trans. Computer-Aided Design,* vol. 14, pp. 1141–1154, Sept. 1995.

[13] T. Thomas, P. Vishakantaiah, and J. Abraham, "Impact of behavioral modifications for testability," in *Proc. 12th IEEE VLSI Test Symp.,* Apr. 1994, pp. 427–432.

[14] P. Vishakantaiah, J. Abraham, and D. Saab, "CHEETA: Composition of hierarchical sequential tests using ATKET," in *Proc. ITC,* Oct. 1993.

[15] S. Bhatia and N. K. Jha, "Genesis: A behavioral synthesis system for hierarchical testability," in *Proc. Eur. Design Test Conf.,* Feb. 1994.

[16] J. Steensma, F. Catthoor, and H. D. Man, " Partial scan at the register–transfer level," in *Proc. Int. Test Conf.,* Oct. 1991, pp. 488–497.

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

[18] S. Bhattacharya, F. Brglez, and S. Dey, "Transformations and resynthesis for testability of RT-level control-data path specifications," *IEEE Trans. VLSI Syst.,* vol. 1, pp. 304–318, Sept. 1993.

[19] V. Chickermane, J. Lee, and J. Patel, "Addressing design for testability at the architectural level," *IEEE Trans. Computer-Aided Design,* vol. 13, pp. 920–934, July 1994.

[20] K. Wagner and S. Dey, "High level synthesis for testability: A survey and perspective," in *Proc. Design Automation Conf.,* June 1996, pp. 131–136.

[21] M. Abadir and M. Breuer, "Test schedules for VLSI circuits having built-in test hardware," *IEEE Trans. Comput.,* vol. C-35, no. 4, pp. 361–367, 1986.

[22] D. Mukherjee, M. Pedram, and M. Breuer, "Merging multiple FSM controllers for DFT/BIST hardware," in *Proc. IEEE Int. Conf. Computer-Aided Design,* Nov. 1993, pp. 720–725.

[23] T. M. Niermann and J. H. Patel, "HITEC: A test generation package for sequential circuits," in *Proc. EDAC,* 1991, pp. 214–218.

[24] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* San Francisco, CA: Freeman, 1979.

[25] C. Tseng and D. Siewiorek, "Automated synthesis of data paths on digital systems," *IEEE Trans. Computer-Aided Design,* vol. 5, pp. 379–395, July 1986.

[26] J. Turner, "Almost all k-colorable graphs are easy to color," *J. Algorithms,* vol. 9, no. 1, pp. 63–82, 1988.

[27] R. Haddad and T. Parsons, *Digital Signal Processing: Theory, Applications and Hardware.* New York: Computer Science, 1991.

[28] *OASIS Users Guide,* K. K., Ed., MCNC, Research Triangle Park, NC, 1991.

[29] E. Sentovich, K. Singh, C. Moon, H. Savoj, R. Brayton, and A. Sangiovanni-Vincentelli, "Sequential circuit design using synthesis and optimization," in *Proc. Int. Conf. Computer Design,* Oct. 1992.

[30] S. Yang, "Logic synthesis and optimization benchmarks, user guide version 3.0," in *Int. Workshop Logic Synthesis,* MCNC, Research Triangle Park, NC, May 1991.