10th IEEE Workshop on RTL and High Level Testing (WRTLT'09), pp. 6-15, Nov. 2009.

A DFT Method for Functional Scan at RTL

Marie Engelene J. Obien and Hideo Fujiwara
Graduate School of Information Science, Nara Institute of Science and Technology
Kansai Science City 630-0192, Japan
{obien-j, fujiwara}@is.naist.jp

Abstract — *F-scan, a novel design-for-testability (DFT) method for high-level description of circuits, provides an effective alternative to current scan-based DFT approaches. By optimally utilizing available functional elements and paths for test, hardware overhead is reduced without compromising fault coverage and test application time. Experimental results show its comparison with full scan.*

Keywords — scan-based DFT, functional RTL circuits, high-level testing, assignment decision diagrams

1. Introduction

As VLSI Design becomes more complicated due to the trends of minimizing chip size and maximizing speed, the importance of testing has increased to ensure the quality of electronic consumer products. Several testing concerns such as the testability of sequential circuits have been focused on because despite the advancements, the problem of sequential test generation remains to be a difficult one. In order to reduce the complexity of sequential automatic test pattern generation (ATPG), various design for testability (DFT) approaches have been proposed such that the circuit structure and functionality change during test mode to allow easier testing. The most popular approach is scan design [1]. This technique increases the testability of sequential circuits considerably and, of the scan-based methods up to date, Full Scan is considered to be the most popular because it effectively reduces the sequential circuit test generation problem into a combinational one.

Full scan-based methodologies have been initially applied at the gate-level, wherein each flip-flop is converted to scan flip-flop. Gate-level full scan, however, has two critical disadvantages: 1) large test hardware overhead (elements added per FF to create scan FF) and 2) long test application time (for shifting test vectors through the scan chain). These penalties prove full scan to be very costly, especially for high-volume, low-cost applications.

One way to reduce area overhead is to use partial scan, however, its resulting fault coverage is lower than full scan. Thus, in recent years, full-scan based methods applicable to circuits at a higher level (i.e., register-transfer level or RTL) have been proposed [4]-[8]. At this level, the number of primitive elements in the circuit is reduced, thus needing lesser DFT elements to be augmented for testability.

The new DFT technique that we propose in this paper is called *F-scan*, which is applicable to register-transfer level (RTL) circuits. We convert the functional RTL description to assignment decision diagrams (ADD) before applying F-scan because this makes ATPG faster due to the determination of the test environment [17].

In gate-level full scan, in order to put flip-flops in scan chains, additional interconnect and multiplexers are automatically augmented. F-scan improves this by efficiently organizing *F-scan-paths* with the least area overhead cost as possible. Every register in the circuit is organized in an F-scan-path by maximizing the use of available functional logic and paths to be used for F-scan, hence keeping hardware overhead due to test at the minimum. Moreover, single F-scan-paths automatically allow parallel and simultaneous scan (dependent on the bit width), thus minimizing test application time as well. For further reduction, we also prioritize the use of multiple F-scan-paths, whenever readily available (dependent on the available primary inputs and outputs). The new concepts and methodology to create *F-scannable circuits* are provided in this paper.

The rest of the paper is as follows. In Section 2, several previous works are reviewed and the merits of our method are discussed. F-scan and other preliminary concepts such as ADDs are introduced in Section 3. We describe the details of F-scan design methodology in Section 4. We also explain the procedure for test generation in Section 5. The experimental results are provided in Section 6 and the conclusion in Section 7.

2. Previous work

The basic scheme of F-scan can be traced back to the proposed method of Lin et al., which establishes scan paths using existing functional logic [2]. To connect "free-scannable" flip-flops through functional logic, they add

constraints at some primary inputs or insert test points. However, starting the method at the gate-level can lead to very high computation cost in identifying and modifying large number of functional logic. Area overhead is also greater.

There are various methods that try to solve the sequential ATPG problem at a higher level of abstraction since this solves the cost problems given by gate-level approaches. Here, we discuss scan-based techniques [3]-[8] and non-scan approaches [9]-[15] that try to utilize existing functional logic.

Historically, Gupta et al. [3] introduced an approach to RTL DFT, which is a structured partial scan design that reduces area overhead by converting only the selected flip-flops into scan flip-flops. However, full-scan-based approaches assure stronger testability of circuits. H-Scan [4, 6] is a full-scan-based technique that utilizes paths between registers, but only through multiplexers. Hence, other possible functional units that can be used for test are not maximally utilized. On the other hand, orthogonal scan [5] uses data path flow as scan path. This concept is comparable with our approach in using F-paths for scan, however, orthogonal scan requires multiple test configurations because it uses hold functions through load enable. *Hold function* is a logic that causes a register to hold the same value when the function is activated. This is necessary when a functional logic is shared by two scan paths because it allows scanning-in and -out of vectors from these paths one at a time, thus allowing the shared element to be used for testing. Our study does not employ this kind of function (with the state register exception) because of the disadvantages of adding extra pins for controlling multiple paths during test and the expected longer test application time because simultaneous scan-in and -out cannot take place. Although we use some sort of initialization to scan-in the state register value first, which is a kind of hold function, we do not use hold whenever a functional operation is shared by candidate F-scan-paths.

Our work can be closely associated to Huang's RTL scan [7] and Dependency scan (or D-scan) [8]. Huang et al. [7] provided an effective approach for RTL scan by arranging registers in scan chains through cost rules, which ensure the lowest possible area overhead for the circuit. Though this method tries to exploit available functional logic as much as possible without the use of hold functions, mask function is not considered. A *mask function* can be applied to operation logic, wherein the value from one input can be passed through the output by masking the other inputs. This function further reduces area overhead, which is a DFT element widely used by our method. Moreover, we extend the function to mask modulo operation, which has not yet been proposed in any previous works. D-scan [8], on the other hand, extends H-Scan by reducing the circuits augmented to control scan paths for test. *Thru functions* (logic that allow values to pass through hardware modules) with predetermined control signals are used for scan paths in the circuit. This means that functional logic is also utilized whenever possible to realize the scan paths. This work, however, utilizes hold functions to handle multiple paths that share the same thru function, which, as discussed previously, is not used by our method.

Similar techniques in non-scan DFT techniques are also related with F-scan. Takabatake et al. [12] proposed a method, which uses *thru modules* (similar to thru functions) to make data paths weakly testable. The method, since based on weak testability, does not exactly pass any value through hardware modules but only some. Although effective for some circuits, this does not guarantee high fault coverage for any arbitrary circuit. Improvements are given by the following non-scan DFT techniques based on strong testability [13]-[15] that also provide complete fault efficiency aside from reducing area overhead. Wada et al. [13] introduced strong testability (I-path) for data paths. In [14], Ohtake et al. completed the method by dealing with both controller and data paths, although the approaches are different for both. A test plan generator is also presented for faster test generation. The drawback though is the need to isolate the controller from the data path using multiplexers. This method results in performance degradation aside from the area overhead problem. Further improvements are given in [15], which presented partially strong testability, wherein there is no need for isolation between controller and data path, thus the number of multiplexers added for test is reduced. However, this approach requires a test controller to be added to the circuit. Moreover, the said methods are applicable to structural description of circuits, unlike our method, which can handle functional RTL. Since most of the designers are increasingly using functional description of circuits, this is an advantage for our proposed technique. Also, our method deals with the circuit in assignment decision diagrams (ADD), which represent both the controller and data path parts similarly. Thus, the application of the DFT method and test is consistent for the entire circuit.

Our novel approach to functional RTL scan, F-scan, improves all of the said works in terms of area overhead and prioritizes to create F-scan-paths that will lead to the least possible scan time as will be discussed in the next section.

3. F-scan

In order to define *functional scan*, we first give a brief introduction about assignment decision diagrams and the nine symbol algebra used for test environment generation. Then, we describe new concepts such as functional scan-in and scan-out, F-path, F-scan-path, and F-scannable circuit.

3.1. ADD and the nine symbol algebra

Assignment Decision Diagram or ADD shown in Figure 1 is a representation developed for high-level synthesis that is complete, efficient, and partially unique. It can be used to describe functional RTL circuits in which the controller part and the data path part are consistently represented.

ADD is an acyclic graph which consists of four parts: 1) the assignment value; 2) the assignment condition; 3) the assignment decision; and 4) the assignment target. There are four types of nodes needed to represent it: a) read nodes and b) write nodes (primary inputs or PI and outputs or PO, registers, or constants), c) operation nodes (arithmetic and logic), and d) assignment decision nodes or ADN (multiplexers) [10]. ADNs work such that when one of the condition inputs becomes true, the value of the corresponding data value input will be passed through [16].

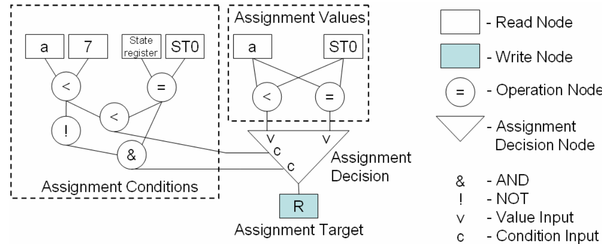


Figure 1. The assignment decision diagram.

The concept of *functional scan* uses the following nine symbol algebra used by Ghosh [17] for automatic test pattern generation (ATPG) of ADD circuits.

1. *Cg* (general controllability) of a register means it can be controlled to any arbitrary value.
2. *Cq* (controllability to a constant) of a register means it is controllable to any fixed constant value. This subsumes *C0*(controllability to zero), *C1*(controllability to one), and *Ca1*(controllability to all one).
3. *O* (observability) of an RTL variable is the ability to observe fault at a variable.
4. *Cs* (controllability to a state) is similar to *Cg* but is applied to state registers to control to a particular state.
5. Other symbols are *Cz* (controllability to the Z value) and *O'* (complement observability), but these are not used for our study.

In Figure 2, functional scan is illustrated with the use of these symbols.

3.2. Functional scan

We introduce the new concepts of functional scan by describing the means of functionally scanning-in and -out test patterns in an ADD circuit. In Figure 2(a), we see that any arbitrary value can pass through available operation nodes and ADN.

For operation nodes, as long as the other inputs to the node (side inputs) are constants, it can be used for functional scan. For addition and subtraction, the side input requirement is *Cq*. For multiplication and division, even though we can use *Cq* except for *C0* when it is already available, whenever masking is done, we allow *C1* and *Ca1* only for simplicity in the computation of test vectors for test application. For modulo, we use *Cq* such that in $c = a \bmod b$, $b = 2^n$, where n is chosen such that b is the maximum possible value within its range. Bitwise, we can describe this such that the most significant bit is 1 and the rest are 0. Moreover, ADN can be used for functional scan by manipulating its control inputs to *C0* and *C1*.

Similarly, we can observe through operation nodes and ADNs as shown in Figure 2(b). Given these, we have the following definitions.

Definition 1. *Fsi(n)* is satisfied if any value within a specified range can be functionally scanned-in to register node *n* (output of operation node or ADN) from a primary input (with some clock cycles). In order to *functionally scan-in* test vectors to an ADD circuit, available functional elements are utilized to assign any value within specified range to each register in the circuit from primary inputs while using constant values as side inputs.

Definition 2. *Fso(n)* is satisfied if any value within a specified range can be functionally scanned-out from register node *n* (input to an operation node or ADN) to a primary output (with some clock cycles). In order to *functionally scan-out* test vectors from an ADD circuit, available functional logic is utilized to retrieve values from all registers in the circuit through primary outputs while using constant values as side inputs.

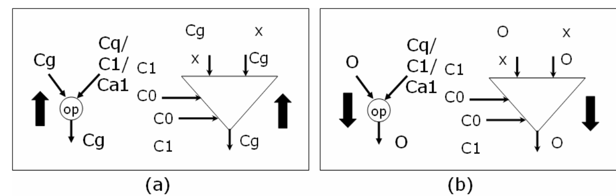


Figure 2. a) General controllability and b) observability of operation nodes and ADNs.

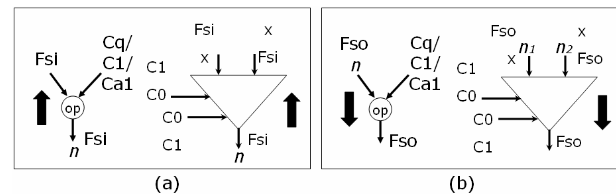


Figure 3. Functional a) scan-in and b) scan-out.

Figure 3 illustrates how available functional logic is exploited for testing. Each node can be *Fsi(n)* and *Fso(n)* by making side inputs *Cq/C0/C1/Ca1*, which is determined according to the type of functional logic. This means that for operation nodes, since we know the constant value of the other input, we can compute for the value of the *Fsi* input such that any arbitrary value within specified range can be passed to the output to make it *Fsi*. In the same way, the value of an *Fso* input to an operation node can be computed in the *Fso* output using the constant value of the other input. For ADNs, any value can be retrieved from the ADN by controlling which *Fsi/Fso* input of the ADN will pass its value to the *Fsi/Fso* output. The ranges are specified by the declaration of registers and PI/PO.

Definition 3. *Functional scan* (abbrev. F-scan) is satisfied when all registers are made *Fsi* and *Fso* to be used for functional scan function. F-scan is a concept that uses available functional elements and paths to create scan chains for testing.

3.3. F-paths and F-scan-paths

The difference between gate-level full scan and F-scan is the method of building scan paths. Gate-level full scan arranges all flip-flops in single or multiple chains for shifting of test vectors while F-scan includes all registers in one or more scan chains called *F-scan-paths*, wherein the least possible scan time is achieved. While full scan augments multiplexers to connect flip-flops, F-scan exploits available functional elements and paths, hence resulting to lesser area overhead. Test vectors are allowed to be functionally scanned-in and -out of the registers along the F-scan-path given that the side inputs to the functional units are constants. F-scan-paths also allow scan-in and -out test vectors simultaneously, thus, similar to full scan, only one test pin is needed to activate scan to handle all registers. Another test pin will be needed to handle the state register, which is further discussed in Section 4.

F-path represents the topology of a path in an ADD circuit from a read node to a write node. This path starts with a read node (PI or register), consists of operation nodes and ADN that the data can pass through along the path, and ends with a write node (PO or register). Side inputs along the path are also included in the path.

Definition 4. An *F-path* is represented by a directed graph $G = (V, A)$ which has the following properties:

1. $V = \{v_1, \ldots, v_p\}$ where v_1 is regarded as the source, which is a read node and v_p is regarded the sink, which is a write node and both cannot be the same register. Each v_i ($i = 2 \sim p-1$) corresponds to an operation node. ADNs are not represented in this graph since this information is not needed for this representation.
2. $\exists k_1, \ldots, k_q$ wherein $q = p-2$, which corresponds to the side input constant for each v_i ($i = 2 \sim p-1$).
3. $(v_m, v_n) \in A$ denotes an arc if there exists a path from one node to another. Every arc $(v_m, v_n) \in A$ has a value range $r(v_m, v_n)$, which consists of a set of values that can be passed through that arc..
4. $\boldsymbol{r(v_1, v_2)}$ and $\boldsymbol{r(v_{p\text{-}1}, v_p)}$ are essential ranges (indicated by the boldface). Essential range $\boldsymbol{r(v_1, v_2)}$ depends on the range assignment of the read node v_1. Essential range $\boldsymbol{r(v_{p\text{-}1}, v_p)}$ is computed from $\boldsymbol{r(v_1, v_2)}$ and the adjustments made by the operations along the path. This range always includes the assigned range to the write node. $\boldsymbol{r(v_{p\text{-}1}, v_p)}$ is obtained before DFT (determination of constant values) so that the constants to be used for operation masks will match the computed essential range at the write node.

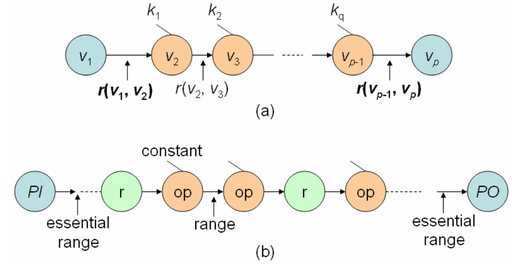


Figure 4. a) F-path and b) F-scan-path.

Definition 5. *A Single F-scan-path* is a sequence of compatible (disjoint) F-paths such that the head of the F-scan-path is a PI and the tail is a PO.

Definition 6. *Multiple F-scan-path* is a set of mutually compatible (disjoint) F-scan-paths.

There may be several possible F-paths available between read and write nodes. However, the F-path candidates to be used in F-scan-paths are only those that can be augmented to match the essential range requirements. Once the F-path candidates are determined, the F-path to be used is chosen such that the lowest area overhead cost possible is achieved. The method of determining F-paths and organizing F-scan-paths is described in the next section.

Definition 7. An ADD circuit is said to be *F-scannable* if every register in the circuit is included in an F-scan-path, wherein it appears once and only once.

4. DFT selection method

We introduce a new functional RTL scan approach called *F-Scan design*, which makes any ADD circuit F-scannable. To do so, we organize F-paths using all of the registers in the ADD circuit. The preliminary concepts and

the DFT algorithm are presented in this section.

4.1. Problem formulation

An ADD circuit is composed of read nodes, write nodes, operation nodes, and ADNs. In order to test it, we control and observe all read and write nodes by organizing them in F-scan-paths. Whenever there is no direct connection from a read node to a write node, the functional logic and path in between can be utilized by augmenting DFT elements that will allow these functional elements to be used for scan. The DFT elements used in F-scan are mask functions and test ADNs. There is an exception to use hold function to handle the state register only.

Definition 7. The DFT for F-scannable ADD circuits is formalized as the following optimization problem.

- **Input:** an ADD circuit
- **Output:** an F-scannable ADD circuit such that there are m F-scan-paths where

$$m = \min\{n_i, n_o\}, \qquad \text{Equation 4.1}$$

and the scan-length is known, which can be solved using m. Given k = # of register nodes, we have:

$$\text{scan-length} = \left\lceil \frac{k}{m} \right\rceil. \qquad \text{Equation 4.2}$$

Shown in Figure 5 is an illustration of the registers of an ADD circuit with more POs than PIs. We choose m, which is the number of F-scan-paths $\{F_1 \ldots F_m\}$, to be the minimum between the number of PIs (excluding reset and clock pins) and POs.

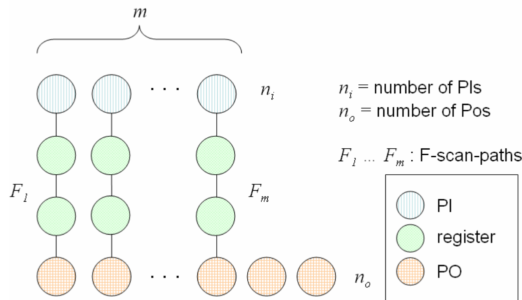


Figure 5. Determining number of F-scan-paths.

- **Optimization:** Minimize *area overhead* (i.e., hardware of augmented DFT elements)

After determining the fixed number of F-scan-paths, we organize the registers to fit the computed scan-length per F-scan-path. Since the ceiling of k/m is considered as the scan-length, there will be cases that the registers cannot completely fill all the F-scan-paths. In this case, one or more F-scan-paths will have less number of registers than the computed scan-length, given that it will result to the least hardware cost and the other F-scan-paths will be longer. Since the scan time is a condition, even if a longer F-scan-path can potentially reduce area overhead further, this is not done.

To assure that the least scan time possible is achieved without adding extra PI and PO, we consider the condition of having m F-scan-paths. We initially compute the number of F-scan-paths to be constructed based on the number of PIs and POs available. However, a *special case* may be considered when the bit widths of the registers in the circuit do not match the bit widths of the available PIs and POs. In this case, we augment a PI or PO or both such that the PIs and POs can handle the register with the most number of bits in the ADD circuit during F-scan.

4.2. Overview of DFT algorithm

This subsection provides an overview of F-scan design. The details are described in the next subsection. The DFT algorithm consists of the following stages.

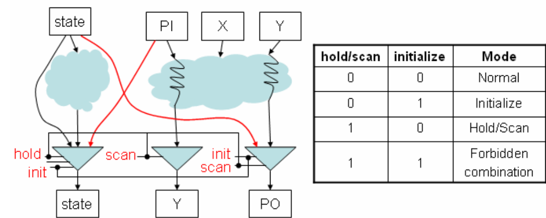
Stage 1. Create a weighted connectivity graph (WCG) based on the information given by the ADD circuit. Here, all possible F-paths between each read/write node are exhaustively determined.

Stage 2. Construct the F-scan-paths to make the circuit F-scannable.

Considering the number of possibilities, determining the F-scan-paths that are disjoint for an ADD circuit, is regarded as an NP-hard problem. Thus, we employ a heuristic algorithm to simplify it.

4.3. DFT algorithm specifics

To further discuss the DFT method, we first explain the special case of handling state registers. Then, we introduce the DFT elements used in F-scan and the cost rules used to estimate the hardware cost due to adding these circuitries for test. We also present the graph representation that aid in determining the best F-path candidate to be included in the F-scan-path.



hold/scan	initialize	Mode
0	0	Normal
0	1	Initialize
1	0	Hold/Scan
1	1	Forbidden combination

Figure 6. Augmentation to handle state registers.

4.2.1. Handling state registers. The state register is not readily accessible from PIs and POs and it usually has a different bit width with the other registers, hence it cannot be included in the F-scan-paths. We augment the circuit in such a way that the state register can be controllable and observable (Figure 6). It is assumed that there is at least one PI and one PO that can handle the bit width of the state register. If not, a PI or PO is added with the bit width of the register having the highest number of bits. Then, after augmentation, we first initialize by scanning-in state value from the PI to the state register. Afterwards, we set the test control inputs in scan mode for the entire circuit, while the state value is being held. When normal mode is done, new state value is scanned out during initialize, followed by the register values. Simultaneously, scan-in can occur while scanning-out.

4.2.2. New ADD elements for masking. Since there is no available ADD node that describes the mask function to keep an input to an operation node constant, we propose the following new ADD elements. These elements are used as DFT elements for F-scan. Figure 7 illustrates the new ADD elements and their corresponding gate-level representation, which are saved in the library.

Definition 10. *C0 mask.* This mask is used for addition and subtraction operation nodes when the side input is not readily a constant. When the scan pin is set to 0, the output of this element is equal to the normal value of the line. If the scan pin is set to 1, the output of this element is 0.

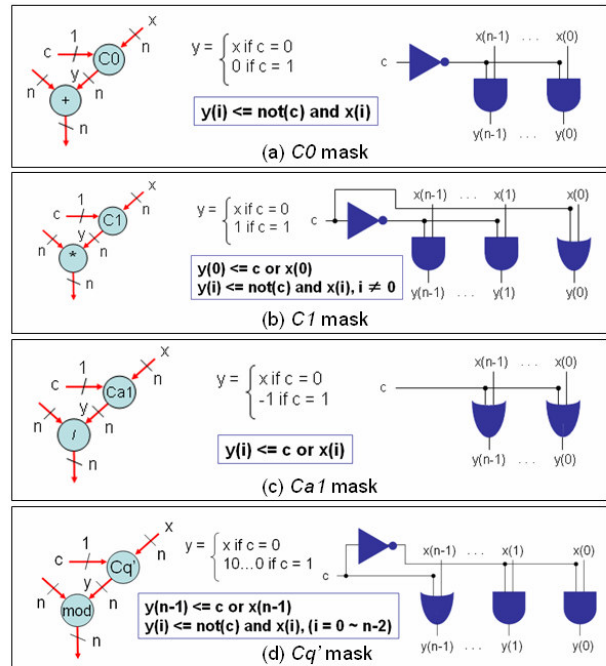
Definition 11. *C1 mask.* This is used for multiplication and division operation nodes for them to pass any value from one input to the output without any changes. The output of this node is equal to the normal value of the line when the scan pin is set to 0. If it is set to 1, the output of this element is 1.

Definition 12. *Ca1 mask.* This is an alternative to *C1* masking applicable to multiplication and division operation nodes as well. When the scan pin is set to 0, normal value of the line applies. If it is set to 1, all bits become 1.

Definition 13. *Cq' masking for modulo.* Since this constant is specific for modulo masking, we indicate it as *Cq'*. Being the highest 2^n value within the range of the line, bitwise, the highest bit is 1 while the rest are zeros. This value (1 0 … 0) is the output of this node if the scan pin is set to 1. If it is set to 0, normal values of the line apply. This type of mask limits the range of a line, which is why using it is subject to the requirements of the essential ranges.

4.2.3. Cost rules. These cost conditions are used in the weighted connectivity graph in order to choose the best F-paths to be used for F-scan.

1. If there is a direct link between registers, the cost is 0. This means that there is no ADN in between registers, just a connecting line.
2. If there is a conditional transfer path denoted by ADN between registers, the cost is equal to 1*g* + 1(# of control signals to ADN-1)*g*, where *g* is the unit for gates.
3. If there is an operational block between two registers, and the other input is not yet a constant value, we use the cost to add mask functions plus the cost to control the ADN. This cost is equal to 1(# operation nodes)(bitwidth of line)*g* + 1*g* + 1(# of control signals to ADN-1)*g*.
4. If there is entirely no connection between two registers, an additional interconnect and ADN will be needed to connect them. Thus, the cost is an estimate of a multiplexer's cost, which is 3(bitwidth of line)*g* + 1*g*.



(a) *C0* mask

(b) *C1* mask

(c) *Ca1* mask

(d) *Cq'* mask

Figure 7. New mask functions for ADD illustrated.

4.2.4. Weighted connectivity graph. The weighted connectivity graph (WGC) represents the topology of an ADD circuit, which includes the read/write nodes and the cost information derived from F-path candidates.

Determining all possible paths from a read node to a write node is a problem that grows exponentially with the circuit size. Thus, F-path candidates for each read-write node pair are limited to at least five possible paths in the circuit. An automatic candidate, of which the cost is similar to full scan, is directly connecting a read node to a

write node through an ADN. Other candidates can be derived from available functional units and paths in the circuit. If incompatibilities exist such that all F-path candidates in a read-write node pair cannot be used, another path is determined and the cost is compared with the full scan cost. The path with the least cost is to be chosen.

Definition 8. A complete representation of an ADD circuit called weighted connectivity graph (WCG) is a directed graph $R = (V, A)$ with the following properties:

1. V is a set of all read and write nodes in the circuit.
2. $(v_i, v_j) \in A$ exists if there is an F-path candidate from the read node corresponding to v_i to the write node corresponding to v_j.
3. Every arc $(v_i, v_j) \in A$ has a cost $c(v_i, v_j)$ computed using the cost rules.

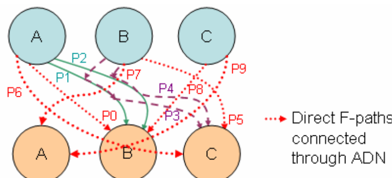


Figure 8. Sample WCG for three read/write nodes.

In Figure 8, it can be seen that multiple candidate F-paths can be derived for each read-write node pair. *P0* … *P9* indicate the cost to realize those F-paths. The graph is comprehensive enough, however, conflicts such as that between paths *P1* and *P3* are not readily shown in this graph. The sample WCG in Figure 8 may show such conflict for discussion here, however, in reality, all paths merely indicate connectivity and weights. The ADD circuit will provide information for compatibility of F-scan-paths to be chosen.

4.2.5. Local optimum heuristic approach. This ensures that in every local location (i.e., read-write node pairs in one scan time frame) the least area overhead due to test possible is achieved by choosing the candidate F-paths that has the least cost.

1. PI/PO Priority. Once the number of F-scan-paths is determined, the primary inputs having F-paths to write nodes (registers) with the least cost are chosen. These F-paths with the least cost are automatically the first in the sequence of F-paths that will create the F-scan-paths. Once chosen, backtrack is not applicable to change these F-paths, therefore they are locked. Similarly, the F-paths with the least cost that connect read nodes (register) to a primary outputs are chosen and locked in the F-scan-paths.
2. Controllability. Starting from the first chosen F-path in each of the F-scan-paths, the next F-path is determined from the F-path candidates such that it is the least cost. The process continues until the registers are arranged in F-scan-paths to guarantee *Fsi* for all registers. If in the process, there is a detected incompatibility, backtrack is done until all F-scan-paths are mutually compatible.
3. Slicing. When the registers are arranged for control, it may occur such that one or more F-scan-paths are longer than the others. Since the length of the F-scan-paths is determined, we slice the long F-scan-paths and move the register or set of registers to shorter F-scan-paths to keep the lengths of all F-scan-paths balanced.
4. Observability. To make all registers *Fso*, we finally connect all F-scan-paths to the F-paths connected to POs. We choose the connection such that it is the cheapest one.

5. Test Generation Procedure

The circuit is tested using the test sequence obtained from the generated test environment and test pattern. The F-scan-paths, however, are tested separately. The testing procedure applied to the ADD circuit and the F-scan-paths are described here.

5.1. Test Generation for the ADD circuit

The test generation procedure involves the following: (a) test environment generation, (b) test pattern generation, and (c) test sequence generation.

The *test environment* consists of the scheduled signal assignment values in order to do F-scan. It involves the F-scan-in phase, test phase, and F-scan-out phase, wherein F-scan-in and -out are overlapped.

Test patterns, on the other hand, are generated through an available automatic test pattern generation (ATPG) tool after synthesizing the circuit to gate-level. In generating these patterns, however, the logic related to F-scan-paths is not included in the synthesized ADD circuit because the F-scan-paths are to be tested separately. This means that the number of test patterns produced for F-scan is the same with the number of test patterns generated for gate-level full scan since the same combinational circuit is used.

The *test sequence* is then derived by embedding the test patterns to the test environment. This includes the input test vectors and the test response. We use the generated test sequence to test the F-scannable ADD circuit. The test sequence is valid if after logic synthesis of the F-scannable ADD circuit, the F-scan-paths are retained. This happens when the test environment variables such as PIs, POs, and registers are preserved after synthesis. Since these variables remain in the gate-level description, there is a

one-to-one correspondence between the F-scannable ADD circuit and its gate-level description, thus the test sequence is valid for fault simulation.

5.1.1. F-scan-in phase. To do F-scan-in in an F-path, all read nodes used to pass values and activate the F-path must contain the computed value (according to equations derived from the F-path information) in order to pass the test pattern obtained during ATPG to the write node. Since the test patterns are yet to be embedded, the F-scan-in environment includes the schedule of signal assignments in terms of equations that completes the F-scan-in phase. Direct value assignments are scheduled wherever applicable, e.g. 1 or 0 for scan/hold pin and initialize pin. Once ATPG is done, the inputs to the PIs involved during test are obtained by generating the test sequence. The input test vectors included in the test sequence consists of the test patterns embedded to the F-scan-in phase and the signals that activate/deactivate F-scan. F-scan-in phase is where all registers are scanned-in with test vectors from the PIs. Hence, the last cycle in this phase is when all registers are *Fsi*.

5.1.2. Test phase. The test phase happens by setting the circuit to normal mode such that all *Fsi* registers are used as input-registers and the same registers (also *Fso*) are used as output-registers for testing the circuit. Here, the test-mode environment includes the PI values, if needed, the output response, and the scan/hold/initialization pins assignment that will turn the circuit to normal mode, i.e. zero value.

5.1.3. F-scan-out phase. To complete the test environment, F-scan-out is done. For a single F-path, the process of F-scan-in simultaneously does F-scan-out. F-scan-in phase and F-scan-out-phase are overlapped, i.e. pipelined, after the first scan-in. Thus, the signals that activate F-scan-in also enable F-scan-out at the same time. This is illustrated in Figure 9(d).

In F-scan-out phase, the values in all *Fso* registers are scanned-out. In order to check all the register values obtained after test phase, the test response is compared with the generated expected response. Since the F-paths are not necessarily I-paths, the test response, which is the pattern of the output of the circuit after test, needs to be adjusted before comparing it with the expected response. The adjustment is made by embedding the test response with the F-scan-out environment. The F-scan-out environment consists of equations derived from the F-scan-path information and the signals that activate F-scan.

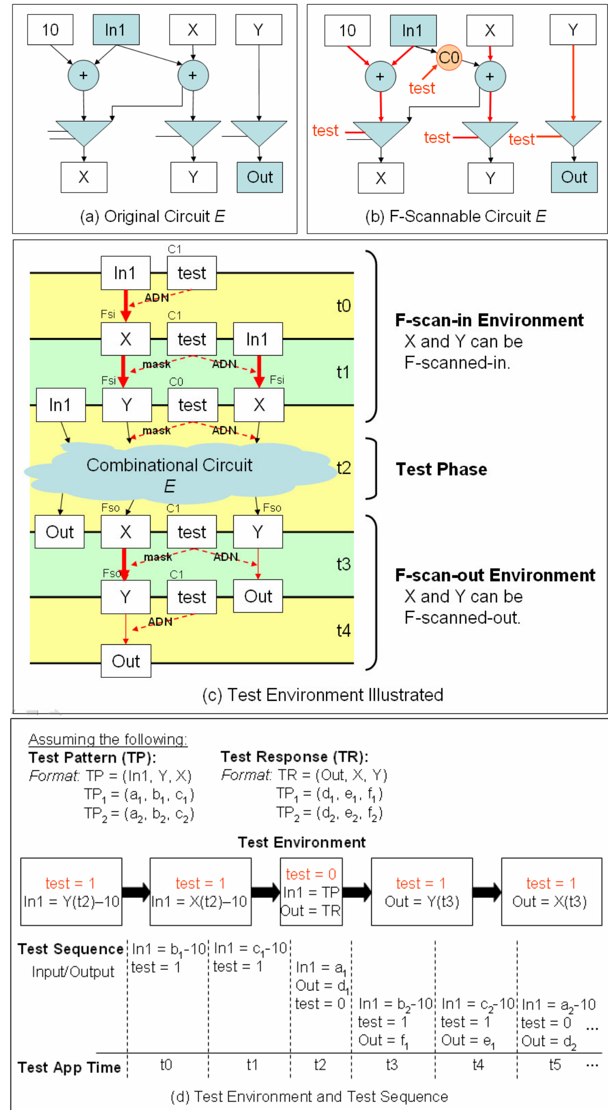


Figure 9. Sample case for test.

5.1.4. An example. A sample case that illustrates how to generate the test environment and the test sequence is shown in Figure 9. The original circuit, *E* (without augmentation), is shown in Figure 9(a). The state register and the control values to the ADNs are not shown, and so these are not included in the example's test environment. Figure 9(b) shows the F-scannable circuit *E*. The F-paths are indicated as red lines and the mask is presented as a C0 element. The test environment is given by Figure 9(c), wherein it shows that a complete scan cycle is equal to five clock cycles, *t0* to *t4*. Figure 9(d) gives the test environment and the resulting test sequence given the test patterns TP_1 and TP_2 and test responses TR_1 and TR_2. Shown in the test sequence, the first F-scan-in occupies *t0*

and *t1*. Test phase happens in *t2*. From *t3* to *t4*, F-scan-in and F-scan-out are overlapped. The same goes on until all the members of the set of test patterns generated by ATPG are embedded with the test environment into the test sequence.

5.2. Test for F-scan-paths

Similar to full scan design, the F-scan-paths are tested separately. This is done by inputting alternating set of all 0's and all 1's (bit width according to the PIs) per clock cycle. By activating the F-scan-paths all the time, the test vectors are F-scanned-in and -out simultaneously, without going into normal mode (test phase). In this test, the test patterns are the 0's and 1's, thus the same procedure in generating test sequence can be done using the same test environment for testing the F-scannable ADD circuit. Since the circuit is not run in normal mode, the expected output response (readjusted based on the F-scan-out environment) should be the same as the input. In case the values all 0's and all 1's are not included in the essential range of a register, these are replaced with values within the essential range.

6. Experimental Results

F-scan is applied to four ITC'99 benchmark circuits namely, B03, B04, B07, and B11. The experimental results for area overhead are given in Table 1. Both F-scan and gate-level full scan are applied to each benchmark circuit and the numbers of extra pins (in bits) and extra gates are given. Comparing the two methods, F-scan proves effective in needing lesser circuitry needed for augmentation to make the circuit testable. B07 is a special case though since the available PI is not enough for the bit width of the internal registers. Thus, an extra 8-bit PI is augmented aside from the usual *hold/scan* and *initialize* pins. Although the pin overhead is greater for F-scan as compared with gate-level full scan whenever there are not enough pins available in the circuit, the augmentation results in better test application time as shown in the next table. Our method is not compared with other functional scan techniques such as Orthogonal Scan [5] and H-Scan [4, 6] through experiments because these techniques are only applicable to the data paths. However, we have already shown the advantages of F-Scan over Orthogonal Scan, in [18].

Table 2 presents the comparison between the test application times of F-scan and gate-level full scan. The test patterns are obtained using TetraMAX of Synopsys. Simultaneous and parallel scanning, with the overlapping of F-scan-in and -out after the first scan in, provide recognizable advantage over the conventional serial scanning of gate-level full scan. Moreover, we can observe from B04 results that as the circuit becomes larger, the advantage of F-scan to make the test application time shorter also increases. This establishes the superiority of our proposed method, since both area overhead and test application time are optimally minimized.

We expect the fault coverage and test generation time of F-scan to be comparable with gate-level full scan, which we will prove through further experiments. Moreover, in order to achieve this, a constrained ATPG methodology is necessary wherein the constraint depends on the essential range of each register.

7. Conclusion and Future Work

A novel approach to functional RTL scan called F-scan has been proposed. It maximally utilizes available functional elements and paths in the circuit to insert scan paths for testing. It minimizes area overhead due to test compared to full scan design, as shown by the experimental results. Test application time is also superior against full scan. Since ADD representation is used, faster test generation time is also expected.

As future work, we are going to conduct experiments for all benchmarks of ITC'99. We expect the same or even better results for large benchmarks compared to the experiments for small benchmarks presented in this paper. That is, we expect F-scan to cause area overhead due to logic for test to be at the minimum and the test application time to be more than five times shorter than gate-level full scan.

Acknowledgment

This work was supported in part by Japan Society for Promotion of Science (JSPS) under Grants-in-Aid for Scientific Research (B) (No. 20300018). The authors thank Prof. Michiko Inoue, Prof. Satoshi Ohtake, and Prof. Tomokazu Yoneda for their valuable comments and suggestions during discussions regarding our work.

References

[1] H. Fujiwara, *Logic Testing and Design for Testability*, Cambridge, MA: MIT Press, 1985.

[2] C-C. Lin, M. M. Sadowska, M. T-C. Lee, K-C. Chen, "Cost-Free Scan: A Low-Overhead Scan Path Design," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 17, No. 9, pp. 852-861, 1998.

[3] R. Gupta and M. A. Breuer, "Partial scan design of register transfer level circuits,"*JETTA*, Vol. 7, pp. 25-46, 1995.

[4] S. Bhattacharya and S. Dey, "H-Scan: A High Level Alternative to Full-Scan Testing with Reduced Area and Test Application Overheads," *Proc. VLSI Test Symposium*, pp. 74-80, 1996.

[5] R. B. Norwood and E. J. McCluskey, "Orthogonal scan: Low overhead scan for data paths," *Proc. Int. Test Conf.*, pp. 659–668, 1996.

[6] T. Asaka, S. Bhattacharya, S. Dey and M. Yoshida, "H-Scan+: A Practical Low-Overhead RTL Design-for-Testability Technique for Industrial Designs," *Proc. International Test Conference*, pp. 265-274, 1997.

[7] Y. Huang, C. C.Tsai, N. Mukhejee, O. Samman, D. Devries, W. T. Cheng, and S. M. Reddy, "Synthesis of scan chains for netlist descriptions at RT-level," *JETTA*, Vol. 18, pp. 189-201, 2002.

[8] C. Y. Ooi and H. Fujiwara, "A new scan design technique based on pre-synthesis thru functions," *Proc. 15th IEEE Asian Test Symposium*, pp. 163-168, 2006.

[9] I. Ghosh, A. Raghunathan, N. K. Jha, "Design for Hierarchical Testability of RTL Circuits Obtained by Behavioral Synthesis," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 16, No. 9, pp. 1001-1014, 1997.

[10]K. Takabatake, M. Inoue, T. Masuzawa, and H. Fujiwara, "Non-scan design for testable data paths using thru operation," *Proc. Asia and South Pacific Design Automation Conf. 1997*, pp. 313–318 (1997).

[11] I. Ghosh, A. Raghunathan, N. K. Jha, "A Design-for-Testability Technique for Register-Transfer Level Circuits Using Control/Data Flow Extraction," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 17, No. 8, pp. 706-723, 1998.

[12] H.Wada, T. Masuzawa, K. K. Saluja, and H. Fujiwara, "Design for strong testability of RTL data paths to provide complete fault efficiency," *Proc. 13th Int. Conf. VLSI Des.*, pp. 300–305, 2000.

[13] S. Ohtake, H. Wada, T. Masuzawa, and H. Fujiwara, "A non-scan DFT method at register-transfer level to achieve complete fault efficiency," *Proc. Asia South Pacific Des. Autom. Conf.*, pp. 599–604, 2000.

[14]S. Ohtake, S. Nagai, H. Wada and H. Fujiwara, "A DFT method for RTL circuits to achieve complete fault efficiency based on fixed-control testability," *Asia and South Pacific Design Automation Conference 2001*, pp. 331-334, Feb. 2001.

[15] H. Fujiwara, H. Iwata, T. Yoneda, and C. Y. Ooi, "A Non-Scan Design for-Testability for Register-Transfer Level Circuits to Guarantee Linear-Depth Time Expansion Models," *IEEE Trans. on Computer-Aided Des. of Integrated Circuits and Systems*, Vol. 27, No. 9, pp. 1535-1544, 2008.

[16]V. Chaiyakul, D. D. Gajski, and L. Ramachandran, "High-level transformations for minimizing syntatic variances," *Proc. Design Automation Conf.*, pp. 413-418, 1993.

[17]I. Ghosh and M. Fujita, "Automatic Test Pattern Generation for Functional Register-Transfer Level Circuits Using Assignment Decision Diagrams," *IEEE Trans. Computer-Aided Design on Integrated Circuits and Systems*, Vol. 20, No. 3, pp. 402-415, 2001.

[18] Marie Engelene J. Obien and Hideo Fujiwara, "F-Scan: An Approach to Functional RTL Scan for Assignment Decision Diagrams," *IEEE 8th International Conference on ASIC (ASICON2009)*, 2009.

Table 1. Overhead Results for Gate-Level Full Scan and F-Scan

ITC'99 Benchmarks	# of PI / # of PO (Pins)	Total Number of Gates Before Scan Insertion	# of Extra Pins		# of Extra Gates / Overhead (%)	
			Gate-Level Full Scan	F-Scan	Gate-Level Full Scan	F-Scan
B03	4 / 4	149	1	2	93 / 62.41	44 / 29.53
B04	11 / 8	597	1	2	201 / 33.67	100 / 16.75
B07	1 / 8	420	1	8-bit + 2	132 / 31.43	70 / 16.67
B11	7 / 6	481	1	2	96 / 19.96	71 / 14.76

Table 2. Test Length Results for Gate-Level Full Scan and F-Scan

ITC'99 Benchmarks	# of Flip-Flops	# of Test Patterns (TP)	Total Test Length (equation / cycles)	
			Gate-Level Full Scan	F-Scan
B03	30	37	TP(30 + 1) + 30 / 1177	TP(6 + 1) + 6 / 265
B04	66	132	TP(66 + 1) + 66 / 8910	TP(10 + 1) + 10 / 1462
B07	49	71	TP(49 + 1) + 49 / 3599	TP(7 + 1) + 7 / 575
B11	31	125	TP(31 + 1) + 31 / 4031	TP(5 + 1) + 5 / 755