A Nonscan Design-for-Testability Method for Register-Transfer-Level Circuits to Guarantee Linear-Depth Time Expansion Models

Hideo Fujiwara, *Fellow, IEEE*, Hiroyuki Iwata, *Student Member, IEEE*, Tomokazu Yoneda, *Senior Member, IEEE*, and Chia Yee Ooi, *Member, IEEE*

***Abstract*—This paper presents a nonscan design-for-testability (DFT) method for register-transfer-level (RTL) circuits. We first introduce the τ^k *notation* to analyze the test generation complexity, as well as two classes of sequential circuits, namely: 1) the *combinationally testable* class and 2) the *acyclically testable* class. Then, we introduce a new class of *linear-depth time-bounded* circuits as one of the acyclically testable classes. The linear-depth time-bounded testability guarantees that the number of time frames required for any testable fault is bounded by a linear function of the number of flip-flops in the circuit during the test generation process. As one of the linear-depth time-bounded classes, we introduce a new class of RTL circuits, called the *cycle-unrollable* RTL circuits, which is shown to be linear depth time bounded. We propose a DFT method to make RTL circuits cycle unrollable and a test generation method for cycle-unrollable RTL circuits. Experimental results show that we can drastically reduce hardware overhead and test application time compared to the full-scan method and the method proposed by Ohtake *et al.* Moreover, our proposed method can achieve 100% fault efficiency for gate-level single stuck-at faults in practical test generation time and allow at-speed testing.**

***Index Terms*—Acyclic testability, at-speed testing, design for testability, register transfer level (RTL), test generation complexity.**

I. Introduction

With the progress of semiconductor technology, testing of very-large-scale integration becomes more difficult, and the cost has been increasing. Therefore, it is important to achieve high fault efficiency (FE) with a low cost. Test generation, even for combinational circuits, is well known to be NP-complete in general [1], although some classes of combinational circuits whose test generation complexity is of polynomial time were introduced in [25] and [26]. However, empirical observation shows that for combinational circuits, test patterns with 100% FE for single stuck-at faults can be obtained even for large circuits by automatic test pattern generators (ATPGs), and hence, the test generation complexity of practically encountered combinational circuits seems to be polynomial [2], [3]. Based on this observation, several classes of sequential circuits whose test generation complexity is equivalent to *combinational test generation complexity* have been introduced [4]–[7]. Such a class is referred to as a *combinationally testable class*. In our previous work [8], [9], we introduced the τ^k *notation* to express the test generation complexity of a given circuit class relative to the combinational test generation complexity denoted as $\tau(n) = \Theta(n^r)$, where n is the number of gates of the combinational circuit, and r is some constant larger than 2. The test generation complexity of combinationally testable sequential circuits is called τ*-equivalent* in terms of the τ^k notation. For acyclic sequential circuits or cycle-free sequential circuits, it is also known that test patterns with 100% FE for gate-level single stuck-at faults can be obtained in practical test generation time [10], [27], [28]. Test generation for those acyclic sequential circuits can be performed with a modified combinational test generation algorithm [10], [27], [28]. Hence, it seems that the test generation complexity for acyclic sequential circuits is τ*-equivalent*. However, we showed in [8] and [9] that the class of acyclic sequential circuits is not τ*-equivalent* but τ^2*-bounded*, which means that the test generation complexity of acyclic sequential circuits is bounded by the square of the combinational test generation complexity, i.e., $O(\tau^2(n))$. We regard acyclic sequential circuits as easily testable. A class of cyclic sequential circuits whose test generation complexity is equivalent to that of acyclic sequential circuits is referred to as an *acyclically testable class*. We introduced in [11] a class of cyclic sequential circuits whose test complexity is τ^2-bounded.

The test generation problem for general sequential circuits, which is modeled by an iterative logic array called the *time expansion model* (TEM), has greater time complexity than that for acyclic sequential circuits. A design-for-testability (DFT) approach is needed to ease the test generation for general sequential circuits. If any testable fault can be test-generated by using a TEM whose depth is $O(m)$, where m is the number of flip-flops of the circuit, the test generation complexity is τ^2-bounded, i.e., acyclically testable. In this paper, we call

Manuscript received September 25, 2007; revised April 22, 2008. Published August 20, 2008 (projected). This work was supported in part by the Japan Society for the Promotion of Science under Grant-in-Aid for Scientific Research B (15300018) and in part by the 21st Century Center of Excellence Program. An earlier version of this paper was presented at the 44th Design Automation Conference, San Diego, CA, June 2007. This paper was recommended by Associate Editor A. Ivanov.

H. Fujiwara and T. Yoneda are with the Nara Institute of Science and Technology, Kansai Science City 630-0192, Japan (e-mail: fujiwara@is.naist.jp; yoneda@is.naist.jp).

H. Iwata was with the Graduate School of Information Science, Nara Institute of Science and Technology, Kansai Science City 630-0192, Japan. He is now with Renesas Technology Corporation, Tokyo 187-8588, Japan.

C. Y. Ooi was with the Graduate School of Information Science, Nara Institute of Science and Technology, Kansai Science City 630-0192, Japan. She is now with the University of Technology Malaysia, Skudai 81310, Malaysia.

Digital Object Identifier 10.1109/TCAD.2008.927757

such a TEM a *linear-depth TEM* and introduce a new DFT method for register-transfer-level (RTL) circuits that guarantees the existence of a linear-depth TEM for any testable fault. The augmented circuit becomes acyclically testable.

The most widely used DFT technique for sequential circuits is the full-scan approach, which makes a given sequential circuit combinationally testable so that 100% FE for gate-level single stuck-at faults can be achieved. However, it requires long test application time because of scan-shift operations. Moreover, it requires a large hardware overhead and cannot allow at-speed testing. To avoid these disadvantages, nonscan DFT methods at higher abstraction levels have been proposed in [12]–[20] and [24]. In [18], a nonscan DFT method, which guarantees 100% FE for RTL controller–data path circuits, has been proposed, where a single stuck-at fault model at the gate level is considered.

In [18], the method of [19] is applied to the controller, and the method of [20] is applied to the data path. The method in [20] is based on hierarchical test generation [15], and a new testability, called the *strong testability*, was introduced as a characteristic of data paths to guarantee the existence of test plans (sequences of control signals) for each hardware element in the data paths, where the strong testability is based on the concept of *I-path* introduced in [24]. However, the DFT methods in [19] and [20] assumed that controllers and data paths are isolated from each other, and that the signal lines in between them are directly controllable and observable from outside of the circuits. Therefore, extra multiplexers (MUXs) are added to the signal lines in between the controller and the data path, and an extra test controller (TC) is also embedded to provide the test plans for the data path. The method in [18] can allow at-speed testing and achieve much shorter test application time compared to the full-scan approach. However, hardware and delay overheads are larger compared to the full-scan approach because of the extra MUXs and the TC.

In this paper, we propose a DFT method for RTL controller–data path circuits to reduce the hardware overhead while achieving 100% FE for gate-level single stuck-at faults, at-speed testing, and shorter test application time compared to the full-scan approach. We first introduce a new testability called the *linear-depth time-bounded testability* for RTL circuits that guarantees the existence of a linear-depth time expansion for any testable fault. Then, we introduce a new class of RTL circuits, called the *cycle-unrollable* RTL circuits, which is shown to be linear depth time bounded.

A DFT method and a test generation method based on cycle unrollability are also proposed. The cycle unrollability is a characteristic of the whole RTL controller–data path circuit, and we do not need to explicitly isolate the controller and the data path from each other during DFT and test generation. Moreover, even if we focus on the data path part, a data path with a strong testability is a subclass of the data path with cycle unrollability. Therefore, the hardware overhead of the proposed method can be expected to be lower than that in [18]. Experimental results show the effectiveness of the proposed method compared to the full-scan approach and the method in [18].

The rest of this paper is organized as follows. In Section II, we define the τ^k notation to analyze the test generation complexity, and the classes of sequential circuits called the *combinationally testable* class and the *acyclically testable* class. We also introduce a new class of sequential circuits, called the *linear-depth time-bounded* class, which is shown to be acyclically testable, and hence, the test generation complexity is τ^2-*bounded*. In Section III, we introduce a new class of RTL circuits, called the *cycle-unrollable* RTL circuits, which is shown to be linear depth time bounded and, hence, τ^2-bounded. In Section IV, we present the overview of the proposed DFT method to augment an arbitrary RTL circuit into a cycle-unrollable RTL circuit. In Section V, we present the DFT method in detail. In Section VI, we present a test generation method for cycle-unrollable RTL circuits. Experimental results are presented in Section VII, and the discussion is concluded in Section VIII.

II. Test Generation Complexity

A. τ^k Notation

Generally, the asymptotic notation is used to describe the asymptotic running time of an algorithm. This notation is also convenient for describing the worst-case running time of the test generation problem. Let $g(n)$ be a given function. The following briefly describes $\Theta(g(n))$, $O(g(n))$, and $\Omega(g(n))$. A function $f(n)$ belongs to the set $\Theta(g(n))$ if $g(n)$ is an asymptotically tight bound for $f(n)$. A function $f(n)$ belongs to the set $O(g(n))$ if $g(n)$ is an asymptotically upper bound for $f(n)$, whereas a function $f(n)$ belongs to the set $\Omega(g(n))$ if $g(n)$ is an asymptotically lower bound for $f(n)$ [21].

To facilitate our discussion, we define the time complexity of test generation as follows.

Definition 1: The time complexity of a problem P is the time complexity of the fastest algorithm for the problem P. Let $T_C(n)$, $T_S(n)$, and $T_\alpha(n)$ be the time complexity of the test generation problem for combinational, sequential, and circuits in class α, respectively.

To further clarify the test generation complexity, we define the τ^k *notation*. We consider $T_C(n)$ as a basic unit of the time complexity of the test generation problem; therefore, $\tau(n)$ is used to denote $T_C(n)$ in the following text, where $\tau(n) = T_C(n) = \Theta(n^r)$ for some constant $r \geq 2$.

Definition 2: $T(n)$ is τ^k-*equivalent* if and only if $T(n) = \Theta(\tau^k(n))$ and is τ^k-*bounded* if and only if $T(n) = O(\tau^k(n))$, where $k > 0$.

Definition 3: Class α is τ^k-*equivalent* if and only if $T_\alpha(n) = \Theta(\tau^k(n))$ and is τ^k-*bounded* if and only if $T_\alpha(n) = O(\tau^k(n))$, where $k > 0$.

B. Combinationally Testable Class

It has been shown in previous works that a strongly balanced sequential circuit [5], a balanced sequential circuit [4], and an internally balanced sequential circuit [6] can be transformed into their combinational equivalents, respectively, whose test patterns can be transformed back to the test sequences of the original sequential circuit. Hence, we have the following theorem.

Theorem 1 [8], [9]: Internally balanced sequential circuits, balanced sequential circuits, and strongly balanced sequential circuits are τ-equivalent.

Definition 4: A class of sequential circuits α is *combinationally testable* if the test generation complexity for α is equivalent to that for combinational circuits, i.e., α is τ-equivalent.

C. Acyclically Testable Class

An acyclic sequential circuit is a sequential circuit without feedback, i.e., *acyclic* in structure. It has been proven in previous works that the test generation complexity for this class is not τ-equivalent if the test generation model is a TEM.

Theorem 2 [8], [9]: There exists an acyclic sequential circuit whose test generation complexity represented by TEM is not τ-equivalent.

Theorem 3 [8], [9]: Acyclic sequential circuit is τ^2-bounded.

Let us define a class of sequential circuits whose test generation complexity is equivalent to acyclic sequential circuits as follows.

Definition 5: A class of sequential circuits α is said to be *acyclically testable* if the test generation complexity for α is equivalent to that of acyclic sequential circuits.

Note that an acyclically testable circuit is not necessary acyclic, i.e., there exists a cyclic sequential circuit that is acyclically testable. From Theorems 2 and 3, the acyclically testable class is not τ-equivalent but τ^2-bounded.

Next, let us introduce a new class of sequential circuits as one of the acyclically testable classes.

Definition 6: A TEM is referred to as a *linear-depth TEM* if the number of time frames is bounded by a linear function of the number of flip-flops of the circuit. The *depth* of a TEM is the number of time frames of the TEM.

Definition 7: A sequential circuit S is said to be *linear depth time bounded* or a circuit with *linear-depth TEM* if any testable fault in S can be test-generated by using a TEM whose depth is $O(m)$, where m is the number of flip-flops in S.

Note that, in general, the depth of a TEM that is necessary for test generation increases to $O(9^m)$ in the worst case when the nine-valued algebra is used, where m is the number of flip-flops of the circuit [30]. Hence, a class of linear-depth time-bounded sequential circuits is a special subclass of general sequential circuits.

We have the following theorem for linear-depth time-bounded sequential circuits.

Theorem 4: The test generation complexity of a linear-depth time-bounded sequential circuit is τ^2-bounded.

Proof: Let S be a linear-depth time-bounded sequential circuit. Let m and n be the number of flip-flops and the number of gates in S, respectively. Then, any testable fault in S can be test-generated by using a TEM whose depth is $O(m)$ and, hence, $O(n)$. Since the size of each time frame of the TEM is $O(n)$, the total size of the TEM becomes $O(n^2)$. Hence, the test generation complexity for the TEM is $O(\tau(n^2)) = O(\tau^2(n))$. Therefore, the test generation complexity for S is τ^2-bounded. ■

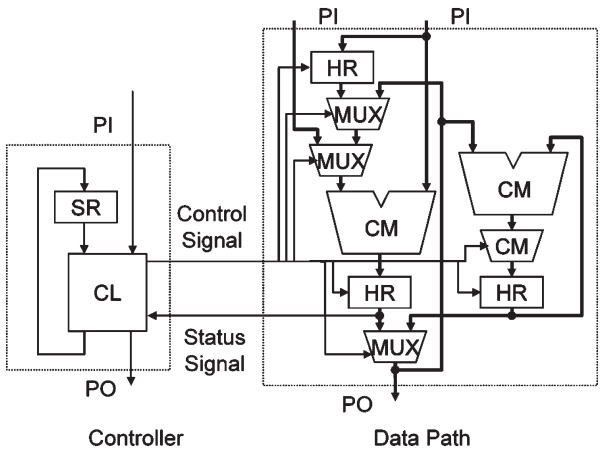


Fig. 1. RTL controller–data path circuit.

III. Acyclically Testable RTL Circuits

A. RTL Circuits

In the RTL description, a circuit generally consists of a controller and a data path. An example of an RTL circuit is shown in Fig. 1. A controller consists of a primary input (PI), at most one primary output (PO), a state register (SR), and a combinational logic circuit (CL). A data path consists of PIs, POs, hold registers (HRs), load registers (LRs), MUXs, and combinational operational modules (CMs). Note that PIs and POs are multiple bits wide. The SR, CL, PIs, POs, HRs, LRs, MUXs, and CMs are called hardware elements. Each hardware element has at most two data inputs, one control input, one status output, and one data output. An SR has one data input from a CL and a data output to the same CL. The CL has two data inputs (one from the SR and one from the PI), at most two data outputs (one to the SR and at most one to the PO), control outputs to the data path, and status inputs from the data path.

Hardware elements are connected by signal lines. The signal lines are classified into data signal lines, control signal lines, and status signal lines. A data signal line connects a data output to a data input. A control signal line connects a control output of a CL to a control input of a hardware element in a data path. A status signal line connects a status output of a hardware element in a data path to a status input of a CL.

An RTL circuit is represented by an RTL graph whose vertices are the inputs and the outputs of hardware elements, and whose arcs are the signal lines and the data flows between the inputs and the outputs of hardware elements. Let $p = (e_1, l_1, e_2, \ldots, e_{k-1}, l_{k-1}, e_k)$ be a path starting from e_1 and ending at e_k, where e_i and l_i denote a vertex and an arc, respectively. The number of registers on p is called the *sequential depth* of p. A path p with no repeated vertices is called a *simple path*. A path p is called a *cycle* if the source vertex e_1 and the sink vertex e_k are the same vertex, and the remaining path from e_2 to e_{k-1} is a simple path. Different simple paths from e_i to e_j are called *reconvergent paths*. If the sink vertex of a path $p1$ and the source vertex of a path $p2$ are identical, then we denote the concatenation of the paths as $(p1, p2)$. For each hardware element on p, the inputs and the

outputs of the hardware element are called *on-inputs and on-outputs*, respectively, if they are on p. Similarly, the inputs and the outputs of the hardware element are called *off-inputs and off-outputs*, respectively, if they are not on p.

B. Strong Testability

In [20], a new testability for RTL data paths, called the strong testability, was introduced as follows.

Definition 8: A data path is said to be *strongly testable* if there exists a test plan for each hardware element m that makes it possible to apply any pattern to m from PIs and to observe any response of m at POs.

The strong testability is based on the hierarchical test, and the testing is performed as follows: 1) the test patterns are generated for each combinational hardware element m by using a combinational ATPG tool and 2) the test patterns are applied from PIs to m, and the responses are observed at a PO by using the test plan for m. To guarantee the existence of the test plan for each hardware element, the DFT method proposed in [20] added a thru function to every CM and a hold function to some LRs. The thru function of a CM provides a functionality to propagate values from one of its data inputs to its data output without any change independently of its other data input. Test generation time for a strong testable data path is short since a combinational ATPG tool is used for each hardware element. However, it may not be necessary to have a test plan for every hardware element to achieve 100% FE for gate-level single stuck-at faults. Moreover, since the DFT methods in [20] assumed that control/status signals from/to a controller are directly controllable/observable from outside the circuits, extra MUXs and embedded TCs are required to provide the test plans when we consider the whole RTL controller–data path circuit.

C. Cycle Unrollability

It is known that ATPGs for combinational circuits can achieve 100% FE for gate-level single stuck-at faults in practical test generation time. For acyclic sequential circuits, it is also known that test patterns with 100% FE can be generated in practical test generation time on the TEM [10]. Therefore, we expect that for an RTL circuit, we can achieve 100% FE for gate-level single stuck-at faults in practical test generation time if there exists a TEM such that the number of time frames required for any testable fault is bounded by a linear function of the number of registers in it. To guarantee the existence of such a TEM, only a part of the hardware elements require the test plans. To satisfy this condition, we introduce a new concept of *cycle unrollability* for RTL circuits. The cycle unrollability is a characteristic of the whole RTL controller–data path circuit, and we do not need explicit isolation for the controller and the data path during DFT and test generation. Moreover, even if we focus on the data path part, a data path with a strong testability is a subclass of the data path with cycle unrollability. Therefore, the hardware overhead of the proposed method can be expected to be lower than that in [18].

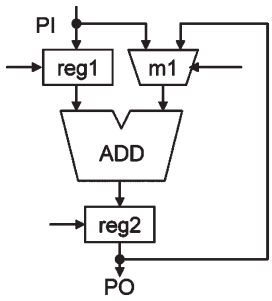


Fig. 2. Example RTL circuit with the cycle-unrolling path.

Definition 9: A set of values that can appear at a signal line l in a normal operation is said to be the range of l. The range of the input (output) values of the hardware element connected with l is defined as *the range of line* l.

Definition 10: Let R_{li} and R_{lj} be the range of signal lines l_i and l_j, respectively. Dependence exists between l_i at time t and l_j at time t' if l_i cannot be set to any value in R_{li} at t when l_j is set to any value in R_{lj} at t'. The dependence between an input (output) connected with l_i and an input (output) connected with l_j is defined as the dependence between l_i and l_j.

Definition 11: Let CD be an RTL circuit. A path p_c is said to be a *cycle-unrolling path* if p_c satisfies the following conditions for a cycle c in CD.

1) Let $p_{\rm cin}$ be a simple path from a PI to an on-output of a hardware element $m_{\rm cin}$ on c. Let $p_{\rm cout}$ be a simple path from an on-output of a hardware element $m_{\rm cout}$ on c to a PO. Let p_{c1} be c starting and ending at the on-output of $m_{\rm cin}$. Let p_{c2} be a simple path from the on-output of $m_{\rm cin}$ to the on-output of $m_{\rm cout}$ along c. Then, $p_c = (p_{\rm cin}, p_{c1}, p_{c2}, p_{\rm cout})$. Note that p_{c1} and p_{c2} are overlapped.
2) For any hardware element m_i on $p_{\rm cin}$, the on-output of m_i can be set to any value.
3) For any hardware element m_i on $(p_{c1}, p_{c2}, p_{\rm cout})$, there exists a value in the range of the off-input of m_i such that the value can propagate any change at the on-input of m_i to the on-output of m_i.
4) For any hardware element m_i on p_c, let d_i be the sequential depth of the path from the PI on p_c to the on-input of m_i along p_c. There exists no dependence between the PI at time t and the off-input of m_i at time $t + d_i$.
5) For any two hardware elements m_i and m_j on p_c, let d_i and d_j be the sequential depth of the paths from the PI on p_c to the on-input of m_i and m_j along p_c, respectively. If d_i is not equal to d_j, there exists no dependence between any off-input of m_i at time $t + d_i$ and any off-input of m_j at time $t + d_j$.

Example 1: Fig. 2 shows an RTL circuit that includes a cycle and its cycle-unrolling path. Suppose that all the control signals for registers and a MUX are controllable. Let c be the cycle, $p_{\rm cin} = ({\rm PI}, m1)$, $p_{c1} = (m1, {\rm ADD}, {\rm reg2}, m1)$, $p_{c2} = (m1, {\rm ADD}, {\rm reg2})$, and $p_{\rm cout} = ({\rm reg2}, {\rm PO})$. Then, $p_c = (p_{\rm cin}, p_{c1}, p_{c2}, p_{\rm cout})$ is a cycle-unrolling path for c. Since $m1$ is a MUX, the output of $m1$ on $p_{\rm cin}$ can be set to any value [condition 1)]. For $m1$, ADD, and reg2 on $(p_{c1}, p_{c2}, p_{\rm cout})$, they can propagate any change at the on-input

of each module to the on-output by applying an on-input select signal, all 0s, and a load signal at the off-input, respectively [condition 3)]. Regarding condition 4), ADD shares PI with the cycle-unrolling path. For ADD on p_{c1}, since the sequential depth from PI along p_c is 0, and the sequential depth from PI to the off-input of ADD is 1, there exists no dependence. For ADD on p_{c2}, the sequential depth from PI along p_c is 1, and the sequential depth from PI to the off-input of ADD is also 1. However, register reg1 has a hold function in this example. Hence, condition 4) is satisfied. Regarding condition 5), all the control signals are assumed to be controllable, and there exists no dependence between any pair of off-inputs.

Here, we define a new class of RTL circuits as follows.

Definition 12: An RTL circuit CD is said to be *cycle unrollable* if there exists a cycle-unrolling path p_c for each cycle c in CD.

Theorem 5: Let CD be a cycle-unrollable RTL circuit. Then, there exists a TEM such that the number of time frames is at most $d_{\max} + n_{\mathrm{REG}}$, and a test sequence can be generated on the TEM for any testable fault in CD, where $d_{\max}$ is the maximum sequential depth of all cycle-unrolling paths in CD.

Proof: Let m denote any hardware element on p_c. According to conditions 4) and 5) in Definition 11, the on-input of m (not m_{cin}) can be set to any arbitrary value in the range of the on-input. Moreover, according to condition 2), all the inputs (outputs) along p_{cin} can be assigned any arbitrary value. Therefore, the on-output of m_{cin} can be justified with any arbitrary value as well. Similarly, the on-inputs along $(p_{c1}, p_{c2}, p_{\mathrm{cout}})$ can be set to any arbitrary value in the range. For m_{cin}, any value in the range of the input on $(p_{c1}, p_{c2}, p_{\mathrm{cout}})$ of m_{cin} can be justified through $(p_{c1}, p_{c2}, p_{\mathrm{cout}})$. Therefore, all the inputs for m on p_c can be set to any value in the range. Accordingly, if there exists a test pattern for a testable fault in m, the fault will be activated. Similarly, according to conditions 2)–5), since the fault effect in m can propagate to its on-output, the fault effect can reach a PO.

Next, let us consider the case where the hardware element is not on any cycle-unrolling path. Let m' denote the hardware element not on any cycle-unrolling path. Since there is no cycle from PI to m', the structure from PI to m' is acyclic. Thus, any value in the range of the input of m' can be justified. If there exists a test pattern for a testable fault in m', the fault can be activated. Let P_{mo} denote a set of all the simple paths from m' to any PO. If a path p in P_{mo} does not intersect with any cycle-unrolling path, this path p is acyclic. If a path p in P_{mo} has a subpath that is a cycle-unrolling path, the path from m' to the subpath is acyclic, and the subpath satisfies the conditions of Definition 11, the error can propagate to the PO along p. Thus, if there exists a test pattern for the testable fault, the error can propagate to a PO.

Let p_m denote a simple path from PI to PO. The sequential depth for p_m is at most $d_{\max}$. Let m_i and m_j denote two hardware elements on p_m, and let d_i and d_j denote the sequential depths of m_i and m_j, respectively, along p_m. Then, at most n_{REG} time frames are needed to justify any values in the range to each off-input of m_i at $t + d_i$ and m_j at $t + d_j$ without any dependence. The number of time frames becomes maximum when the sequential depth of p_m is $d_{\max}$, and the justification of any value in the range to the off-input of the hardware element whose sequential depth from PI is zero needs at most n_{REG} time frames. Therefore, the number of time frames is at most $d_{\max} + n_{\mathrm{REG}}$ for fault activation and propagation. ■

From Theorem 5, we can see that a cycle-unrollable RTL circuit is linear depth time bounded, and hence, we have the following theorem.

Theorem 6: Any cycle-unrollable RTL circuit is linear depth time bounded, and the test generation complexity is τ^2-bounded.

IV. Overview of the Proposed Method

First, we apply the DFT to make a given RTL circuit cycle unrollable. In Section V, the DFT problem that we consider in this paper is formally presented, and the DFT method is explained in detail. After that, we generate a TEM such that the number of time frames required for any testable fault is bounded by a linear function of the number of registers in the circuit. Then, a combinational ATPG is applied to the TEM. Test generation for the TEM requires a combinational ATPG, which can deal with multiple stuck-at faults. We use the circuit model, which can express multiple stuck-at faults in a TEM as a single stuck-at fault [22], [29] in our experiments since TestGen (Synopsys) cannot deal with multiple stuck-at faults. Finally, we transform the test patterns for the TEM into test sequences for the RTL circuit.

The proposed DFT method consists of the following four steps: 1) construct a control forest; 2) construct an observation forest; 3) resolve the dependence; and 4) generate a TC. In step 1), we decide a control path for each data input of each hardware element. The control path is used for propagating any value in the range of the data input. To guarantee the propagation through the control paths, we add thru functions if necessary. Similarly, in step 2), we decide an observation path for each data output of each hardware element. In step 3), if dependence exists between inputs of a hardware element by using the paths decided in steps 1) and 2), we resolve it by using the hold functions of HRs. We add a hold function to LRs if necessary. In step 4), a TC is generated to activate the control/observation paths and hold functions.

Here, we summarize the advantages of the proposed method compared to that in [18]. In [18], the authors first completely separated the circuit into the data path and the controller by adding extra MUXs to the control/status signals between them. Then, they applied the different DFT methods and independently generated test patterns for the data path and the controller. The test patterns for the control/status signals between the data path and the controller are provided from the additional TC, which is a sequential circuit. On the other hand, the proposed method in this paper does not need explicit separation for the data path and the controller. We add a simple TC that is a combinational circuit to activate the control/observation paths. Then, we generate test patterns for the whole data path–controller circuit. This is because the cycle unrollability is a characteristic of the whole RTL controller–data path circuit. Therefore, the proposed method can be expected to be lower than that in [18].

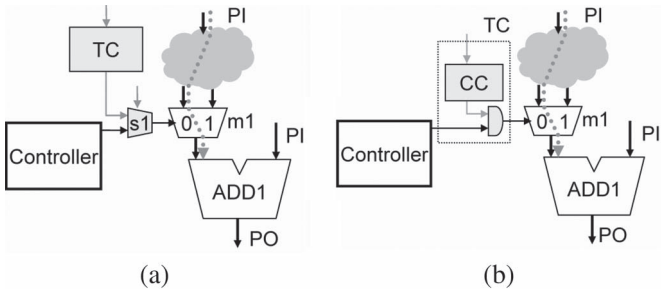


Fig. 3. Comparison of two methods. (a) Method of the strong testability. (b) Proposed method.

An example of the method proposed in [18] and an example of the proposed DFT method are shown in Fig. 3(a) and (b), respectively. The main differences between the two TCs are given as follows: 1) the TC in Fig. 3(a) is a sequential circuit, whereas the TC in Fig. 3(b) is a combinational circuit; and 2) the TC in Fig. 3(b) provides control signals to activate the control/observation paths, whereas the TC in Fig. 3(a) provides test patterns for $m1$, as well as control signals to activate the control/observation paths. Suppose that MUX $m1$ exists on a control path (the dotted line) for ADD1. To propagate values for ADD1 by using the path, the control signal of $m1$ should be "0." Both the method discussed in [18] and the proposed method provide the "0" from the TC. Furthermore, the control input of $m1$ must be able to be set to both "0" and "1" for testing $m1$ itself. In [18], these test patterns for the control input of $m1$ are also provided from the TC. Therefore, test MUX $s1$ is added for switching the control signal from the controller to the TC [Fig. 3(a)]. On the other hand, in the proposed method, the test patterns for the control input of $m1$ are provided through the normal controller. Therefore, it is sufficient to provide "0" to the control input of $m1$, and only an AND gate is added in between the controller and the data path [Fig. 3(b)].

V. DFT Method

Before explaining the proposed DFT method, we formally present a DFT problem for making a given RTL circuit cycle unrollable as follows.

Definition 13: DFT Problem for Cycle Unrollability:

Input: an RTL circuit.

Output: an augmented RTL circuit that is cycle unrollable.

Optimization: minimizing the hardware overhead.

The proposed DFT method consists of four steps, as explained in Section IV. The details of the four steps are described as follows.

Step 1—Construct a Control Forest: We construct simple paths from PIs to data inputs of each hardware element. The simple path is called a control path, and a set of simple paths is called a control forest. A control path is guaranteed to propagate any value in the range of each signal line. To propagate any value through the control paths, thru functions are added to hardware elements on the path if necessary. Then, we choose the control signals for the hardware elements on the path. From these control signals, a TC is generated in step 4). When we construct a control forest, we start searching for control paths from a PI. Then, we decide on a path from the PI to a hardware element. We try to minimize the number of additional thru functions by giving high priority to the hardware elements with thru functions for the path selection during the construction of the control forest.

An example of the control forest for Fig. 1 is shown in Fig. 4(a). In this example, a thru function is added to SUB. An additional path from PI1 to SR by the test MUX (TM1) is also added.

Step 2—Construct an Observation Forest: We construct simple paths from the outputs of each hardware element to POs. The simple path is called an observation path, and a set of simple paths is called an observation forest. An observation path is guaranteed to propagate any value in the range of each signal line to a PO. To propagate any error by using observation paths, thru functions are added to hardware elements on the paths if necessary. Then, we choose the control signals for the hardware elements on the paths. From these control signals, a TC is generated in step 4). When we construct an observation forest, we start searching for observation paths from a PO. Then, we decide on a path from the PO to a hardware element. We try to minimize the number of additional thru functions and the required control signals by sharing the control paths and the observation paths as much as possible during the construction of the observation forest. An example of the observation forest construction for Fig. 1 is shown in Fig. 4(b). Additional observation paths from the CL to the POs are added.

Step 3—Resolve the Dependence: If there exist reconvergent paths in the control forest or the observation forest, and if the sequential depths of the paths are the same, then dependence exists between the paths. If dependence exists, we decide which registers can resolve the dependence and the number of the hold cycles to resolve them. The control signals to the registers for resolving the dependence are provided from the TC, which is generated in step 4). Moreover, the information on the hold cycles is used for the TEM generation shown in Section VI. In the proposed method, we try to resolve the dependence by using HRs as much as possible to reduce additional hold functions. In Fig. 6, a hold function is added to the SR by using test MUX TM2.

Step 4—Generate a TC: This section describes the generation of a TC, which provides the control signals for the control forest and the observation forest and for resolving dependence.

A TC is a combinational hardware element, and the output patterns required for the TC are selected by additional PIs. Let n be the number of output patterns of a TC. Then, the number of additional PIs is $\lceil \log_2 n \rceil$. The TC is inserted between the CL and the additional observation paths added in step 3).

Moreover, the observation paths and the control signal lines for the additional hardware elements are also added from the TC.

An example of the TC for the RTL circuit in Fig. 1 is shown in Fig. 5. In Fig. 5, the patterns in rows 1–3 are the control signals for the control forest and the observation forest and for resolving dependence, respectively. The patterns in rows 4 and 5 are the control signals for testing the RTL circuit. The pattern in the last row refers to the control signals for the

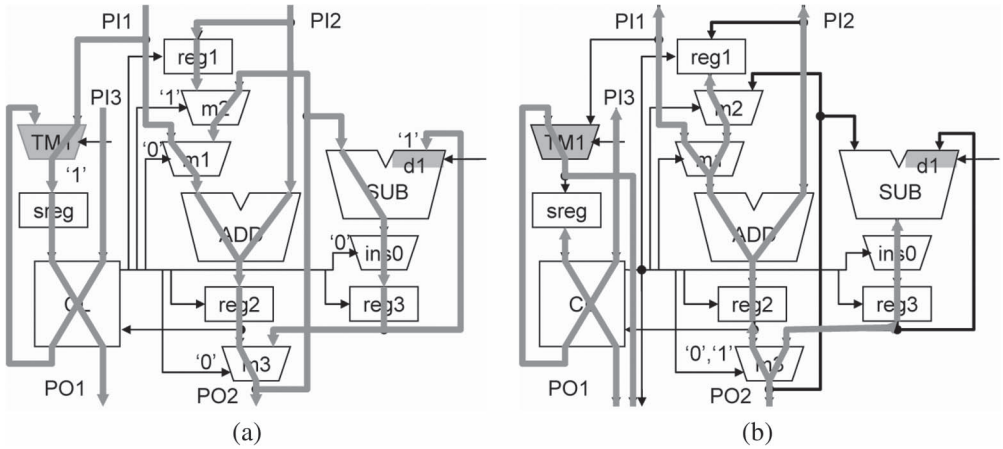


Fig. 4. Examples of the forests. (a) Control forest. (b) Observation forest.

m1	m2	m3	ins0	reg1	reg2	reg3	d1	TM1	TM2
X	X	1	X	X	X	X	X	1	1
0	0	0	0	1	1	1	1	1	0
0	0	0	0	0	1	1	1	1	0
T	T	T	T	1	1	1	0	0	0
T	T	T	T	1	1	1	0	1	0
T	T	T	T	T	T	T	0	0	X

T : output pattern of the CL
X : don't care

Fig. 5. Example of the output patterns of a TC.

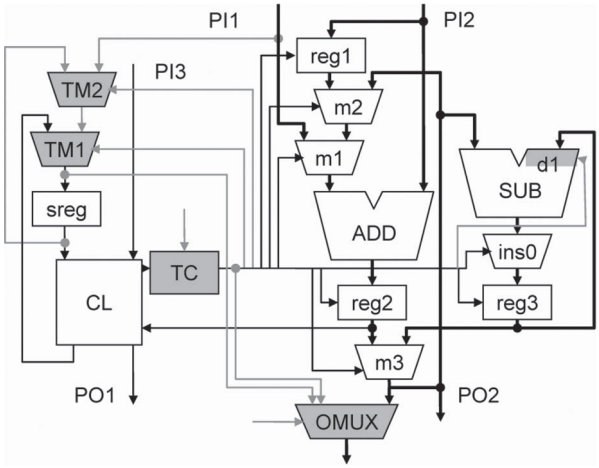


Fig. 6. Example of a loop-unrollable RTL circuit.

normal operation. In this example, the number of additional PIs is three because the number of the output patterns of the TC is six. Moreover, by adding the TC, at most two gates—an AND gate and an OR gate—are inserted between the controller and the data path.

An example of a cycle-unrollable RTL circuit is shown in Fig. 6. To reduce the pin overhead for additional observation paths, a MUX (OMUX) is added to the PO in the data paths to observe the additional observation paths.

The OMUX is controlled with additional PIs. By adding the OMAX, the pin overhead becomes the number of control inputs of the OMUX instead of the bit width of the additional observation paths.

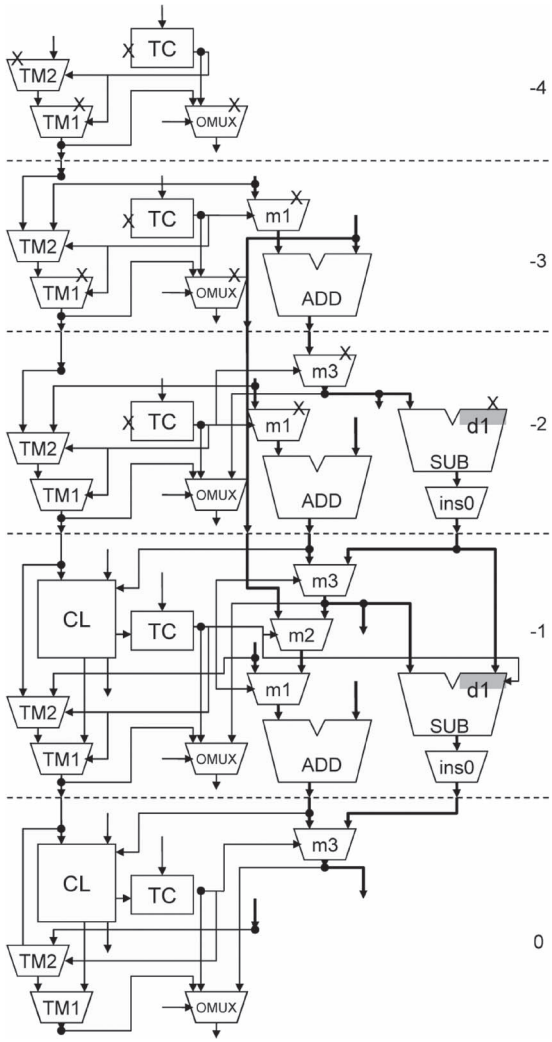


Fig. 7. Example of the time expansion model.

VI. Test Generation Method

For the test generation of the cycle-unrollable RTL circuit, we generate a TEM, and a combinational ATPG is applied to the TEM. The TEM is generated as follows. First, we generate a TEM that consists of a PO with time frame 0. Then, a hardware element m connected to the PO and the signal line between m

TABLE I
CIRCUIT CHARACTERISTICS

Circuits	Area [#gates]	Controller						Data Path					
		#PI	#PO	#FF	#Status	#Control	Area	#PI	#PO	|bit|	#Reg.	#Mod.	Area
GCD	1127.0	1	1	2	3	7	116.3	32	16	16	3	8	1127.0
LWF	1413.3	1	0	2	0	8	49.7	32	32	16	5	8	1363.6
JWF	4322.5	1	0	3	0	38	172.0	80	80	16	14	28	4150.5
Paulin	4430.6	1	0	3	0	16	107.6	32	32	16	7	15	4323.0
RISC	40827.9	1	2	4	54	62	1463.9	32	96	32	40	107	39364.0
MPEG	52169.5	6	0	8	0	271	3459.8	56	148	8	241	368	47883.9

TABLE II
HARDWARE OVERHEADS

Circuits	Area overhead [%]											Pin overhead [#]		
	Full scan	[18]					Proposed					Full scan	[18]	Proposed
			C	DP	TC	MUX		C	DP	TC	MUX			
GCD	26.6	39.7	1.1	2.6	23.2	12.8	8.6	1.4	0.0	4.3	2.8	3	5	4
LWF	26.7	37.1	0.4	5.2	21.9	9.7	6.3	1.1	0.0	2.7	2.5	3	5	4
JWF	33.4	48.6	0.8	18.1	21.1	8.6	6.7	0.5	0.0	2.9	3.3	3	5	4
Paulin	20.8	31.7	1.1	3.4	19.4	7.8	5.7	0.5	1.7	1.7	1.7	3	5	4
RISC	16.7	27.3	0.1	10.9	12.5	3.6	3.3	0.1	0.0	2.2	1.1	3	6	6
MPEG	19.7	24.9	0.2	4.0	13.0	7.2	5.1	0.2	0.6	2.4	1.9	3	7	6

and the PO are added to the TEM. When m is added to the TEM for the first time, all the hardware elements connected with inputs of m are added to the TEM. When m, which already exists in the TEM, is added to the TEM again, only the hardware elements connected to the control forest are added to the TEM. Moreover, all the hardware elements that exist on the observation path for m and the corresponding signal lines are also added to the TEM. If m is a register, then only the signal line connected with m is added to the TEM, and the time frame is decreased by 1. If m is used to resolve the dependence, the time frame is decreased by the corresponding number of hold cycles. This process repeats until m becomes a PI, and all POs are added to the TEM. A combinational ATPG is applied to the generated TEM. Then, generated test patterns are transformed so that the test patterns can be applied to the original RTL circuit.

An example of the TEM for Fig. 6 is shown in Fig. 7. Consider the test generation process for a fault in module SUB, which exists on a cycle in the circuit. To generate a test for the fault, we need to control reg2, reg3, and sreg for the left data input, the right data input, and the control input, respectively. From the control forest shown in Fig. 4(a), reg2 is controllable from PI2 and PI1 through $m1$, and reg3 is controllable from reg2 through $m3$, SUB, and ins0. sreg is also directly controllable from PI1 through TM1. Similarly, we need to observe reg3, and reg3 is observable at PO2 through $m3$ from the observation forest shown in Fig. 4(b). The TEM shown in Fig. 7 effectively guides these propagation paths during the test generation process. The faults in SUB are assumed to be activated at time frame -1. Before that, at time frame -4, the TEM includes the control path for sreg. At time frame -3, the control paths for reg2 from PI2 and PI1 through $m1$ and ADD can be realized. The TEM includes the control paths for reg3 from reg2 through $m3$, SUB, and ins0 at time frame -2. At time frame -2, the control path for reg2 is also included to avoid the dependence between reg2 and reg3. Then, the fault in SUB is assumed to be activated by reg2, reg3, and sreg, and the error is assumed to be propagated to reg3 at time frame -1. The observation path from reg3 to PO2 through $m3$ is provided at time frame 0. Therefore, we can easily generate a test for the fault in SUB using the TEM, although SUB exists on a cycle in the circuit.

VII. EXPERIMENTAL RESULTS

We evaluated the effectiveness of the proposed method by experiments. RTL benchmark circuits used for the experiments are circuits named GCD, LWF, JWF, and PAULIN, which are popularly used circuits, in addition to two circuits named RISC and MPEG, which are more practical and larger circuits designed by a semiconductor company. In our previous work [8], these benchmark circuits were used. The circuit characteristics of these circuits are shown in Table I. "#PI" and "#PO" denote the numbers of PIs and POs, respectively. "#FF," "#status," and "#Control" denote the numbers of FFs, status inputs, and control outputs, respectively. "#Reg" and "#Mod." denote the number of registers and the number of hardware elements except for registers, respectively. "bit" denotes the bit width of the data paths. In our experiments, we used AutoLogicII (MentorGraphics) as a logic synthesis tool with its sample libraries to synthesize those circuits. In Table I, "Area" denotes the total circuit size. We compared the proposed method with original circuits, the full-scan method, and the method in [18]. In the full-scan method, all the FFs in the circuits are replaced by the scan-FFs, and a single scan chain is constructed.

The results of the hardware overhead are shown in Table II. "C," "DP," "TC," and "MUX" denote the area overhead of the controller, the data path, the TC, and the additional MUXs, respectively. The area overhead of the proposed method is much smaller than others. Compared to [18], we can see that the reduction of the area overhead mainly comes from the TC and

TABLE III
TEST GENERATION RESULTS

Circuits	Fault efficiency [sec]				Test generation time [sec]				Test application time [cycles]			
	Original	Full scan	[18]	Proposed	Original	Full scan	[18]	Proposed	Original	Full scan	[18]	Proposed
GCD	83.39	100.00	100.00	100.00	3070.07	0.27	1.13	1.72	421	4232	456	588
LWF	99.05	100.00	100.00	100.00	85.45	0.17	0.89	1.01	392	2904	295	108
JWF	96.16	100.00	100.00	100.00	2873.34	0.88	1.22	5.91	412	20975	1000	675
Paulin	96.55	100.00	100.00	100.00	4290.51	0.53	1.60	8.07	201	6147	1136	798
RISC	63.95	99.97	100.00	99.97	156808.67	98870.71	105.26	166.88	6928	1233859	7914	4345
MPEG	74.48	100.00	100.00	100.00	195260.82	55.72	17.64	1208.90	148	462942	150019	8515

the additional MUXs. Consequently, the delay overhead of the proposed method is lower than that in [18]. This is because at most two gates are inserted in between the controller and the data path in the proposed method, whereas at most four gates are inserted in between them in [18]. We can also observe that the pin overhead is smaller than [18]. The pin overhead of RISCs and MPEGs in the proposed method is larger than that of other benchmark circuits. In RISCs, the pin overhead for the TC increased because of the number of control signals utilized for hold functions. Moreover, in RISCs and MPEGs, two or more OMUXs are required because the bit width of the control outputs of the TC is larger than the bit width of the POs of the data paths.

The results of the test generation are shown in Table III. We used TestGen (Synopsys) as a sequential and combinational ATPG tool on Sun Blade 2000 (Sun Microsystems). Test generation for sequential circuits using a TEM requires a combinational ATPG, which can deal with gate-level multiple stuck-at faults. In these experiments, since TestGen cannot deal with gate-level multiple stuck-at faults, we use the circuit model, which can express a gate-level multiple stuck-at fault in a TEM as a single stuck-at fault [22], [29]. "Test generation time" denotes the time spent for the ATPG and does not include the time spent for the DFT. However, the time spent for the DFT is negligible as compared with the time spent on the ATPG. We observe that the full-scan method, the method in [18], and the proposed method can achieve 100% FE, except for RISCs in practical test generation time. For RISCs, the full-scan method and the proposed method cannot generate test vectors for a subset of CMs in the data path. For MPEGs, the TEM becomes large since the sequential depth of a simple path from a PI to a PO is very big. Therefore, the test generation time of the proposed method becomes long compared with the full-scan method. Although the proposed method guarantees the existence of a linear-depth TEM for any testable fault, the depth depends on the circuits' characteristics. From the results for two industrial circuits, we can say that the test generation time is reasonable for the circuits where the depths from PIs to POs are shallow such as RISCs. On the other hand, for the circuits where the depths from PIs to POs and the length of cycle-unrolling paths are long such as MPEGs, the size of the TEM becomes large. Hence, it results in long test generation time. This is a limitation of the proposed method. However, we can relax this limitation by proposing a DFT method that makes the cycle-unrolling paths shorter.

The proposed method can reduce the test application time by up to 99.6% compared with the full-scan method. This is because the proposed method does not require scan-shift operations. The proposed method can allow at-speed testing because the test pattern can be applied not by the scan clock but by the operational clock. The proposed method can also reduce the test application time by up to 94.3% compared with the method in [18] for five circuits. We consider that faults were efficiently detected by the fault simulation in the proposed method since the whole circuit is the target for test generation, whereas the method discussed in [18] is based on hierarchical test generation. On the other hand, the proposed method requires longer test generation time compared with the previous methods. However, the test application time is a per-chip cost, whereas the test generation time is a one-time cost for a design. Therefore, the proposed method can effectively reduce the total test cost.

VIII. CONCLUSION

In this paper, we have introduced a new class of sequential circuits, called the linear-depth time-bounded class, which has been shown to be acyclically testable, and hence, the test generation complexity is τ^2-bounded. As one of the linear-depth time-bounded classes, we have introduced a new class of RTL circuits, called the cycle-unrollable RTL circuits, which has been shown to be linear depth time bounded and, hence, τ^2-bounded. We have also proposed a DFT method and a test generation method based on the cycle unrollability. The proposed method can achieve 100% FE for gate-level single stuck-at faults in practical test generation time by using a combinational ATPG. It also allows at-speed testing. Furthermore, the proposed method can drastically reduce hardware overhead and test application time compared to that discussed in [18] by paying an acceptable cost of test generation time. The test application time and the hardware overhead are costs for every manufactured chip, whereas the test generation time is a one-time cost for a design before manufacturing. Therefore, the proposed method is effective.

REFERENCES

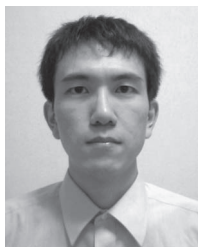
[1] H. Fujiwara, *Logic Testing and Design for Testability*. Cambridge, MA: MIT Press, 1985.
[2] P. Goel, "Test generation costs analysis and projections," in *Proc. 17th Des. Autom. Conf.*, Jun. 1980, pp. 77–84.
[3] M. R. Prasad, P. Chong, and K. Keutzer, "Why is ATPG easy?" in *Proc. 36th Des. Autom. Conf.*, Jun. 1999, pp. 22–28.
[4] R. Gupta and M. A. Breuer, "The BALLAST methodology for structured partial scan design," *IEEE Trans. Comput.*, vol. 39, no. 4, pp. 538–544, Apr. 1990.

[5] A. Balakrishnan and S. T. Chakradhar, "Sequential circuits with combinational test generation complexity," in *Proc. IEEE Int. Conf. VLSI Des.*, Jan. 1996, pp. 111–117.
[6] H. Fujiwara, "A new class of sequential circuits with combinational test generation complexity," *IEEE Trans. Comput.*, vol. 49, no. 9, pp. 895–905, Sep. 2000.
[7] M. Inoue, C. Jinno, and H. Fujiwara, "An extended class of sequential circuits with combinational test generation complexity," in *Proc. 20th Int. Conf. Comput. Des.*, Sep. 2002, pp. 200–205.
[8] C. Y. Ooi and H. Fujiwara, "Classification of sequential circuits based on τ^k notation," in *Proc. IEEE 13th Asian Test Symp.*, Nov. 2004, pp. 348–353.
[9] C. Y. Ooi, T. Clouqueur, and H. Fujiwara, "Classification of sequential circuits based on τ^k notation and its applications," *IEICE Trans. Inf. Syst.*, vol. E88-D, no. 12, pp. 2738–2747, Dec. 2005.
[10] T. Inoue, D. K. Das, T. Mihara, C. Sano, and H. Fujiwara, "Test generation for acyclic sequential circuits with hold registers," in *Proc. Int. Conf. Comput.-Aided Des.*, Nov. 2000, pp. 550–556.
[11] C. Y. Ooi and H. Fujiwara, "A new class of sequential circuits with acyclic test generation complexity," in *Proc. 24th IEEE Int. Conf. Comput. Des.*, Oct. 2006, pp. 425–431.
[12] S. Dey and M. Potkonjak, "Non-scan design-for-testability of RT-level data paths," in *Proc. Int. Conf. Comput.-Aided Des.*, Nov. 1994, pp. 640–645.
[13] R. B. Norwood and E. J. McCluskey, "Orthogonal scan: Low overhead scan for data paths," in *Proc. Int. Test Conf.*, 1996, pp. 659–668.
[14] R. B. Norwood and E. J. McCluskey, "High-level synthesis for orthogonal scan," in *Proc. 15th VLSI Test Symp.*, 1997, pp. 370–375.
[15] J. Lee and J. H. Patel, "Hierarchical test generation under architectural level functional constraints," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 15, no. 9, pp. 1144–1151, Sep. 1996.
[16] I. Ghosh, A. Raghunathan, and N. K. Jha, "Design for hierarchical testability of RTL circuits obtained by behavioral synthesis," in *Proc. IEEE. Int. Conf. Comput. Des.*, 1995, pp. 173–179.
[17] I. Ghosh, A. Raghunathan, and N. K. Jha, "A design for testability technique for RTL circuits using control/data flow extraction," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 17, no. 8, pp. 706–723, Aug. 1998.
[18] S. Ohtake, H. Wada, T. Masuzawa, and H. Fujiwara, "A non-scan DFT method at register-transfer level to achieve complete fault efficiency," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2000, pp. 599–604.
[19] S. Ohtake, T. Masuzawa, and H. Fujiwara, "A non-scan DFT method for controllers to achieve complete fault efficiency," in *Proc. 7th Asian Test Symp.*, 1998, pp. 204–211.
[20] H. Wada, T. Masuzawa, K. K. Saluja, and H. Fujiwara, "Design for strong testability of RTL data paths to provide complete fault efficiency," in *Proc. 13th Int. Conf. VLSI Des.*, Jan. 2000, pp. 300–305.
[21] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction of Algorithms*. Cambridge, MA: MIT Press, 1990.
[22] H. Ichihara and T. Inoue, "A method of test generation for acyclic sequential circuits using single stuck-at fault combinational ATPG," *IEICE Trans. Fundam.*, vol. E86-A, no. 12, pp. 3072–3078, Dec. 2003.
[23] H. Iwata, T. Yoneda, and H. Fujiwara, "A DFT method for the time expansion model at the register transfer level," in *Proc. 44th Des. Autom. Conf.*, Jun. 2007, pp. 682–687.
[24] M.S. Abardir and M. A. Breuer, "A knowledge based system for designing testable VLSI chips," *IEEE Des. Test Comput.*, vol. 2, no. 4, pp. 56–68, Aug. 1985.
[25] H. Fujiwara, "Computational complexity of controllability/observability problems for combinational circuits," *IEEE Trans. Comput.*, vol. 39, no. 6, pp. 767–962, Jun. 1990.
[26] S. T. Chakradhar, V. D. Agrawal, and M. L. Bushnell, "Polynomial time solvable fault detection problems," in *Proc. 20th Fault-Tolerant Comput. Symp.*, Jun. 1990, pp. 56–63.
[27] H. B. Min and W. A. Rogers, "A test methodology for finite state machines using partial scan design," *J. Electron. Test., Theory Appl.*, vol. 3, no. 2, pp. 127–137, May 1992.
[28] Y. C. Kim, V. D. Agrawal, and K. K. Saluja, "Combinational automatic test pattern generation for acyclic sequential circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 6, pp. 948–956, Jun. 2005.
[29] Y. C. Kim, V. D. Agrawal, and K. K. Saluja, "Multiple faults: Modeling, simulation and test," in *Proc. 7th ASP-DAC/15th Int. Conf. VLSI Des.*, Jan. 2002, pp. 592–597.
[30] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Norwell, MA: Kluwer, 2000.

Hideo Fujiwara (S'70–M'74–SM'83–F'89) received the B.E., M.E., and Ph.D. degrees in electronic engineering from Osaka University, Osaka, Japan, in 1969, 1971, and 1974, respectively.

From 1974 to 1985, he was with Osaka University. From 1985 to 1993, he was with Meiji University, Tokyo, Japan. Since 1993, he has been with the Nara Institute of Science and Technology, Kansai Science City, Japan, where he is currently a Professor with the Graduate School of Information Science. He served as an Editor of the *Journal of Electronic Testing: Theory and Application* and the *Journal of Circuits, Systems and Computers* from 1989 to 2004 and the *VLSI Design: An Application Journal of Custom-Chip Design, Simulation, and Testing* from 1992 to 2005. He also served as a Guest Editor for the Special Issues of the *IEICE Transactions of Information and Systems*. He is the author of *Logic Testing and Design for Testability* (MIT Press, 1985). His research interests include logic design, digital systems design and test, VLSI CAD, and fault-tolerant computing, which includes high-level/logic synthesis for testability, test synthesis, design for testability, built-in self-test, test pattern generation, parallel processing, and computational complexity.

Dr. Fujiwara is currently an Advisory Member of the Institute of Electronics, Information and Communication Engineers of Japan (IEICE) *Transactions on Information and Systems*. He is a Golden Core Member of the IEEE Computer Society, a Fellow of the IEICE, and a Fellow of the Information Processing Society of Japan. He served as an Editor of the IEEE TRANSACTIONS ON COMPUTERS from 1998 to 2002. He was the recipient of the 1977 Institute of Electronic Communications Engineers Young Engineer Award, the 1991, 2000, and 2001 IEEE Computer Society Certificate of Appreciation Awards, the 1994 Okawa Prize for Publication, the 1996 and 2005 IEEE Computer Society Meritorious Service Awards, the 2001 IEEE Computer Society Outstanding Contribution Award, and the 2005 IEEE Computer Society Continuing Service Award.



Hiroyuki Iwata (S'05) received the B.E. degree in electrical and computer engineering from the Yokohama National University, Yokohama, Japan, in 2003 and the M.E. and Ph.D. degrees in information science from the Nara Institute of Science and Technology, Nara, Japan, in 2005 and 2007, respectively.

Since 2007, he has been with the Renesas Technology Corporation, Tokyo, Japan. His research interests include VLSI CAD and design for testability.

Tomokazu Yoneda (M'04–SM'08) received the B.E. degree in information systems engineering from Osaka University, Osaka, Japan, in 1998 and the M.E. and Ph.D. degrees in information science from the Nara Institute of Science and Technology, Kansai Science City, Japan, in 2001 and 2002, respectively.

He is currently an Assistant Professor with the Graduate School of Information Science, Nara Institute of Science and Technology. His research interests include VLSI CAD, design for testability, and SoC test.

Dr. Yoneda is a member of the Institute of Electronics, Information and Communication Engineers of Japan.

Chia Yee Ooi (S'04–M'07) received the B.E. and M.E. degrees in electrical engineering from the Universiti Teknologi Malaysia, Skudai, Malaysia, in 2001 and 2003, respectively, and the Ph.D. degree in information science from the Nara Institute of Science and Technology, Kansai Science City, Japan, in 2006.

She is currently a Lecturer with the Universiti Teknologi Malaysia. Her research interests include VLSI design and testing.