Nonscan Design-for-Testability Techniques Using RT-Level Design Information

Sujit Dey, *Member, IEEE*, and Miodrag Potkonjak, *Member, IEEE*

***Abstract*— This paper presents *nonscan* design-for-testability (DFT) techniques applicable to register-transfer (RT)-level data path circuits. Knowledge of high-level design information, in the form of the RT-level structure, as well as the functions of the RT-level components is utilized to develop effective nonscan DFT techniques. Instead of conventional techniques of selecting flip-flops (FF's) to make controllable/observable, execution units (EXU's) are selected using the EXU S-graph introduced in this paper. Controllability/observability points can be implemented using register files and constants.**

We introduce the notion of k-level controllable and observable loops and demonstrate that it suffices to make all the loops k-level controllable/observable, $k > 0$, to achieve very high test efficiency. The new testability measure eliminates the need by traditional DFT techniques to make all loops directly (zero-level) controllable/observable, reducing significantly the hardware overhead required and making the nonscan DFT approach feasible and effective. We discuss ways of avoiding the formation of reconvergent regions while adding test points to make loops k-level controllable/observable. We introduce dual points, which utilize the different controllability/observability levels of loops, to make one loop controllable while making another loop observable. We present efficient algorithms to add the minimal hardware possible to make all loops in the data path k-level controllable/observable, without the use of scan FF's.

The nonscan DFT techniques were applied to several data path circuits. The experimental results demonstrate the effectiveness of the k-level testability measure, and the use of distributed and dual points, to generate easily testable data paths with reduced hardware overhead. The hardware overhead and the test application time required for the nonscan designs are significantly lower than for the partial scan designs. Most significantly, the experimental results demonstrate the ability of the RT-level DFT techniques to produce nonscan testable data paths, which can be tested at-speed.

***Index Terms*— At-speed testing, data paths, k-level testable, nonscan, testability measure, test points.**

I. Introduction

Among the several design-for-testability (DFT) techniques that have been proposed to ease the task of sequential test-pattern generation, the partial-scan technique has become increasingly popular [1]–[3]. Unlike full scan, where all the flip-flops (FF's) in the circuit are made observable and controllable, the partial-scan technique selects a subset of FF's for scan. While it may be possible to achieve test efficiency like that achieved by a full-scan circuit, a partial-scan circuit usually requires less area and delay overheads and smaller test application times due to the presence of fewer FF's in the scan chain.

The scan-based techniques have the disadvantage that the test application time is very large compared to nonscan designs, however, since the test vectors have to be shifted through the scan chain. Reduction of test application time has been addressed in several ways, like arranging scan flip-flops in parallel scan chains [4] and reconfiguring scan chains [5]. In the parallel-scan chain approach, the number of parallel-scan chains, and hence the number of vectors that can be shifted in parallel, is limited by the minimum of the number of primary inputs and primary outputs of the circuit. The reconfigurable scan chain approach [5] is limited by the ability of the circuit to be decomposed into a set of kernels, which are disjoint portions of logic that can be tested independently. On the other hand, a nonscan DFT technique, such as the one presented in this paper, would not require the scan of any FF's, thus eliminating the need to shift test vectors through scan chains and greatly reducing the test application time.

Another key disadvantage of scan-based DFT techniques is that the test vectors cannot be applied at the operational speed of the circuit, that is, test vectors cannot be applied at consecutive clock cycles. The inability of scan designs to be tested at-speed assumes significance in light of recent studies, which show that a stuck-at test set applied *at-speed* identifies more defective chips than a test set having the same fault coverage but applied at a lower speed [6]. The main advantage of nonscan designs is that the test vectors can be applied at-speed.

A. Nonscan DFT Using Test Points

A DFT technique that does not involve scan is the use of test points. The application of test points has focused mainly on combinational circuits, most often with the goal of reducing the number of required test vectors. In the early 1970's, Heyes and Friedman [7] and Saluja and Reddy [8] proposed the insertion of test points in a combinational circuit as a means to make the circuit fully testable by a test set of small cardinality. Krishnamurthy [9] analyzed the computational complexity of the associated optimization problem and proposed an optimal dynamic programming-based approach for a special case of the problem. Pomeranz and Kohavi [10] proposed an optimization method for test-point insertion for general combinational circuits. More recently, Pomeranz and Reddy [11] proposed

Manuscript received October 4, 1994; revised July 27, 1995. This paper was recommended by Associate Editor K.-T. Cheng.

S. Dey was with C&C Research Laboratories, NEC USA, Princeton, NJ 08540 USA. He is now with the Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093-0407 USA.

M. Potkonjak is with the Computer Science Department, University of California, Los Angeles, CA 90095 USA.

Publisher Item Identifier S 0278-0070(97)09364-0.

0278–0070/97$10.00 © 1997 IEEE

a technique to insert test points to improve significantly the path delay fault coverage of large combinational circuits. A technique to provide test points is "bed of nails" [12], where the tester probes the underside of a board so that numerous, spatially isodistant points of controllability and observability are provided. Controllability and observability points are also provided in silicon-based solutions such as CrossCheck [13], [14], but test application is slow due to the need to scan observation and control points and due to transmission delays on long diffusion lines used to select the observation and control points.

The recent findings of the advantages of at-speed testing on the defect coverage of test sets [6] motivated researchers to investigate nonscan DFT techniques to make sequential circuits testable by introducing controllability and observability points [15], [16]. The feasibility of nonscan DFT techniques to produce testable sequential circuits with high test efficiency was demonstrated in [15] and [16].

B. High-Level DFT Techniques

Recently, several behavioral-level design and synthesis approaches have been proposed to generate easily testable data paths for both built-in-self-test (BIST)-based testing methodology [17]–[20] and automatic test-pattern generation (ATPG) methods [21]–[28]. These techniques either modify the behavioral description of a design to improve the testability of the resulting circuit [21], [27], [28] or consider testability as one of the design objectives during the behavioral synthesis process.

In [21], [29], and [30], high-level testability analysis techniques were developed to identify hard-to-test parts of a circuit from a behavioral description. The testability information is then used to select test points or flip-flops for partial scan or insert test statements in the behavioral description [21], [28] to enhance the testability of the resulting circuit. Another set has been developed of techniques that modify the behavioral synthesis process itself to synthesize easily testable circuits. Most of the behavioral test synthesis techniques that target sequential ATPG aim to synthesize data paths without loops, by using proper scheduling and assignment, and/or scan registers to break loops [22], [24]–[26].

Almost all of the BIST-based approaches assume a scan design methodology since random testing is not well suited for sequential circuits [17]–[19], [31]. Also, almost all of the ATPG-based behavioral test synthesis approaches, with the exception of methods proposed in [23] and [25], assume the use of scan registers to make the data paths testable. The nonscan techniques presented in [23] and [25] produce testable data paths only when the designs have a large number of primary inputs (PI's) and primary outputs (PO's), however, and do not have any loops [25]. For instance, for the popular design example of fifth-order elliptical wave filter, the nonscan scheme could not make the data path testable [25]. In general, most designs have a small number of PI's and PO's and have several types of loops formed, partly due to the presence of such loops in the specification itself and partly due to resource sharing employed to generate area-efficient data paths [26]. Consequently, the existing high-level testability techniques either are based on using scan or are not suitable for practical data paths having loops.

Several techniques have been developed to improve the testability of circuits by exploiting the register-transfer (RT)-level description of designs. Transformation and optimization techniques were proposed in [32], which utilize RT-level information to generate optimized designs that are 100% testable under full scan. Chickermane *et al.* [33] showed that the use of RT-level information to select scan flip-flops results in significantly better performance when compared to techniques limited to gate-level information only. Steensma *et al.* [34] proposed an efficient partial-scan methodology applicable to data paths described at the RT level. The method is based on eliminating loops by making existing registers scannable or by adding extra transparent scan registers. In [35], an RT-level method was presented to generate self-testable RT-level data paths, using allocation and automatic test-point selection to reduce the sequential depth from controllable to observable registers. A comprehensive survey of the behavioral and RT-level test synthesis techniques can be found in [36]. Like the behavioral synthesis for testability techniques summarized above, all the existing RT-level techniques are scan based and cannot generate testable data paths without the use of scan.

C. Testability of Sequential Circuits

The dependencies of the FF's of a sequential circuit are captured by an S-graph [1], [2]. It has been empirically determined that sequential test generation complexity may grow exponentially with the length of cycles in the S-graph [1], [2]. An effective partial-scan approach selects scan FF's in the minimum feedback vertex set (MFVS) of the S-graph so that all loops, except self-loops, are broken and the sequential depth is minimal [1], [2]. Existing nonscan techniques also restrict themselves to FF's as the nodes to be made controllable. The nonscan technique presented in [15] and [16] selects FF's to load from primary inputs (add control point) such that the loops in the circuit are broken.

Breaking all the cycles in a circuit by scan FF's may be very expensive in terms of the scan overhead, especially for data paths that have a tendency to have complex loop structures [26]. Also, the presence of a large number of FF's in the scan chain increases the test application time. For nonscan designs, effective controllability of FF's is limited by the number of primary inputs available.

D. The New Approach

This paper presents new DFT techniques to make data paths testable, without using scan registers. The need for a large number of test points with a nonscan DFT approach may adversely affect the testability of the resulting circuit by introducing correlation between signals and making the test application time undesirably long. In the proposed approach, high-level design information, in the form of the RT-level description of the data path, is utilized to employ various test-point minimization techniques that contribute to make the nonscan DFT method feasible in producing testable designs.

Since we target data-flow-intensive application domains, like digital signal processing, communications, and graphics, only a few FF's are needed for the states of the controller. In our DFT framework, we make the state FF's controllable by multiplexing them with primary input signals. If the control signals from the controller are not fully controllable after making the state FF's controllable, we apply our control DFT technique outlined in [37] to make them controllable. For data-flow-intensive designs, there usually are no status signals coming back from the data path to the controller. If there are such status signals, however, we make them observable by multiplexing them with the primary outputs. Typically, the number of status signals is just a few, if any, and there are always enough primary outputs to observe the status signals. We assume that the underlying hardware model used is the dedicated register file model. This model assumes that all registers are grouped in a certain number of register files (each register file contains one or more registers) and that each register file can send data to exactly one EXU. At the same time, each EXU can send data to an arbitrary number of register files. This model is used in several high-level synthesis systems [38], [39], as well as many manual application-specific integrated circuits and general-purpose data paths [40]. Note that although the discussions in this paper assume the dedicated register file model, the nonscan DFT techniques presented here are applicable to any arbitrary hardware model.

We use the RT-level structure of data paths to introduce the EXU S-graph, which captures the dependencies between the EXU's of the data path. We show that the choice of EXU's (their outputs) as the nodes to be made controllable/observable is more effective than the choice of FF's (registers at the RT level) used by traditional scan and nonscan DFT techniques [1], [2], [15], since the MFVS of the EXU S-graph is a lower bound to the MFVS of the S-graph of its registers. Also, as opposed to making the same node controllable and observable (as in scan approaches), we use a more cost-effective distributed approach, where some nodes are made controllable while others are made observable.

This paper shows that breaking all the loops is not necessary to make a circuit testable. We develop a new testability measure, based on k-level controllability and observability of loops. It is demonstrated that it suffices to make loops k-level controllable/observable, instead of directly breaking them, to make the data paths highly testable. We introduce nonscan DFT techniques based on RT level, such as adding constants and dual points, to make all loops in the data path k-level controllable/observable. We show that adding controllability/observability points can introduce reconvergent structures in the circuit. The effects of different types of reconvergent structures on the testability of the circuit are analyzed, and a technique is proposed to trade off between the formation of reconvergences and the level of controllability/observability achieved. We present an efficient algorithm to add the minimal hardware possible to make all loops in the data path k-level controllable/observable, using the RT-level information and both the distributed and dual points approach introduced.

We applied our technique to several moderately sized data paths. The experimental results demonstrate the effectiveness of the k-level heuristic and our nonscan DFT technique to design highly testable data paths with nominal hardware overhead. Besides the main advantage of at-speed testing, experimental results also demonstrate that the hardware overhead and the test application time required for the nonscan designs are significantly lower than for the partial-scan designs. In the next section, we illustrate the proposed nonscan DFT process using a design example.

II. Scan and Nonscan DFT of RT-Level Data Paths: An Illustration

Fig. 1(a) shows the RT-level data path for the fourth-order IIR cascade filter, synthesized from behavioral description [37] using the HYPER high-level synthesis system [39]. The basic RT-level components of a typical data path are EXU's (like adders, multipliers, arithmetic and logical units, and transfer units), registers, multiplexors, and interconnects. The data path in Fig. 1(a) has two adders, three multipliers, 12 multiplexors, and 12 registers, as shown by row "4IIR" in Table I. There are several transfer units—TU1, TU2, TU3, and TU4–which are used to transfer data produced in a clock cycle of one iteration to another module in some clock cycle of a subsequent iteration. The transfer units are composed of registers and multiplexors, the multiplexors needed to hold current data until new data can be stored. Each interconnect shown in the data path is n-bit wide, where n is the word size of the design.

Similar to the S-graph of a gate-level sequential circuit [1], [2], the S-graph of a data path identifies the dependencies between the registers of the data path. The S-graph in Fig. 1(b), corresponding to the data path in Fig. 1(a), reveals the existence of several loops involving the registers of the data path. As can be expected, sequential ATPG is very difficult for the data path, as indicated in Table III by the row "Orig."

The testability of the data path can be improved using partial-scan techniques to break all the loops of the circuit. Since the MFVS of the S-graph in Fig. 1(b) is three, breaking all the loops requires scanning at least three registers, namely, LA1, LA2, and LM1. For the 20-bit IIR filter data path shown in Fig. 1(a), 60 scan FF's are needed by the gate-level partial-scan tool OPUS [3], and the Lee–Reddy partial-scan tool [2], shown by the rows "Opus" and "LR," respectively, in Table III. The sequential ATPG program HITEC [41] can achieve 100% test efficiency on the scan designs, requiring 156 test vectors for the Opus design. Besides the high area overhead, the scan designs have high test application time, indicated by the column "Tappl" in Table III. For instance, the Opus design needs $156 * (60 + 1) = 9516$ clock cycles to apply all of the 156 test vectors. Most important, the scan designs cannot be tested at-speed.

In the following sections, we first introduce an EXU S-graph, which is used to select points to be made controllable/observable using nonscan techniques. Next, possible implementations of the nonscan techniques are discussed. Last, it is shown how the IIR cascade data path can be modified to make it testable without the use of scan and using significantly less hardware overhead than the scan solutions mentioned above.

(a)

(b)

(c)

Fig. 1. Characteristics of a design implementing the fourth-order IIR cascade filter. (a) RT-level data path. (b) Register S-graph. (c) EXU S-graph.

A. EXU S-Graph

It is not necessary for a nonscan DFT technique to restrict to only registers the choice of nodes to break, that is, make controllable/observable. Also, as opposed to making the same point controllable as well as observable, it may be more cost effective to make some points controllable and others observable. In this section, we introduce the EXU S-graph to show that in a data path, (the outputs of) EXU's are better choices for controllable/observable points than registers. Each node in the EXU S-graph represents an EXU in the data path. There exists a directed edge from node u to node v, labeled i and denoted $u \xrightarrow{i} v$, if there exists a direct path from EXU u to the ith register file of EXU v without going through any other register. In the rest of the paper, by "nodes" and "edges," we generally refer to the EXU S-graph.

TABLE I
CHARACTERISTICS OF THE RT-LEVEL DATA PATHS

Design	Bits	Add	Mult	Reg	Mux	Inter	Area (tr pairs)
4IIRcas	20	2	3	12	12	20	7486
Speech	20	2	3	12	9	9	6022
EWF	16	3	3	23	29	20	6968
EWFhigh	20	1	1	18	23	6	7538
4IIRpar	20	6	6	23	0	23	9757

The EXU S-graph for the data path in Fig. 1(a) is shown in Fig. 1(c). In the data path of Fig. 1(a), there is a direct path from the multiplier M3 to the left (first) register file of adder A2; hence, the edge M3 $\xrightarrow{1}$ A2 in the EXU S-graph shown in Fig. 1(c). Similarly, M1 sends data to both the first and second register files of A2, reflected by the M1 $\xrightarrow{1}$ A2 and M1 $\xrightarrow{2}$ A2 in the EXU S-graph. (Without loss of generality, we assume that the left and right register files of an EXU are numbered 1 and 2, respectively.) Note that for ease of illustration, instead of showing k separate edges with k different labels from a node u to node v, we show a single edge having all the k labels.

The EXU S-graph shown in Fig. 1(c) has several loops, as did the register S-graph shown in Fig. 1(b). There are two loops in the EXU S-graph between M1, A2, and TU1, namely, M1 $\xrightarrow{1}$ A2 $\xrightarrow{1}$ TU1 $\xrightarrow{1}$ M1 and M1 $\xrightarrow{2}$ A2 $\xrightarrow{1}$ TU1 $\xrightarrow{1}$ M1. However, all the loops in the EXU S-graph pass through the two EXU's, A1 and A2. Hence, the MFVS of the EXU S-graph is two, as opposed to an MFVS of three for the register S-graph of Fig. 1(b). In general, for a circuit using the dedicated register-file model, the MFVS of an EXU S-graph is a lower bound of the MFVS of the register S-graph, as proved in the appendix. Hence, EXU's are better candidates for being made controllable/observable than the registers of the data path.

B. Techniques to Add Controllability/Observability Points

A node can be made controllable/observable in several ways. We discuss two possible schemes, which do not use scan registers and are applicable to data paths at the RT level. Fig. 2(a) shows an EXU, A2, and its register files used in the elliptical wave digital filter (EWF) data path reported in Table V. The left register file of A2, RF1, consists of two registers, L1 and L2; L1 receives data from EXU's A1, A2, A3, and M2, while L2 receives data from A2.

1) Direct Scheme: The output of A2 can be made controllable by adding an interconnect from a PI to a multiplexor placed at the output of A2, as shown in Fig. 2(b). The multiplexor is controlled by the test pin *ntest,* which is set to "0" during the normal operation of the data path and can be set to any value required during the test mode, as determined by sequential ATPG. Hence, the output of the multiplexor Z remains unchanged to be the output of A2 during the normal operation of the data path, while it can be set to the value of the PI during the test mode, making the point Z directly controllable.

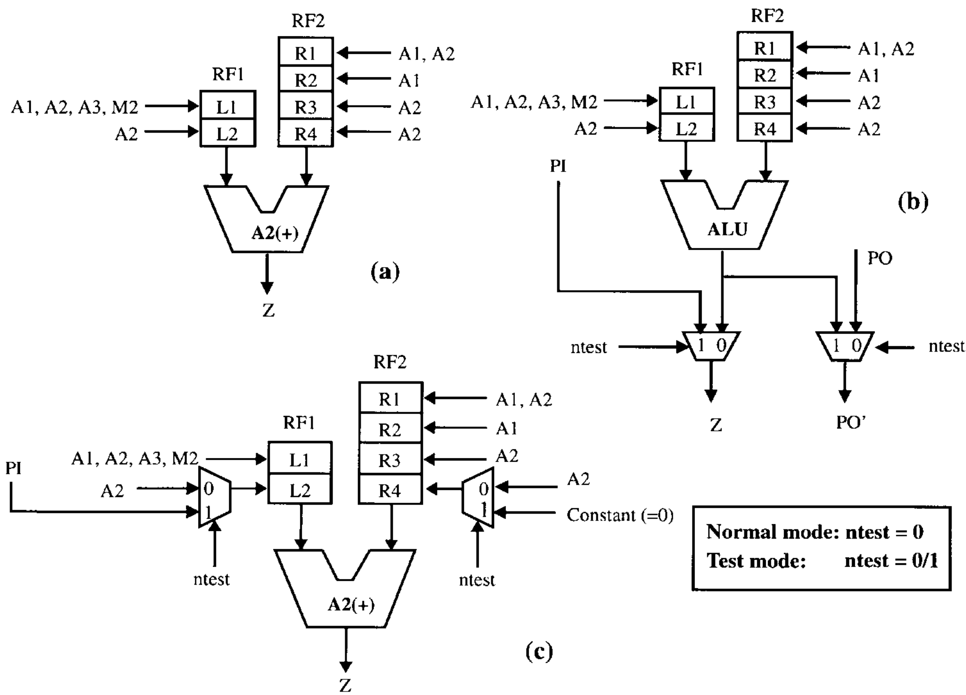


Fig. 2. Nonscan schemes to incorporate controllability/observability. (a) An EXU and its register files from EWF data path (Table V). (b) Direct scheme. (c) Register-file-based scheme.

Similarly, the output of A2 can be made observable by adding an observability point (probe point) from A2 and multiplexed with the PO, as shown in Fig. 2(b). The multiplexor is controlled by the test point *ntest,* which is operated in a way similar to the control point multiplexor, ensuring that the functionality of the data path remains unchanged. Note that at any clock cycle, either the probe point or the PO, but not both, is observable at PO′, the new output. Also, the number of probe points that can be added is limited by the number of PO's of the circuit unless multiple probe points are multiplexed to the same PO and multiple test pins are used. The latter scheme would potentially increase test time, since only one of the multiple probe points can be observed in any clock cycle.

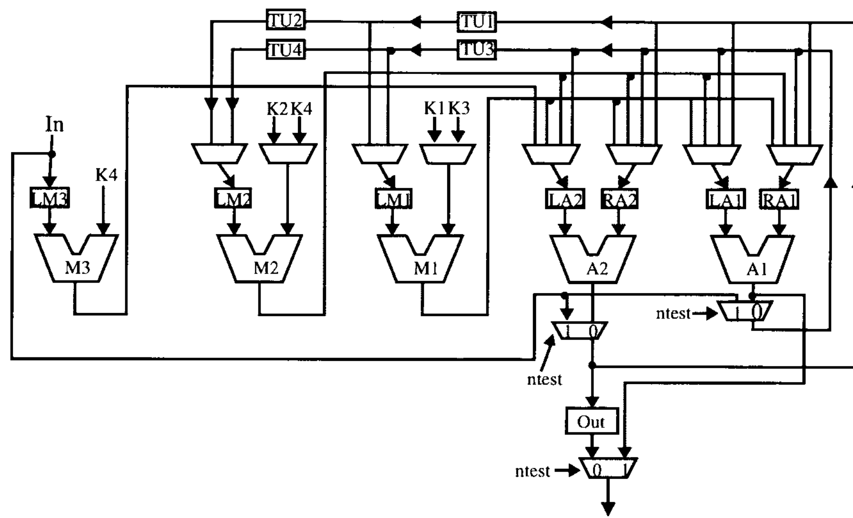
2) Register-File-Based Scheme: Instead of adding the controllability point to the output of the EXU, controllability points can be added to the register files associated with the EXU also. In that case, it suffices to make only one register of each register file of the EXU controllable. Fig. 2(c) shows a possible way: register L2 of the left register file is made controllable by adding the control point from PI. Register R4 is made controllable by adding a constant (say, the identity element of the operation performed by the EXU, in this case "0"). It can be verified that any value at the output of A2 can be justified by setting a proper value at the PI. The register-file-based scheme may be attractive if one of the register files already has a controllable register or in the context of k-level controllability, which will be introduced later in this paper.

C. Nonscan DFT of the IIR Cascade Filter

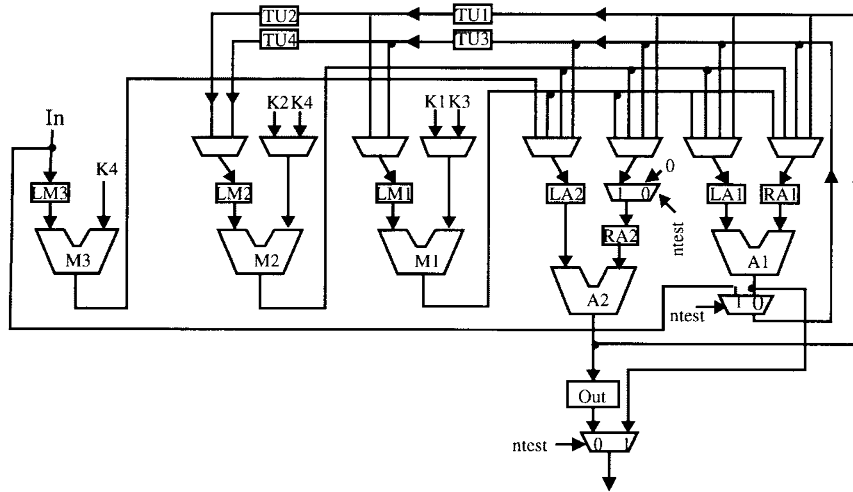
Having discussed two basic ways by which nodes (outputs of EXU's) can be made controllable/observable, we proceed with the task of nonscan DFT of the data path of the fourth-order IIR cascade filter, shown in Fig. 1(a). As mentioned in Section II-A, the MFVS of the EXU S-graph of the data path, shown in Fig. 1(c), is A1 and A2. Hence, making the outputs of A1 and A2 controllable/observable would break all the loops directly, that is, make all the loops *zero-level* controllable/observable. That is, any value at the outputs of A1, A2 can be controlled and observed in one clock cycle (time frame). Compared to the register S-graph solution, which requires making three registers controllable/observable, this solution seems better. We show, however, that much less expensive nonscan DFT techniques would suffice to make the data path testable.

The EXU S-graph in Fig. 1(c) reveals that all loops through A2 are observable, since A2 goes directly to the PO "Out." Hence, we need to add only a controllability point to the output of A2, while adding both a controllability and observability point to the output of A1. Fig. 3(a) shows the modified data path of Fig. 1(a), with test hardware added (shown in bold) to insert one controllability point at the output of A1 and A2 and one observability point from the output of A2. A test efficiency of 100% could be achieved on the resultant data path, as evidenced by the row "0-lev" in Table III. The test hardware overhead required for the modified data path is 429 cells (5.7% of the original data path), which is less than the overhead of 665 cells needed for the scan designs (rows "Opus" and "LR" in Table III). Besides having the main advantage of at-speed testing, the number of clock cycles required for test application (column "Tappl") for the nonscan design is much less than the scan design. However, the main advantage of the nonscan design shown in Fig. 3(a) over the scan designs is the ability of at-speed testing.

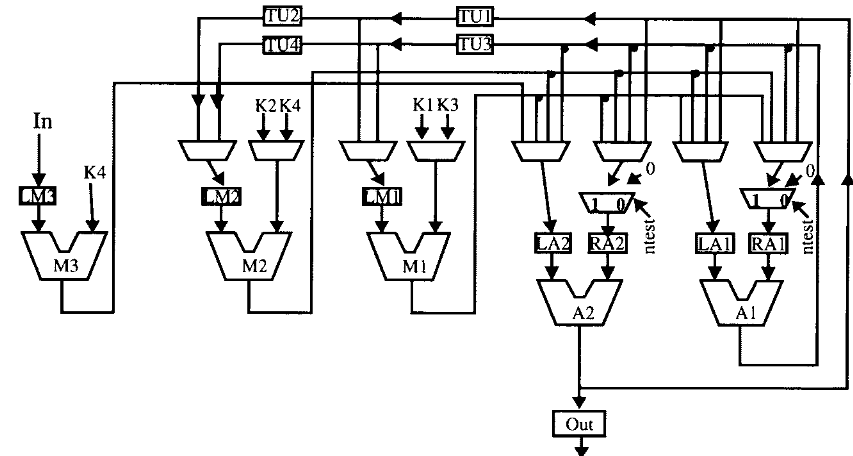
It is not necessary to make the loops of the data path directly (zero-level) controllable/observable. Fig. 3(b) shows an alternate testable design, with the nonscan test hardware



(a)



(b)



(c)

Fig. 3. Nonscan DFT of the data path in Fig. 1(a). (a) Zero-level testable data path: all loops are zero-level controllable/observable. (b) One-level testable data path. (c) Two-level testable data path.

shown in bold. Instead of adding a controllability point to the output of A2, only a constant ("0," the identity element of addition) is added to the right register file (RA2) of A2. Any value at the output of A2 can still be justified by at most two time frames. For example, if a value of nine needs to be justified at the output of A2, in one time frame the registers LA2 and RA2 can be set to appropriate values nine and zero, and in the next time frame, the values of LA2 and

RA2 can be justified by "In" and the constant K4. Adding the constant requires much less hardware overhead than adding a controllability point at the output of A2, since the multiplexor logic associated with the constant signals can be pruned. Note that the new nonscan design makes use of the *register-file-based* scheme, described in Section II-B1. The loops through A2 are now one-level controllable. The resultant (one-level controllable/observable) data path shown in Fig. 3(b) has less hardware overhead than the zero-level solution shown in Fig. 3(a). Also, a very high test efficiency of 98% could be achieved on the resultant data path, as evidenced by the row "1-lev" in Table III.

The data path in Fig. 3(c) demonstrates more effectively the benefits of nonscan DFT at the RT level and the new notion of k-level controllable/observable loops. The data path shows the addition of just two constants to the right registers, RA1 and RA2, of the EXU's A1 and A2, respectively. As will be explained in Section III, all the loops in the EXU S-graph now become two-level or less controllable/observable. The test hardware required is significantly less than the zero-level and one-level testable data paths shown in Fig. 3(a) and (b), respectively, as shown in row "2-lev" in Table III. The area overhead is only 120 cells, as compared to an overhead of 665 cells for the scan design, 429 cells for the zero-level nonscan design, and 349 cells for the one-level nonscan design. The two-level testable design, however, has a very high test efficiency of 98%, comparable with the test efficiency achieved by the more expensive scan designs and the zero-level and one-level nonscan designs.

The nonscan designs and their high test-efficiency results demonstrate the feasibility of using the nonscan DFT schemes introduced in Section II-B. Moreover, the highly testable nonscan designs shown in Fig. 3(b) and (c) establish the technique of making loops k-level controllable/observable as a viable, efficient, and cost-effective alternative to the traditional DFT technique of breaking all loops directly, that is, making the loops zero-level controllable/observable. The new and effective testability metric of k-level controllable/observable loops is explained in detail in the next section.

III. k-LEVEL CONTROLLABLE/OBSERVABLE LOOPS: A COST-EFFECTIVE DFT APPROACH

In this section, we first define k-level controllable/observable nodes. As mentioned in the previous section, we exclusively consider (the outputs of) EXU's as the possible nodes. Next, we introduce nonscan DFT techniques to make nodes k-level controllable/observable. Given a data path, we show how the controllability/ observability levels can be calculated. Next, we define k-level controllable/observable loops and k-level testable data paths in terms of k-level controllable/observable nodes.

Definition 1: An EXU M is k-level controllable/observable if any value on the output of M can be justified/propagated in at most $k+1$ clock cycles (time frames). Alternatively, for any value that needs to be justified/propagated at the output of M, there exists at least one vector sequence of length at most $k+1$ that justifies/propagates the value.

Consider the data path shown in Fig. 3(c). The output of A1 is two-level controllable, as explained below. For example, to justify a value of 15 at the output of A1, in the first time frame LA1 can be set to 15 and RA1 to zero. In the second time frame, the value of RA1 can be immediately justified by the constant. To justify the value of LA1, which is the output of A2, the input registers of A2, LA2 and RA2, are set to 15 and zero, respectively. In the third time frame, RA2 can be justified because of the presence of the constant to RA2. Suppose the constant K4 to M3 is one. LA2 can be justified by setting "In" to 15. Similarly, any value at the output of A1 can be justified in three time frames, making A1 two-level controllable. Note that without the addition of the constants, the output of A1 is not controllable, as is in the original data path of Fig. 1(a).

The output of A1 is two-level observable, since any value at the output of A1 can be propagated out in three clock cycles in the following way. In the first clock cycle, A1 can be propagated to LA2. Since RA2 can be independently controlled to the constant (here zero), in the next clock cycle, LA2 can be propagated to the output of A2, and hence register "Out." In the third clock cycle, register "Out" can be directly observed at the PO. Consequently, the output of A1 is made two-level observable.

A. DFT for k-Level Controllability/Observability

As introduced in Section II-B, the output of an EXU, Z, can be made k-level controllable/observable by either the direct scheme or the register-file-based scheme. In the direct scheme, the EXU output is directly multiplexed with a k-level controllable node to make Z k-level controllable. The EXU output is made k-level observable by directly multiplexing it with another node that is k-level observable. Consider the EXU shown in Fig. 4. Fig. 4(b) shows how ALU1 is made k-level controllable and observable using the direct scheme.

In the register-file-based scheme, an EXU (output) is k-level controllable if at least one register of each register file of the EXU has a $(k-1)$-level controllable input, as shown in Fig. 4(c). An EXU is k-level observable if it has an interconnect to a register file of another EXU, which is $(k-1)$-level observable, and whose other register file has a one-level controllable input. Fig. 4(d) shows how ALU1 is made k-level observable. Note that the signals X and Y in Fig. 4(c) should be different signals; otherwise, a reconvergent structure from fanout point X to node Z will limit the k-level controllability achieved. The adverse effects of different types of sequential reconvergent regions, their possible introduction while adding controllability/observability points, and a way to avoid them will be addressed in Section IV.

B. Calculating the Controllability/Observability Levels

To be able to add appropriate testability hardware to make nodes k-level controllable/observable, for a user-specified k, we need to be able to calculate the controllability/observability levels of nodes. Let the controllability level of a node X be denoted as clevel(X). That is, clevel$(X) = k$ indicates that node X is k-level controllable. Let RF(M) denote the set of

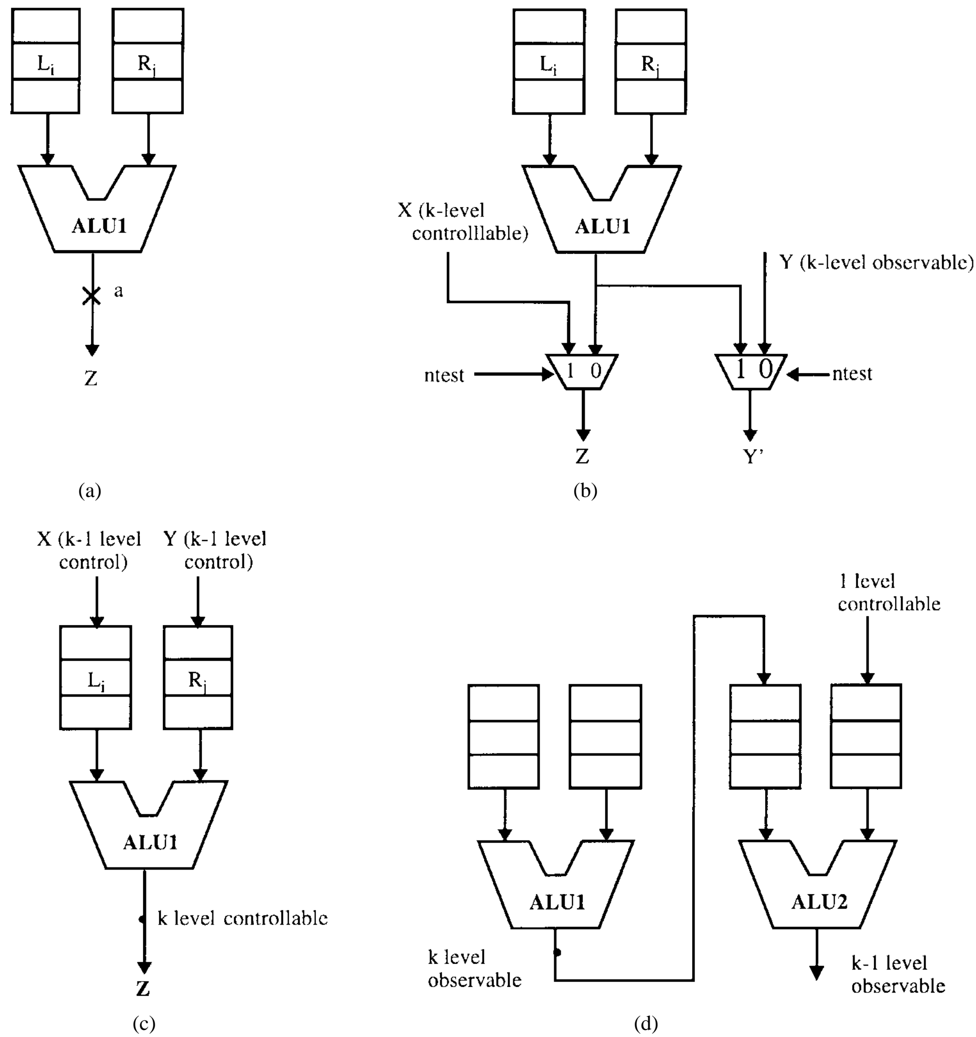


Fig. 4. k-level controllability/observability. (a) An EXU. (b) Direct scheme for making ALU1 k-level controllable/observable. (c) Register-file-based scheme for making ALU1 k-level controllable. (d) Register-file-based scheme for making ALU1 k-level observable.

register files of EXU M and out(M) denote the output bus of EXU M. Then, the controllability level of an EXU M is shown in (1) at the bottom of the page.

Similarly, the observability level can be calculated using the direct and register-file-based schemes of making a node k-level observable, discussed in Section III-A. Let EXU_i be the set of execution units in the fanout of M, such that M sends data to one of the register files of EXU_i and the other register file is controllable to level 1. See (2) at the bottom of the page.

Consider the EXU shown in Fig. 2(a). Assume clevel(A1) $= 1$, clevel(A3) $= 3$, and clevel(M2) $= 2$. In that case, clevel(A2) $= \max\{1, 1\} + 1 = 2$. Fig. 2(c) shows the nonscan DFT, using the register-file-based scheme, to make clevel(A2) $= 1$. In the data path shown in Fig. 3(c), olevel(A1) $= 1 + 1 = 2$.

C. k-Level Controllable/Observable Loops, k-Level Testable Data Path

Definition 2: A loop is k-level controllable if there is at least one node in the loop that is k-level controllable. A loop is k-level observable if there is at least one node in the loop that is k-level observable.

Definition 3: A data path is k-level testable if all loops in the data path are k-level or less controllable and observable.

$$\text{clevel(M)} = \begin{cases} k & \text{out(M) multiplexed with } X\text{, clevel}(X) = k \\ \max_{\mathrm{R}\in\mathrm{RF(M)}}\{\min_{i \text{ an input to } \mathrm{R}}[\text{clevel}(i)] + 1\} & \text{otherwise} \end{cases} \tag{1}$$

$$\text{olevel(M)} = \begin{cases} k & \text{out(M) multiplexed with } Y\text{, olevel}(Y) = k \\ \min_{\mathrm{EXU}_i}\{\text{olevel}(\mathrm{EXU}_i) + 1\} & \text{otherwise} \end{cases} \tag{2}$$

Consider the data path shown in Fig. 3(c). It has been derived from the data path in Fig. 1(a) by adding two constants ("0") to the right registers, RA2 and RA1, of the EXU's A2 and A1, respectively. All loops going through A1 are two-level controllable and two-level observable since A1 is two-level controllable/observable, as shown before. Similarly, all loops going through A2 are one-level controllable/observable. Hence, the data path shown in Fig. 3(c) is two-level testable.

Note that to make the data path in Fig. 1(a) zero-level testable, two controllability points and one observability point need to be inserted [Fig. 3(a)]. On the other hand, to make the data path two-level testable, only two constants need to be added [Fig. 3(c)]. As shown in Table III, the test area overhead of the resultant two-level testable data path (120 cells) is significantly smaller than the overhead of the zero-level testable data path (429 cells). Because of its very low test hardware overhead (1.6%) and high test efficiency (98%), the two-level testable design is a cost-effective alternative to the more expensive zero-level testable design and the much more expensive scan designs, as reported in Table III. In the next section, we study the effect of reconvergences that may be introduced while inserting test points and propose techniques to avoid them.

IV. Reconvergent Regions Introduced While Adding Nonscan DFT

Adding controllability and observability points to make loops k-level controllable/observable, as discussed in Section III, may introduce signal reconvergences. The presence of reconvergent regions across time frames in sequential circuits may cause problems for sequential ATPG that are similar to the reconvergent fanout problems that combinational ATPG experiences. We first briefly discuss the adverse effect of reconvergent regions on test-pattern generation and some attempts to alleviate the problem. Next, we analyze the role of reconvergent regions in sequential circuits in making sequential ATPG difficult. We then explain how the nonscan DFT technique proposed in this paper can introduce signal reconvergences. A technique is described to avoid creation of reconvergences, which entails a tradeoff with the level of controllability/observability achieved.

If there is more than one disjoint path from a fanout node s to another node r, then fanout s is said to reconverge at node r, termed a reconvergent node. All nodes and edges in all the paths from s to r consist of a reconvergent region. Single stuck-at-fault test generation of combinational circuits free of reconvergent regions can be solved in time linear to the number of signal lines in the circuit. On the other hand, the presence of reconvergent regions may necessitate backtracking due to possible conflict at the fanout point of the reconvergent region. In [42], headlines were introduced and used to reduce the search space involved in the line justification process. Several other heuristics, like unique sensitization [42], basis nodes [43], and stem regions [44], have been developed to ease the task of test-pattern generation and fault simulation in the presence of reconvergent regions. It has been shown [45] that if a circuit C can be partitioned into blocks such that each block has no greater than k inputs and the graph, where each vertex represents a block and each edge represents a connection between blocks, has no reconvergence, then the test generation problem can be solved in polynomial time, provided $k = \log p(m)$, where m is number of signals in C and $p(m)$ is a polynomial of m. A technique using partitioning and resynthesis to reduce the number of reconvergent regions in a circuit has been shown to be successful in improving the testability of the circuit [46].

A. Effect of Sequential Reconvergent Regions on ATPG

Presence of reconvergent regions across time frames in sequential circuits may cause similar problems to sequential ATPG. We classify sequential reconvergent regions into two categories.

1) In an *equal-weight* reconvergent region, each path from the fanout point to the reconvergent node has k FF's (registers), $k > 1$, that is, each path has *weight* k.
2) When all the paths from the fanout point to the reconvergent node do not have an equal number of FF's (registers), the reconvergent region is called an *unequal-weight* reconvergent region.

Justifying two different values on nodes of two different paths of an equal-weight reconvergent region may lead to a conflict at the fanout point, similar to the case of combinational reconvergent regions. As explained later, however, such conflicts do not arise in the case of an unequal-weight reconvergent region because of varying time frames that have to be traversed to justify the two conflicting values. Unlike the case of combinational circuits, neither DFT nor ATPG heuristics have been developed to circumvent the problem of reconvergences in sequential circuits. Hence, in the proposed nonscan DFT scheme, we adopt the strategy of avoiding the formation of equal-weight reconvergences during the process of nonscan DFT.

B. Creation of Reconvergent Regions During Nonscan DFT and Effect on Testability

Since the number of primary inputs and primary outputs of a circuit is limited, and often less than the number of controllability and observability points that need to be added, the proposed nonscan DFT technique may introduce new reconvergent regions in the circuit.

Consider the data path shown partially in Fig. 5(a). Assume that each of the three EXU's is in loops, and all loops in the circuit pass through the outputs of the EXU's, such that the outputs X, Y, and Z have to be made k-level controllable and observable. Assume that there is a single primary input PI. A possible nonscan DFT technique to achieve one-level controllability is shown in Fig. 5(b), where the direct scheme is used to make both X and Y zero-level controllable and the register-file scheme is used to make Z one-level controllable. All loops through X and Y are zero-level controllable, and all loops through Z are one-level controllable, hence achieving one-level controllability for all loops in the circuit.

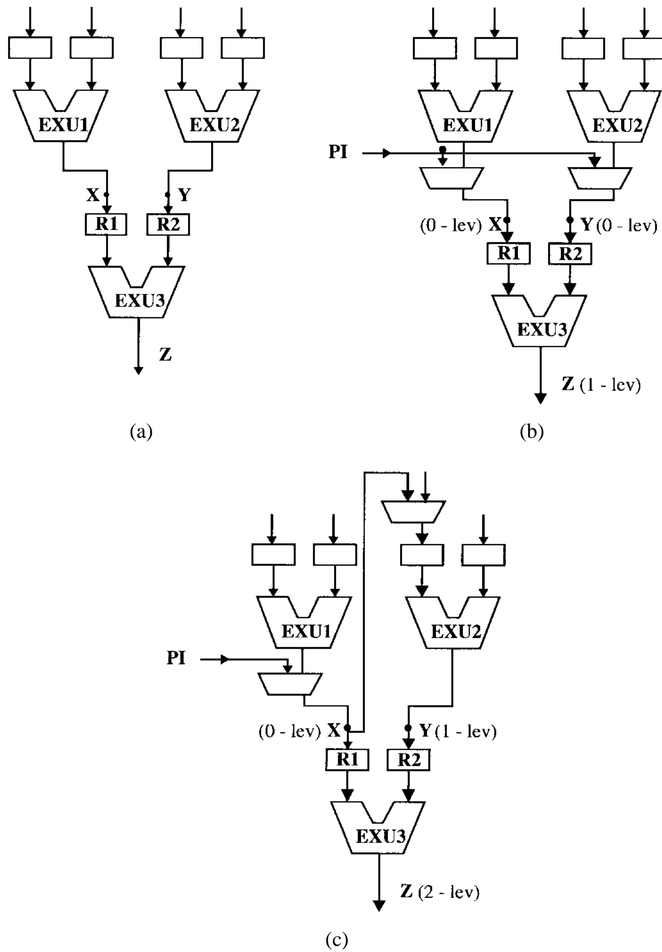


Fig. 5. Reconvergent regions formed while adding controllability point. (a) A partial data path (loops containing EXU1, EXU2, and EXU3 not shown for convenience). (b) Nonscan DFT for one-level controllability creates an equal-weight reconvergent region. (c) Nonscan DFT, achieving two-level controllability, creates an unequal-weight reconvergent region.

Since only one primary input bus PI was available, however, and PI was used to provide controllability to both the EXU output buses X and Y, the signals at X and Y are correlated. The registers R1 and R2 can only hold the same value at any given time frame. Moreover, the signals reconverge at the output of EXU3, Z. An equal-weight reconvergent region is formed from fanout point PI to reconvergent node Z, with each path from PI to Z containing one register (weight of one). Assuming the data path has a word size of n, each bus B is n bits wide, $B[1], \cdots, B[n]$. Hence, at the gate level, there are n equal-weight reconvergent regions of weight one, from fanout points $B[1], \cdots, B[n]$ to reconvergent nodes $Z[1], \cdots, Z[n]$.

Depending upon the functionality of the units present in the reconvergent paths, justification of values is inhibited to varying degrees. Consider the data path shown in Fig. 5(b). If EXU3 is an adder, attempting to justify an odd number—say, 11—at Z would require justification of two different numbers at registers R1 and R2 in the next time frame, which will not be possible due to a conflict at the input bus PI. If EXU3 is a fixed-point adder, $Z[1] = 0$ cannot be justified. Besides making $Z[1]$ stuck-at-zero untestable, the gate-level ATPG program has to backtrack every time it includes $Z[1] = 0$ in its objective. Similarly, if EXU3 is a floating-point adder, the last bit in the mantissa cannot be justified.

The adverse effect of equal-weight reconvergence may be more pronounced if the reconvergent node is the output of other types of functional units. For instance, if EXU3 is either a subtractor or a comparator, no value except zero can be justified at the output bus Z. On the other hand, if EXU3 is a division unit, only the value of one can be justified at the output Z. If EXU3 is a multifunctional unit, the justification problem at Z will be influenced by the types of functions supported by EXU3.

C. Avoiding Equal-Weight Reconvergence by Trading with Testability Level

Having discussed how equal-weight reconvergent regions can be created during the nonscan DFT process and how they adversely affect sequential ATPG, we now investigate how creation of such regions can be avoided. Since the number of primary inputs and primary outputs cannot be increased, creation of reconvergences may not be avoidable while adding controllability/observability points. Hence, we adopt the strategy of creating *unequal-weight* reconvergences while adding the test points.

Consider an alternative nonscan DFT of the design of Fig. 5(a) shown in Fig. 5(c). Instead of making both X and Y zero-level controllable using PI directly in Fig. 5(b), Y is made one-level controllable using the zero-level controllable signal X in a register-file-based scheme. Consequently, Z becomes two-level controllable. Hence, all loops in the circuit become two-level or less controllable, as opposed to the one-level controllability of loops achieved in the nonscan design in Fig. 5(b). The new nonscan design again introduces a reconvergent region from fanout point X to reconvergent node Z. But the reconvergent region is not equal weight, since the path from X to Z through register R1 has one register on it, while the path from X to Z through register R2 has two registers on it. Similarly, at the gate level, all paths from fanout point $X[i]$ to reconvergent node $Z[j]$ that pass through R1 have a weight of one, while all paths from $X[i]$ to reconvergent node $Z[j]$ that pass through R2 have a weight of two. Hence, the nonscan DFT scheme shown in Fig. 5(c) forms unequal-weight reconvergent region(s) while compromising the controllability level of the loops from one in Fig. 5(b) to two.

Unequal-weight reconvergent regions do not pose the justification problem posed by equal-weight reconvergent regions, since different values on the registers of the reconvergent EXU can be justified without causing conflicts at the fanout point. Consider the registers R1 and R2 on the two disjoint reconvergent paths of the unequal-weight reconvergence introduced in Fig. 5(c). Though both the registers get their values from X, and hence from PI, the desired value in R1 has to be justified in one time frame, while the desired value in R2 can be justified in two time frames. Hence, two different values like R1 $=$ 5 and R2 $=$ 6 can be justified by setting PI to six in the first time frame and PI to five in the second time frame. In general, registers of the different reconvergent paths in an unequal-weight reconvergent region can be justified to

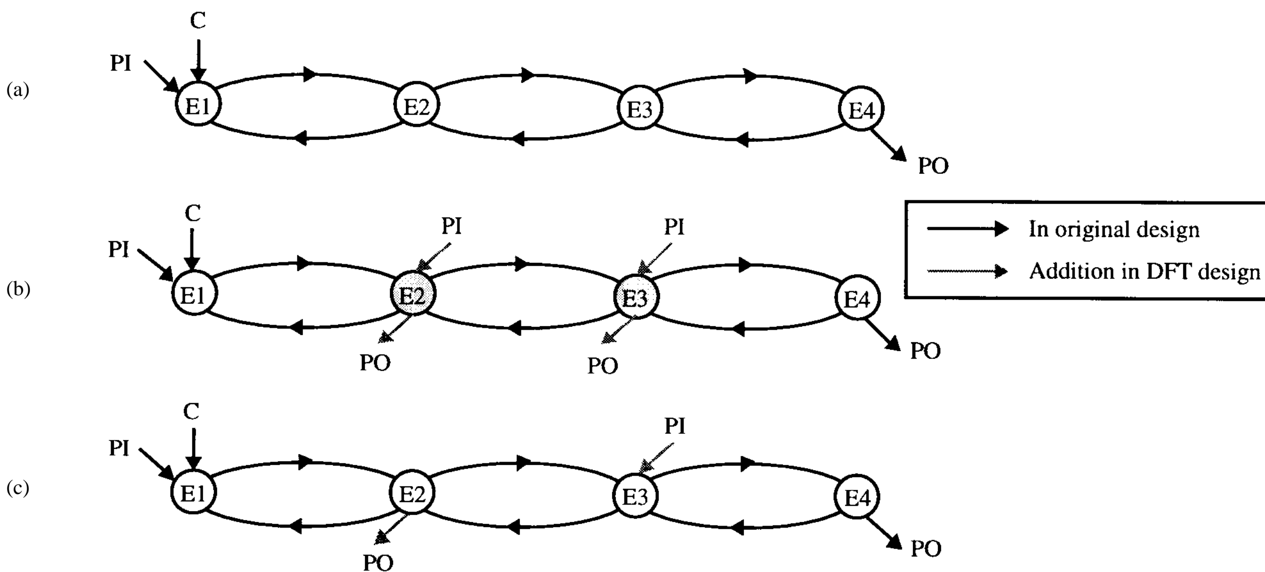


Fig. 6. Optimizing the number of cp's and op's. (a) EXU S-graph of original data path. (b) Lumped solution using two cp's, two op's. (c) Distributed solution using one cp, one op.

different values since they can be justified in different time frames.

The ability to justify different values at the registers of an unequal-weight reconvergent region enables justification of any value at the output of the reconvergent node in at most a number of time frames equal to the maximum weight of a reconvergent path. Assume that EXU3 is an adder in the unequal-weight reconvergence introduced in Fig. 5(c). To justify an odd value—say, 11—at the output of EXU3, Z, two different values—say, five and six—have to be justified at the registers R1 and R2, respectively. As explained before, this can be accomplished by setting PI to six and five in the first and second time frame, respectively, requiring a justification sequence of two clock cycles. Irrespective of the functionality of the execution unit EXU3, all values can be justified at the output of EXU3.

The technique presented here for avoiding equal-weight reconvergent regions encompasses a tradeoff between ease of sequential test generation (absence of equal-weight reconvergence) and increase in test length (presence of unequal-weight reconvergence and increase in level of controllability). While the presence of an equal-weight reconvergence produces limitations in justification, the presence of unequal-weight reconvergence increases the level of controllability and hence the number of test vectors needed for justification. The tradeoff can be better decided depending upon the functionality of the reconvergent EXU. For instance, if EXU3 is a comparator, creating an unequal-weight reconvergence as in Fig. 5(c) is preferable to creating a low-fault coverage circuit by introducing an equal-weight reconvergence as in Fig. 5(b).

Reconvergent regions can be similarly formed while adding observability points and can be similarly avoided by increasing the level of observability achieved. In the next two sections, we introduce two techniques—distributing controllability and observability points and using dual points—to minimize the test hardware overhead required to make circuits k-level testable.

V. Optimizing Test Hardware: Distributing Controllability and Observability Points

In the previous sections, we have always selected the same node (output of EXU) to simultaneously make it k-level controllable/observable. A loop L1 may be more controllable (have a node with less controllability level) than some other loop L2 in the original data path, however, while loop L2 may be more observable than loop L1 in the original data path. Instead of adding simultaneously controllability points (cp's) and observability points (op's) to selected nodes on both L1 and L2, it may be more economical to add an observability point to L1 and a controllability point to L2.

Consider the EXU S-graph shown in Fig. 6(a). Suppose we want to make all loops one-level controllable/observable, that is, produce a one-level testable design. If the same nodes are selected for adding both controllability and observability points simultaneously, we get the *lumped* solution shown in Fig. 6(b). The nodes E2 and E3 have been selected. A broken arrow from PI indicates the addition of a controllability point from a PI. Similarly, a broken arrow to a PO indicates the addition of an observability point. After the addition of the two controllability points and two observability points, all the loops in the data path are zero-level controllable/observable, thus making the modified data path zero-level testable.

Notice, however, that the loop E1 $\rightarrow$ E2 $\rightarrow$ E1 is already one-level controllable in the original data path in Fig. 6(a), since node E1 is one-level controllable. Similarly, the loop E3 $\rightarrow$ E4 $\rightarrow$ E3 is one-level observable in the original data path, since node E4 is one-level observable. Instead of attempting to make the loops controllable/observable simultaneously, it may be more cost effective to first add observability points to make all loops observable and subsequently add controllability points to make all loops controllable. In this way, the controllability/observability points will be added in a distributed manner and fewer points will suffice to make the design k-level testable.

For the example shown in Fig. 6(a), it would suffice to make the loops E1 → E2 → E1 and E2 → E3 → E2 one-level observable, which can be achieved by just adding one observability point to E2. Also, it would suffice to make the loops E2 → E3 → E2 and E3 → E4 → E3 one-level controllable, which can be achieved by a single controllability point added to E3. The resultant distributed solution, shown in Fig. 6(c), uses only one cp and one op, and hence is more economical than the lumped solution shown in Fig. 6(b), which uses two cp's and two op's.

VI. Dual Points to Optimize Test Hardware

This section introduces dual points as another powerful technique to optimize nonscan test hardware. A controllability point primarily enhances the controllability of a loop. An observability point primarily enhances the observability of a loop. However, a dual point is used for the dual purpose of enhancing the controllability of one loop while enhancing the observability of another loop. The following example illustrates the dual point and its advantages.

Definition 4: Let us assume that a loop L1 is k_1-level controllable and that another loop L2 is k_2-level observable. A dual point involves multiplexing the output of an EXU, which is k_1-level controllable in loop L1, with either an input register (register-file-based scheme) or the output (direct scheme) of another EXU, which is k_2-level observable in loop L2. The dual point simultaneously enhances the observability of loop L1 to $k_2 + 1$ (k_2 for direct scheme) and the controllability of loop L2 to $k_1 + 1$ (k_1 for direct scheme).

Consider the data path of the fourth-order IIR parallel filter shown in Fig. 7(a). For ease of understanding, the corresponding EXU S-graph is shown in Fig. 8. The original data path is very untestable, as shown by the results of running HITEC (row "Orig") in Table VIII. A nonscan zero-level testable design, using three controllability points and two observability points, is shown in Fig. 7(b). The test hardware added is shown in bold. The nonscan design has a very high test efficiency, as evidenced by the row "0-lev" in Table VIII.

A closer look at the data path in Fig. 7(a) reveals that the loops through the EXU's 1+ and 3+ can be made one-level controllable just by adding a constant zero to the left register of 1+ and 3+, respectively. Also, the loop through 6+ is already one-level observable. Hence, a more cost-effective nonscan design can be obtained using distributed controllability/observability points, as mentioned in Section V. To make the data path one-level testable, we would need two constants, two observability points from 1+ and 3+, respectively, and one controllability point to the output of 6+.

However, using dual points reduces the test hardware requirement further. Adding a constant to the left register of 1+ makes all loops through 1+ one-level controllable. A dual point added from 1+ to the left register of 3+, and another dual point added from 3+ to the right register of 6+ (with a constant added to the left register of 6+) makes the loops through 3+ two-level controllable and two-level observable, the loops through 1+ three-level observable, and the loops through 6+ three-level controllable. The resulting data path, shown in Fig. 7(c), is three-level testable. The test hardware added is shown in bold. Note that the hardware overhead for a dual point is the same as a controllability or an observability point. Hence, the dual-point solution is less expensive than the zero-level solution shown in Fig. 7(b), which employs controllability and observability points. In fact, the hardware overhead of the dual-point solution (row "3-lev") is 40% less than the overhead of the zero-level solution, as shown in Table VIII. Also, the dual-point solution has a very high test efficiency, 99%, as shown in row "3-lev" in Table VIII. We show the insertion of the test points and dual points on the corresponding EXU S-graph in Fig. 8, which makes it easier to see the test points and dual points and how they make the loops k-level controllable/observable.

VII. Algorithms to Add Test Hardware for k-Level Testable Data Paths

In this section, we introduce algorithms that add the minimal hardware possible to make all loops in the data path k-level controllable and k-level observable, for a user-specified value of k, using the nonscan DFT techniques discussed in the previous sections. We first propose an algorithm that uses distributed controllability and observability points. Next, we discuss modifications to use dual points to reduce the hardware overhead of the nonscan DFT approach. The minimum set of nodes whose breaking (that is, making the nodes zero-level controllable/observable) makes all loops k-level controllable/observable is termed the *k-level MFVS*. The special case is the (zero-level) MFVS, which is traditionally used in several partial-scan approaches [1], [2] to break all the loops directly, that is, to make all loops zero-level controllable and observable. As shown in the appendix, the general problem of finding the k-level MFVS is NP-complete. For both the nonscan DFT approaches (distributed test points and dual test points), we present both heuristic and optimal implicit enumeration-based solution.

A. Using Distributed Controllability/Observability Points

The key idea behind the algorithm is iteratively to select controllability and observability points that will ensure that all loops are k-level observable and controllable, using the lowest hardware cost. Since addition of a cp or op requires a new interconnect and a multiplexor, we consider that it is always preferable to add constants as a means of enhancing observability and controllability than to add either a cp or an op. Note that the number of loops in an EXU S-graph can be exponential, so it is not possible to enumerate them individually. Instead, at each step of the algorithm, we count the number of nodes in all loops (strongly connected components) that have either the level of controllability or the level of observability higher than required. Last, note that all nodes in the EXU S-graph have to be considered for addition of cp or op, not only the nodes in strongly connected components, as is the case when MFVS has to be found.

The input to the algorithm is the target data path, the maximum number of allowed cp's and/or op's, and the level k of controllability/observability desired by the user. The

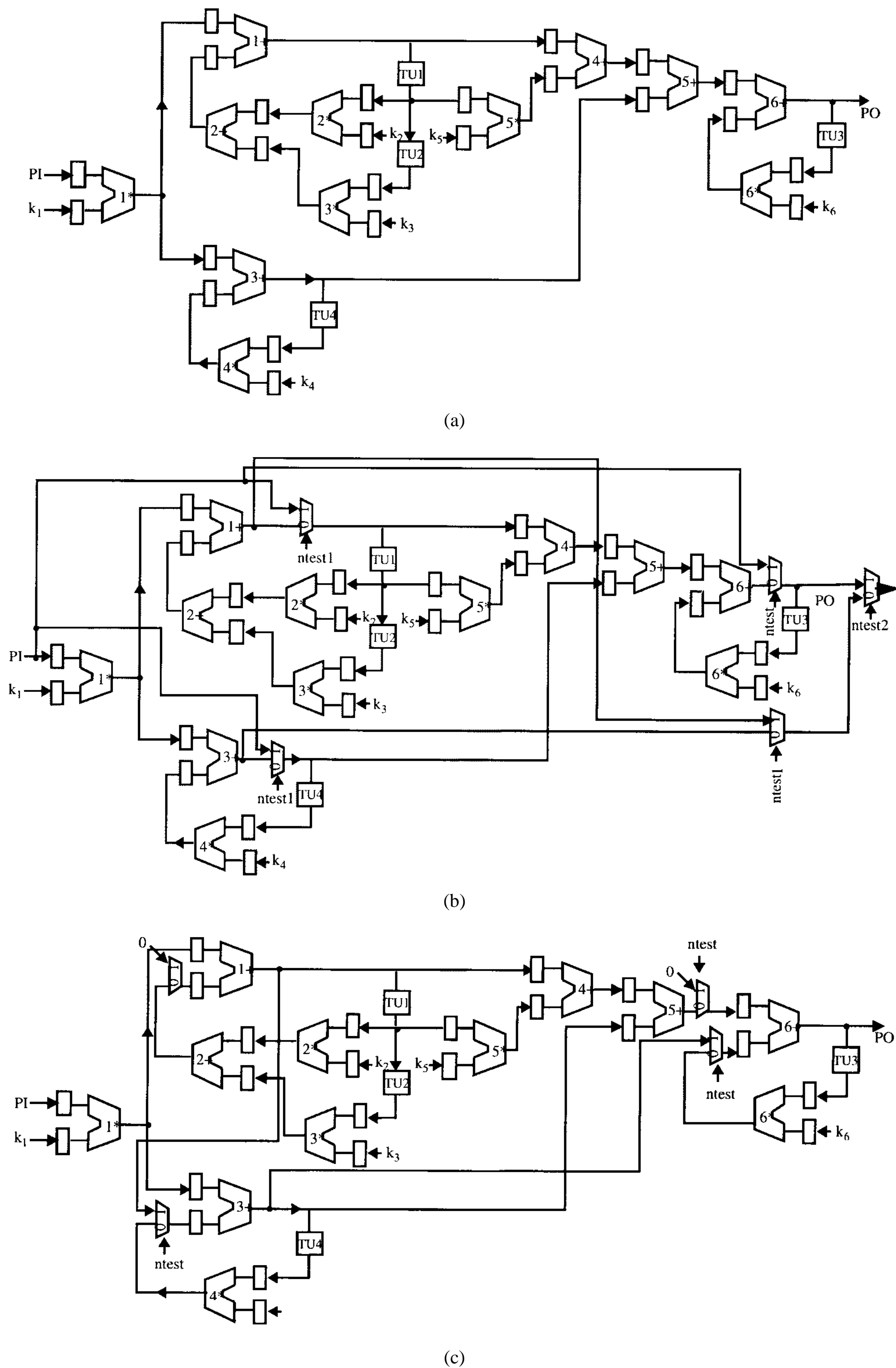


Fig. 7. Use of dual points to optimize test hardware. (a) RT-level data path of the fourth-order IIR parallel filter. (b) Zero-level testable design using three cp's, two op's, and five interconnects. (c) Three-level testable design using two dual points, two constants, and two interconnects.

pseudocode in Fig. 9 summarizes the heuristic algorithm used. A test point p refers to either a controllability point or an observability point.

Both test points and constants are evaluated according to the following objective function $E(p)$, where p is the test point or the constant being evaluated:

$$E(p) = \Delta[\text{LCM}(p)] + \Delta[\text{LOM}(p)] - \text{Reconv}(p).$$

The loop controllability measure (LCM) is equal to the number of nodes that are in loops whose controllability level is greater than k after adding p. Similarly, the loop observability measure (LOM) is equal to the number of nodes that are in loops whose observability level is greater than k after adding p. LCM is calculated as follows. At first, the controllability point or constant p is added, and the level of controllability

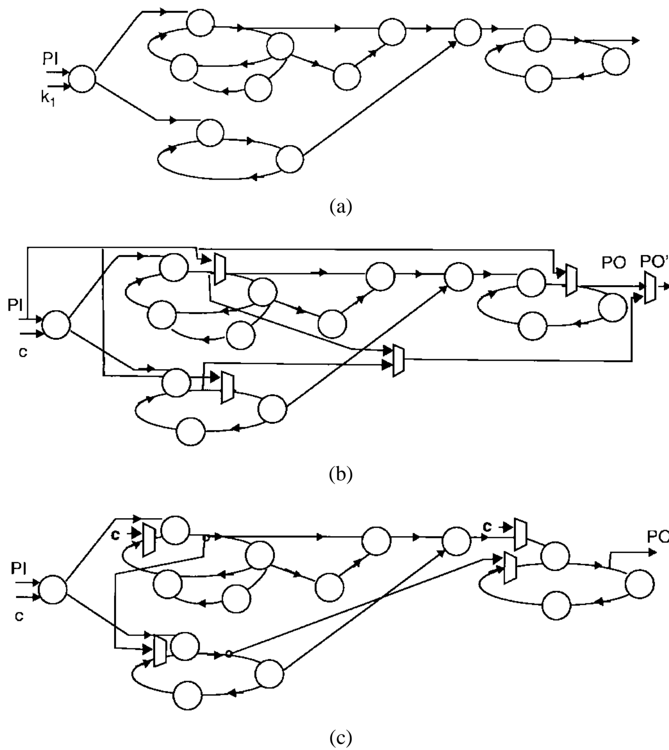


Fig. 8. Illustrating the example in Fig. 7 using EXU S-graph for (a) RT-level data path, (b) zero-level testable design using three controllability points, two observability points, and five interconnects, and (c) three-level testable design using two dual points, two constants, and two interconnects. The test points added, including interconnects, are shown in bold.

of each node in the EXU S-graph is calculated, using the equation in Section III-B. Next, a new S-graph is formed by deleting all nodes whose controllability level is less than or equal to k. Consecutive identification of strongly connected components (SCC's) in the new S-graph, and counting the number of nodes in the SCC's, gives the number of nodes that are in loops whose controllability level is greater than k, which is the LCM. The LOM is computed similarly. Δ denotes the change (decrease) in the LCM and LOM due to insertion of the candidate test point or constant. Reconv(p) cost reflects the penalty due to the possible introduction of equal-weight reconvergent regions in the data path by adding the test point p. It is equal to the number of equal-weight reconvergent regions formed by inserting the test point p. Currently, we are using an approximation, counting only the equal-weight reconvergent regions of weight one, that is, reconvergent regions going across two time frames.

A major advantage of operating on designs at the RT level, instead of at the gate level, is that the size of the optimization problem is significantly smaller than the size of the corresponding problem at the gate level. For example, consider the fourth-order IIR parallel filter shown in Fig. 7. As reported in Table II, while the number of nodes of the FF S-graph at the gate level is 440, the number of nodes of the RT-level register S-graph is 22, and the number of nodes of the RT-level EXU S-graph is only 16. The small size of instances of the optimization problem at the RT-level can often be utilized to solve the problem using algorithms with exponential asymptotic worst case run times, which guarantee the optimum solution. In spite of the asymptotic complexity, the run times are low for small instances like the EXU S-graph that the nonscan DFT algorithm needs to consider.

To minimize the hardware overhead in nonscan DFT designs, we use the following implicit enumeration branch and bound algorithm [47]. The algorithm starts by first applying the heuristic algorithm to provide the initial best solution. After that, all solutions are evaluated, one by one, in a lexicographical order. During this process, each solution is represented as a string, alphabetically ordered by the names of nodes where test points and constants are added. Each solution is evaluated only as long as its cardinality (number of test points used) is smaller than the current best solution.

B. Using Dual Points

To minimize hardware overhead using dual points, we just need to modify algorithm *add_test_points()*. In step 5 of each iteration, instead of evaluating the test points, candidate dual points are evaluated. We consider all pairs of nodes u, v in the EXU S-graph, such that u, v belong to different SCC's, to be candidate dual points. For each candidate dual point p, the required interconnect is added, and LCM(p), LOM(p), and Reconv(p) are calculated as before. The pair-wise evaluation is possible because unlike the FF S-graph at the gate level, an EXU S-graph at the RT-level has very few nodes, as reflected by Table II. In the next section, we report results of applying the proposed nonscan DFT technique and compare it with traditional scan-based DFT techniques.

VIII. Experimental Results

We applied the nonscan DFT algorithms presented in this paper to different types of data-path circuits, synthesized using the high-level synthesis system HYPER [39] from behavioral descriptions [48]. In this section, we report the results obtained on the following data paths:

1) fourth-order IIR cascade filter (4IIRcas);
2) speech filter (Speech);
3) fifth-order EWF;
4) fifth-order EWF, synthesized using high hardware sharing (EWFhigh);
5) fourth-order IIR parallel filter, synthesized using no hardware sharing.

Table I shows various parameters of the data paths obtained using HYPER [39]. The word size of the designs ("Bits") and the number of adders ("Add"), multipliers ("Mult"), registers ("Reg"), multiplexers ("Mux"), and interconnects ("Inter") are reported. Also, the number of transistor pairs needed for the final technology mapped circuit, using the SIS technology mapper [49], and the *lib2.genlib* standard cell SCMOS 2.0 library [50], is reported in the column "Area."

Table II shows the characteristics of the several possible S-graphs for each circuit. The columns "FF S-graph," "REG S-graph," and "EXU S-graph" show the number of nodes and the size of the MFVS for the gate-level S-graph of FF's, RT-level S-graph of registers, and EXU S-graph, respectively. The table shows the advantage of using the RT-level S-graphs in

```
add_test_points()
1.   create the EXU S-graph
2.   while (there exists a loop whose controllability/observability level > k)
3.       if (there is still an available test point)
4.          {for each vertex in S-graph
5.              E(p) ← evaluate test point(p), ∀ test points;
6.          select test point with highest E(p);
7.          add best test point;}
8.       else if (there exists a register file without a constant)
9.          {for each vertex
10.             E(p) ← evaluate constant(p);
11.         select constant with highest E(p);
12.         add best constant;}
13.      else {request more test points; EXIT;}
14.      update_the_number_of_nodes_in_remaining_SSC();}
```

Fig. 9. Pseudocode summarizing the heuristic algorithm used.

TABLE II
THE DIFFERENT S-GRAPHS OF THE DATA PATHS

Design	FF S-graph		REG S-graph		EXU S-graph	
	Nodes	MFVS	Nodes	MFVS	Nodes	MFVS
4IIRcas	220	60	11	3	9	2
Speech	220	20	11	1	9	1
EWF	368	240	23	15	6	3
EWFhigh	360	280	18	14	2	1
4IIRpar	440	60	22	3	16	3

terms of lower complexity. The advantage of selecting EXU's as the nodes to insert controllability/observability/dual points is shown by the lower MFVS of the EXU S-graph compared to the traditionally used FF S-graph or register S-graph.

The results of applying partial-scan and nonscan DFT techniques on the data paths in Table I are reported in the Tables III–VIII. The row "Orig" shows the original design. The rows "Opus" and "LR" show the partial-scan designs obtained by using OPUS [3] from the University of Illinois and LR [2] from the University of Iowa. The designs produced by the nonscan DFT techniques are presented in the subsequent rows, with the rows "2-lev," "1-lev," and "0-lev" representing the two-level, one-level, and zero-level testable designs, respectively.

For each of the designs, the column "Test Hardware" summarizes the test hardware that had to be added to the original design to make the circuit testable. In the case of partial-scan designs, the number of scan FF's needed is reported. In the case of each nonscan design, the number of constants c, the number of control points cp, the number of observability points op, or the number of dual points dp that had to be added to the original design is shown. For each scan and nonscan design, the test hardware overhead, in terms of extra transistor pairs used, is shown in the column "Overhead." The overhead for scan-based designs includes the extra cost (transistor pairs) of each scan FF compared to a normal FF, the cost to generate the scan clock, and the cost of buffers needed to distribute the scan clock to the scan FF's. Note that the hardware overhead does not include the area needed to route the extra interconnects needed for the DFT designs.

We ran the sequential test-pattern generator HITEC [41] on the original design as well as the scan and nonscan designs. The following tables show the results of running HITEC on each of the designs. The total number of faults and the faults aborted ("Abt") 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 redundant (that is, the percentage of faults not aborted). Also, the number of test vectors needed ("Vec"), and the test generation time taken ("Tgen") in CPU seconds on a Sparc2 are reported. Last, the test application time—that is, the number of clock cycles needed to apply the test vectors to the designs—is shown in the column "Tappl."

Tables III–VIII show the ability of the nonscan DFT techniques to make the highly untestable data paths very easily testable, with a significantly smaller test hardware overhead than the scan designs. Consider the results for the data-path circuit EWFhigh, shown in Table VI. The original data path has a very low test efficiency (2%). The scan designs achieve very high test efficiency, but the test hardware overhead is very high (3085 transistor pairs), and the test application time needed is 45 920 clock cycles. On the other hand, the nonscan DFT solutions produced by the proposed techniques achieve comparable test efficiency with significantly lower area overheads and test application times. For example, the one-level testable data path uses only 20 extra transistor pairs while still achieving a very high test efficiency of 97%. The zero-level nonscan solution achieves almost a 100% test efficiency while requiring only 282 extra transistor pairs, as compared to 3085 required by the scan designs. As expected, the nonscan solutions need significantly lower test application time than the scan designs. For example, only 218 clock cycles are needed to apply the test vectors to the zero-level design as compared to 45 920 clock cycles needed for the scan

TABLE III
4IIRcas: Cost and Effect of Several DFT Schemes

Type	Test Hardware	Overhead (Cells)	Faults Total	Faults Abt	FC%	TE%	Vec	Tgen (secs)	Tappl (cycles)
Orig	None	0	10004	9667	0.00	3.00	-	23884	-NA-
Partial-Scan DFT									
Opus	60 scan FFs	665	10004	3	96.86	99.97	156	226	9360
LR	60 scan FFs	665	10004	3	96.87	99.97	168	210	10080
Non-Scan DFT									
2-lev	2c	120	10086	239	94.49	97.63	109	6982	109
1-lev	1c,1cp,1op	349	10334	70	96.17	99.32	149	686	149
0-lev	2cp,1op	429	10448	5	96.95	99.95	268	292	268

TABLE IV
Speech Filter: Cost and Effect of Several DFT Schemes

Type	Test Hardware	Overhead (Cells)	Faults Total	Faults Abt	FC%	TE%	Vec	Tgen (secs)	Tappl (cycles)
Orig	None	0	8514	8186	0.00	4.00	-	19614	-NA-
Partial-Scan DFT									
Opus	20 scan FFs	225	8514	29	96.07	99.65	111	193	2220
LR	20 scan FFs	225	8514	16	96.25	99.81	131	160	2620
Non-Scan DFT									
1-lev	1c	57	8553	300	92.78	96.49	70	7491	70
0-lev	1cp	142	8676	10	96.38	99.88	149	412	149

TABLE V
EWF: Cost and Effect of Several DFT Schemes

Type	Test Hardware	Overhead (Cells)	Faults Total	Faults Abt	FC%	TE%	Vec	Tgen (secs)	Tappl (cycles)
Orig	None	0	9088	8879	0.00	2.00	-	27423	-NA-
Partial-Scan DFT									
Opus	240 scan FFs	2645	9088	0	97.99	100.00	118	209	28320
LR	240 scan FFs	2645	9088	9	97.89	99.90	137	223	32880
Non-Scan DFT									
3-lev	3c	96	9186	4180	52.50	54.43	52	18043	52
2-lev	1c,1cp,1op	256	9380	891	88.85	90.50	72	4668	72
1-lev	1cp,1op	224	9346	178	96.14	98.09	253	985	253
0-lev	3cp,3op(2ntest pins)	671	9798	296	95.10	96.97	278	1222	278

TABLE VI
EWF with High Hardware Sharing (EWFhigh): Cost and Effect of Several DFT Schemes

Type	Test Hardware	Overhead (Cells)	Faults Total	Faults Abt	FC%	TE%	Vec	Tgen (secs)	Tappl (cycles)
Orig	None	0	9375	9158	0.00	2.00	-	29220	-NA-
Partial-Scan DFT									
Opus	280 scan FFs	3085	9375	2	97.68	99.97	164	240	45920
LR	280 scan FFs	3085	9375	3	97.64	99.96	192	251	53760
Non-Scan DFT									
1-lev	1c	20	9397	271	94.78	97.11	218	2625	218
1-lev	1c,1op	161	9519	130	96.32	98.63	297	992	297
0-lev	1cp,1op	282	9657	20	97.54	99.79	347	250	347

implementation. Note that HITEC requires more CPU time to generate test patterns for the zero-level nonscan solution than for the corresponding scan solution for each design.

The experimental results also validate the effectiveness of the k-level controllable/observable loops measure introduced in this paper. The results show that it is not needed to make all the loops directly (zero-level) controllable/observable to achieve high test efficiency, as evidenced by the very high test efficiency reported for the k-level testable data paths, $k > 0$. Most significantly, the experimental results demonstrate the

TABLE VII
EWFHIGH (16 BIT): COST AND EFFECT OF SEVERAL DFT SCHEMES

Type	Test Hardware	Overhead (Cells)	Faults Total	Faults Abt	FC%	TE%	Vec	Tgen (secs)	Tappl (cycles)
Orig	None	0	7608	7478	0.48	2.00	3	22879	-NA-
Non-Scan DFT									
Opus	224 scan FFs	2469	7608	2	98.75	99.97	161	132	36064
LR	224 scan FFs	2469	7608	3	98.73	99.96	195	92	43680
Non-Scan DFT									
1-lev	1c	16	7626	288	95.00	96.22	163	2253	163
1-lev	1c,1op	129	7724	89	97.60	98.84	402	592	402
0-lev	1cp,1op	226	7834	21	98.54	99.73	324	157	324

TABLE VIII
4IIRPAR: COST AND EFFECT OF SEVERAL DFT SCHEMES

Type	Test Hardware	Overhead (Cells)	Faults Total	Faults Abt	FC%	TE%	Vec	Tgen (secs)	Tappl (cycles)
Orig	None	0	14215	13646	0.00	4.00	-	77527	-NA-
Partial-Scan DFT									
Opus	60 scan FFs	665	14215	5	96.80	99.96	98	913	5880
LR	60 scan FFs	665	14215	3	96.82	99.97	108	399	6480
Non-Scan DFT									
3-lev	2c,2dp	347	14497	145	95.90	98.99	151	13919	151
0-lev	3cp,2op	565	14953	13	96.91	99.91	190	602	190

feasibility of producing nonscan testable data paths, which can be tested at-speed, with marginal area overheads. Note that while zero-level testability can be always achieved by multiplexing directly with inputs/outputs, achieving k-level testability, for each $k > 0$, may or may not always be possible, depending upon the level of controllability/observability for the existing registers. Hence, the one-level and two-level testable data paths are missing from Table VIII.

As expected, the test efficiency of a $(k-1)$-level testable solution is always better than a k-level testable solution for all design examples, except EWF. The anomaly in the case of EWF, where the one-level testable solution has a higher test efficiency than the zero-level testable solution, as shown in Table V, can be explained by the analysis of reconvergent regions presented in Section IV. To achieve zero-level testability, several equal-weight reconvergences are introduced. Consequently, unlike the zero-level solutions for other design examples, the zero-level testable solution for EWF, as reported in row "0-lev," had only a 96.97% test efficiency. Formation of the equal-weight reconvergent regions can be avoided by the one-level testable solution. Hence, we see that the test efficiency of the one-level testable solution is better than the zero-level testable solution for the EWF design.

IX. CONCLUSIONS

We have proposed new DFT techniques to make RT-level data paths testable. As opposed to the earlier DFT techniques for data paths, the proposed techniques do not use scan, and the resultant designs can be tested at-speed, enhancing the chances of detecting defective chips. The effectiveness of the proposed DFT techniques is largely due to the effectiveness of a new testability measure introduced in this paper, which eliminates the need to break loops explicitly. Experimental results demonstrate the feasibility of generating data paths with high test efficiency without using scan and with significantly lower test area overhead and test application time than the corresponding partial-scan designs.

APPENDIX

Here, we provide proofs of the following two theorems.

Theorem 1: For an RT-level circuit that uses a dedicated register-file model, the MFVS of an EXU S-graph is a lower bound of the MFVS of the register S-graph.

Proof: For an RT-level circuit that uses a dedicated register-file model, each register sends data to only one execution unit, while each execution unit can receive/send data from/to multiple registers. Hence, if breaking (scanning) a register R breaks a set of loops L, breaking (inserting test point to) the corresponding execution unit E_R will also break the same set of loops. Now consider an MFVS set of the register S-graph $V_R = \{R1, R2, \cdots, Rn\}$ of cardinality n. An MFVS set of the corresponding EXU S-graph can be constructed as $V_E = \{E_{R1}, E_{R2}, \cdots, E_{Rn}\}$, where E_{Ri} is the corresponding execution unit for register R_i. Note that the cardinality of V_E is at most n but can be less since multiple registers in V_R can send data to the same execution unit in V_E. Hence, $|V_E| \leq |V_R|$. □

We next prove that finding the k-level MFVS is a computationally intractable problem. We start by posing the problem in the standard form [51] for studying its computational complexity.

PROBLEM: k-level feedback vertex set.

INSTANCE: Directed graph $G_k = (V_k, A_k)$, positive integer $M \leq |V_k|$, positive integer k.

QUESTION: Is there a subset $V'_k \subseteq V_k$ with $|V'_k| \leq M$ such that V'_k contains vertices which make every directed cycle in G_k k-level observable and k-level controllable.

We present the following theorem regarding the computational complexity of the k-level feedback vertex set problem.

Theorem 2: The k-level feedback vertex set is an NP-complete problem.

Proof: We use Karp's polynomial transformation method to prove that the k-level feedback vertex set is an NP-complete problem [52]. In particular, we employ the local replacement technique [51, pp. 66–72].

It is easy to see that the k-level feedback vertex set belongs to NP, since a nondeterministic algorithm needs only to guess a subset of vertices A′ of A and check in polynomial time that each cycle in G has at least one point that is k-level observable and k-level controllable. For the checking, it is sufficient to conduct the following two-phase polynomial time procedure. In the first phase, the controllability/observability level of each node is calculated as shown in Section III-B. Next, each cycle is checked to see whether it has at least one point k-level observable by finding all points that are k-level observable and deleting them. If the resulting graph is acyclic, it can be concluded that indeed all cycles are k-level observable. Checking whether a given graph is acyclic can be done in polynomial time using the standard reverse topological ordering algorithm [53]. In the second phase, the same procedure is used to establish that each cycle has at least one point that is k-level controllable.

To complete the proof, we now polynomially transform a known NP-complete problem, the feedback vertex set problem, to the k-level feedback vertex set problem.

PROBLEM: feedback vertex set.

INSTANCE: Directed graph $G = (V, A)$, positive integer $M \leq |V|$.

QUESTION: Is there $V' \subseteq V$ with $|V'| \leq M$ such that V' contains at least one vertex from every directed cycle in G.

We transform an arbitrary instance G of the feedback vertex set to the corresponding instance of the k-level feedback vertex set in the following way. First, we form L copies of G,, where $L \geq |V|$. All nodes in G, and in the copies, form the set of nodes in G_k. Each node in G is connected to its corresponding nodes in all L copies. The connection from a node in G to the corresponding nodes in all copies is such that it establishes k-level observability and k-level controllability of the corresponding node in the copy. This is achieved by connecting each node to the corresponding nodes using two one-directional connections going through $(k-1)$ nodes.

We show that the instance of k-level feedback vertex set in G_k has a solution of size M if and only if the instance of the feedback vertex set problem has a solution in G.

Suppose first that a feedback vertex set problem has a solution that contains subset V' in V, of cardinality $|V'| \leq M$. By construction, from vertices of the subset V', their corresponding vertices in all copies are k-level controllable and observable. So, in the subgraph G of G_k, all cycles are zero-level controllable and observable, and in the rest of G_k, all cycles are k-level controllable and observable. Hence, V' is a k-level feedback vertex set of cardinality smaller than M.

Suppose now that the k-level feedback vertex set problem has a solution that contains subset V', of cardinality $|V'| \leq M$. The key observation is that the subset of nodes in V' that is in G, denoted by V'_G, is also a solution to the k-level feedback set problem. This is so because the nodes from any copy of G cannot be used to establish k-level observability and k-level controllability in any other copy of the L copies. Since there are more copies than nodes in the feedback vertex set, there is at least one copy in which k-level observability and k-level controllability is established using only nodes from G. Since all other copies are in an identical relationship to G, the nodes from G are indeed sufficient to solve k-level feedback set problem.

Note that in order to establish k-level observability/controllability in a copy, zero-level observability/controllability is required in G. So V'_G is a feedback vertex set in G with cardinality smaller or equal to M. Therefore, V' is a feedback vertex set in G with cardinality $|V'| \leq M$. □

It is easy to see that the generalization when the level of observability and the level of controllability are not identical does not alter the proof.

ACKNOWLEDGMENT

The authors wish to acknowledge V. Gangaram for his help in the project and several helpful discussions.

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. Reddy and R. Dandapani, "Scan design using standard flip-flops," *IEEE Design Test Comput. Mag.,* pp. 52–54, Feb. 1987.
[5] S. Narayanan and M. Breuer, "Reconfigurable scan chains: A novel approach to reduce test application time," in *Proc. ICCAD,* 1993, pp. 710–715.
[6] P. Maxwell, R. Aitken, V. Johansen, and I. Chiang, "The effect of different test sets on quality level prediction: When is 80% better than 90%?" in *Proc. Int. Test Conf.,* Oct. 1991, pp. 358–364.
[7] J. Heyes and A. Friedman, "Test point placement to simplify fault detection," in *Proc. 1973 Symp. Fault-Tolerant Computing (FTCS),* 1973, pp. 73–78.
[8] K. Saluja and S. Reddy, "On minimally testable logic networks," *IEEE Trans. Comput.,* vol. C-23, no. 11, pp. 1204–1207, 1974.
[9] B. Krishnamurthy, "A dynamic programming approach to the test point insertion problem," in *Proc. 24th Design Automation Conf.,* 1987, pp. 695–705.
[10] I. Pomeranz and Z. Kohavi, "Polynomial complexity algorithms for increasing testability of digital circuits by testing-module insertion," *IEEE Trans. Comput.,* vol. 40, no. 11, pp. 1198–1212, 1991.
[11] I. Pomeranz and S. Reddy, "Design-for-testability for path delay faults in large combinational circuits using test points," in *Proc. 31st Design Automation Conf.,* 1994, pp. 358–364.
[12] J. Stewart, "Future testing of large LSI circuit cards," in *Semiconductor Test Symp.,* 1973, pp. 6–17.
[13] T. Gheewala, "CrossCheck: A cell based VLSI testability solution," in *Proc. DAC,* 1989, pp. 706–709.

[14] S. Chandra, T. Ferry, T. Gheewala, and K. Pierce, "ATPG based on a novel grid-addressable latch element," in *Proc. DAC,* 1991, pp. 282–286.
[15] V. Chickermane, E. Rudnick, P. Banerjee, and J. H. Patel, "Non-scan design-for-testability techniques for sequential circuits," in *Proc. Design Automation Conf.,* June 1993, pp. 236–241.
[16] E. Rudnick, V. Chickermane, and J. Patel, "An observability enhancement approach for improved testability and at-speed test," *IEEE Trans. Computer-Aided Design,* vol. 13, pp. 1051–1056, Aug. 1994.
[17] S. S. K. Chiu and C. Papachristou, " A built-in self-testing approach for minimizing hardware overhead," in *Proc. Int. Conf. Computer Design,* 1991, pp. 282–285.
[18] L. Avra, " Allocation and assignment in high-level synthesis for self-testable data paths," in *Proc. Int. Test Conf.,* 1991, pp. 463–472.
[19] L. Avra and E. McCluskey, "Synthesizing for scan dependence in built-in self-testable designs," in *Proc. Int. Test Conf.,* Oct. 1993, pp. 734–743.
[20] I. Harris and A. Orailoğlu, "Microarchitectural synthesis of VLSI designs with high test concurrency," in *Proc. Design Automation Conf.,* June 1994, pp. 206–211.
[21] 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.
[22] A. Majumdar, R. Jain, and K. Saluja, "Incorporating testability considerations in high-level synthesis," *J. Electron. Testing: Theory Appl.,* pp. 43–55, Feb. 1994.
[23] T. C. Lee, W. Wolf, N. K. Jha, and J. M. Acken, "Behavioral synthesis for easy testability in data path allocation," in *Proc. Int. Conf. Computer Design,* 1992, pp. 29–32.
[24] T. C. Lee, W. H. Wolf, and N. K. Jha, "Behavioral synthesis for easy testability using data path scheduling," in *Proc. Int. Conf. Computer-Aided Design,* 1992, pp. 616–619.
[25] 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.
[26] S. Dey, M. Potkonjak, and R. Roy, "Exploiting hardware sharing in high level synthesis for partial scan optimization," in *Proc. Int. Conf. Computer-Aided Design,* Nov. 1993, pp. 20–25.
[27] S. Dey and M. Potkonjak, "Transforming behavioral specifications to facilitate synthesis of testable designs," in *Proc. Int. Test Conf.,* Oct. 1994, pp. 184–193.
[28] 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.
[29] C.-H. Chen, C. Wu, and D. G. Saab, "BETA: Behavioral testability analysis," in *Proc. Int. Conf. Computer-Aided Design,* Nov. 1991, pp. 202–205.
[30] P. Vishakantaiah and J. Abraham, "High level testability analysis using VHDL descriptions," in *Proc. EDAC,* Feb. 1993, pp. 170–174.
[31] C. H. Gebotys and M. I. Elmasry, "Integration of algorithmic VLSI synthesis with testability incorporation," *IEEE J. Solid-State Circuits,* vol. 24, pp. 458–462, Apr. 1989.
[32] 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.
[33] 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.
[34] 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.
[35] 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.
[36] K. Wagner and S. Dey, "High level synthesis for testability: A survey and perspective," in *Proc. Design Automation Conf.,* June 1996, pp. 131–136.
[37] S. Dey, V. Gangaram, and M. Potkonjak, "A controller-based design-for-testability technique for controller-data path circuits," in *Proc. Int. Conf. Computer-Aided Design,* Nov. 1995, pp. 534–540.
[38] H. D. M. *et al.,* "Synthesis of DSP systems at Leuven," in *Proc. IEEE ICCD,* 1987, pp. 133–145.
[39] J. Rabaey, C. Chu, P. Hoang, and M. Potkonjak, "Fast prototyping of data path intensive architectures," *IEEE Design Test Comput. Mag.,* pp. 40–51, 1991.
[40] D. Patterson and J. Hennessy, *Computer Architecture: A Quantitative Approach.* San Mateo, CA: Kaufman, 1989.
[41] T. M. Niermann and J. H. Patel, "HITEC: A test generation package for sequential circuits," in *Proc. EDAC,* 1991, pp. 214–218.
[42] H. Fujiwara and T. Shimono, "On the acceleration of test generation algorithms," *IEEE Trans. Comput.,* vol. C-32, pp. 1137–1144, Dec. 1983.
[43] T. Kirkland and M. Mercer, "A topological search algorithm for ATPG," in *24th Design Automation Conf. (ACM/IEEE),* June 1987, pp. 502–508.
[44] F. Maamari and J. Rajski, "A method of fault simulation based on stem regions," *IEEE Trans. Computer-Aided Design,* vol. 9, pp. 212–220, Feb. 1990.
[45] H. Fujiwara, "Computational complexity of controllability/observability problems for combinational circuits," *IEEE Trans. Comput.,* vol. 39, pp. 762–767, June 1990.
[46] S. Dey, F. Brglez, and G. Kedem, "Corolla based circuit partitioning and resynthesis," in *Proc. 27th ACM/IEEE Design Automation Conf.,* June 1990, pp. 607–612.
[47] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization Algorithms and Complexity.* Englewood Cliffs, NJ: Prentice-Hall, 1982.
[48] R. Haddad and T. Parsons, *Digital Signal Processing: Theory, Applications and Hardware.* New York: Computer Science, 1991.
[49] 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, pp. 328–333.
[50] S. Yang, " Logic synthesis and optimization benchmarks, user guide version 3.0," in *Proc. Int. Workshop Logic Synthesis,* Microelectronics Center of North Carolina, Research Triangle Park, NC, May 1991.
[51] M. Garey and D. Johnson, *Computers and Intractibility: A Guide to the Theory of NP-Completeness.* San Francisco, CA: Freeman, 1979.
[52] R. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations,* R. Miller and J. Thatcher, Eds. New York: Plenum, 1972, pp. 85–103.
[53] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms.* Cambridge, MA: MIT Press, 1990.

Sujit Dey (S'90–M'91) received the Ph.D. degree in computer science from Duke University, Durham, NC, in 1991.

From 1991 to 1997, he was with C&C Research Laboratories, NEC USA, Princeton, NJ, where he was a Senior Research Staff Member. In 1997, he joined the University of California, San Diego, where he is an Associate Professor in the Electrical and Computer Engineering Department. He also was a Research Intern with the Microelectronics Center of North Carolina. He has been involved in the research and development of computer-aided design tools to support high-level design and test methodologies, focusing on low-power and high-performance systems. He currently is involved in developing design, validation, and test methods for core-based system-on-chips. He has published more than 60 journal and conference papers and has received seven U.S. patents. He is a coauthor of *High Level Power Analysis and Optimization* (Norwell, MA: Kluwer, 1997). He has served on the program committees of several conferences, including the International Conference on Computer-Aided Design, the International Conference on Computer Design, the International Test Conference, and the VLSI Test Symposium. He is the Guest Editor of the *ACM Transactions on Design Automation of Electronic Systems* special issue on high-level design validation and test, to be published.

Dr. Dey received Best Paper awards from the 31st Design Automation Conference in 1994 and the 11th VLSI Design Conference in 1998. He also has received several best paper nominations. He is the General Chair of the 1998 IEEE International High Level Design Validation and Test Workshop and the Program Chair of the 1998 IEEE International Workshop on Test Synthesis.

Miodrag Potkonjak (S'90–M'91) received the Ph.D. degree in electrical engineering and computer science from the University of California, Berkeley, in 1991.

In September 1991, he joined C&C Research Laboratories, NEC USA, Princeton, NJ. Since 1995, he has been an Assistant Professor in the Computer Science Department at the University of California, Los Angeles. His research interests include intellectual property protection, system core-based design, functional verification, collaborative design, integration of computations and communications, and experimental algorithmics.