Regular Paper

Non-scan Design for Single-Port-Change Delay Fault Testability

Yuki Yoshikawa,† Satoshi Ohtake,† Michiko Inoue† and Hideo Fujiwara†

We propose a non-scan design-for-testability (DFT) method at register-transfer level (RTL) based on hierarchical test generation: the DFT method makes paths in a data path single-port-change (SPC) two-pattern testable. For combinational logic in an RTL circuit, an SPC two-pattern test launches transitions at the starting points of paths corresponding to only one input port (an input, which has some bits, of an RTL module) and sets the other ports stable. Hence, during test application, the original hold function of a register can be used for stable inputs if the hold function exists. Our DFT method can reduce area overhead compared to methods that support arbitrary two-pattern tests without losing the quality of robust test and non-robust test. Experimental results show that our method can reduce area overhead without losing the quality of test. Furthermore, we propose a method of reducing over-test by removing a subset of sequentially untestable paths from the target of test.

1. Introduction

The speed of VLSI circuits has been increased in recent years. A high-speed circuit needs delay testing to verify that its logic operates correctly at the desired clock speed. A path delay fault, a defect that cumulative propagation delays along a path exceed an upper limit [1], is one of the delay fault models and can model the delay between two flip-flops (FFs). To detect a path delay fault, a vector pair (two-pattern test) is required for FFs that are the starting points of the target path and other related paths. However, it is impossible to apply any two-pattern tests to the starting points. To enhance two-pattern testability for FFs, there are the functional justification [3] and the scan shift [2] techniques with standard scan. However, these techniques cannot still guarantee the application of any two-pattern. The enhanced scan [4] (ES) approach that can apply any two-pattern incurs high area overhead. Moreover, scan approaches cause long test application time because of scan-shift operation.

Non-scan design-for-testability (DFT) approaches at register-transfer level (RTL) based on hierarchical test generation have been proposed [5),6]. The approaches utilize the data flow at RTL to test a circuit. The advantages are that the number of primitive elements at RTL is much smaller than that at gate level, and a number of gate-level paths between two registers are regarded as a bundled path, which is called RTL path [6]. An RTL path is a path passing through only combinational logic, which starts at a primary input (PI) or a register and ends at a register or a primary output (PO). Hierarchical test generation consists of two processes: (i) generating test patterns for combinational blocks at gate-level, (ii) generating control sequences to justify the generated patterns from PIs to registers that are inputs of every combinational block, and generating observation paths to propagate the responses to POs. Our previous work [6] defined hierarchically two-pattern testable (HTPT) data path, in which any two-pattern tests can be applied to every combinational block from PIs and the responses can be observed at POs. We presented a DFT method to augment a given data path to an HTPT data path, which requires lower area overhead and test application time than enhanced scan approach does.

In this paper, we introduce a new concept of testability called single-port-change (SPC) two-pattern testability. A port means an input or an output of a primitive element at RTL, and it has a bit width. We propose a DFT method that guarantees to make every RTL path SPC two-pattern testable. For a target RTL path in a combinational block, an SPC two-pattern test launches transitions at the starting points of paths corresponding to the RTL path while keeping the other related ports stable. The method of generating SPC two-pattern tests for a combinational block is explained in Section 3.1, and how to generate control and observation paths is shown in Section 5. During

† Nara Institute of Science and Technology

test application for each combinational block, the original hold function of a register can be used for stable inputs if the hold function exists on its control path. Hence, our proposed DFT method can reduce area overhead than that of HTPT using arbitrary two-pattern tests. According to the quality of two-pattern tests, testable path delay faults are generally classified into three classes: robust testable, non-robust testable and functional sensitizable (FS) [1]. SPC two-pattern tests can guarantee robust (resp. non-robust) test for a path if the path is robust (resp. non-robust) testable and can also detect a subset of FS path delay faults (shown in Section 3.2).

We also address the reduction of over-testing. We propose a method of identifying RTL paths that never propagate a value from the starting register to the ending register within one clock period at normal operation. We refer to such paths as control-dependent untestable paths (CUPs). By removing CUPs from the target of test, over-testing is reduced and test application time is also reduced. Moreover, it may be possible to reduce hardware overhead if an RTL path that cannot apply SPC two-pattern tests without DFT is judged as CUP.

Our experimental results show that the proposed method can reduce area overhead and test application time compared to those for HTPT.

2. Target Circuit and Fault

An RTL design generally consists of a controller and a data path, and they are connected each other by control signal lines and status signal lines. Our target part is the data path separated from the controller part. All the control signals and the status signals of the data path are assumed to be directly controllable and directly observable, respectively. In order to realize controllability and observability of control signals and status signals, respectively, we need some mechanism to generate control signals and to observe status signals in test mode. In this paper, the implementation of such mechanism is not considered.

A data path consists of hardware elements (e.g., PIs, POs, registers, multiplexers, operational modules, and observation modules) and lines to connect output ports of hardware elements with input ports of others. There are two types of input ports of a hardware element: data input ports and control input ports. Each data input port is reachable directly or indirectly from at least one PI. Each control input port is connected with control signal line. Similarly, there are two types of output ports of a hardware element: data output ports and status output ports. Each data output port is reachable directly or indirectly to at least one PO. Each status output port is connected with status signal line. An operational module has one or two data input ports, one data output port and at most one status output port, and an observation module has one or two data input ports, one status output port, at most one control input port. We assume that (i) all lines have same bit width. (ii) There is no chaining of operational modules. Note that chaining modules can be regarded as n input and one output operational module. We relax the second assumption by extending the consideration of two input modules. We target all the path delay faults except for faults on paths that start at control inputs or end at status outputs.

3. SPC Two Pattern Testability

3.1 SPC Two-pattern Test

In this section, a combinational block that consists of combinational hardware elements on an input cone to a register is considered at RTL. We refer to an RTL path that is a target of testing as *on-path*. As opposed to on-path, we refer to an RTL path that supports the propagation of a transition launched at the starting point of an on-path along the on-path as *off-path*. For the input port of an operational module on an on-path, one of the RTL paths passing through the other port can be an off-path (See the left picture of **Fig. 1**). In this paper, we assume that an operational module has one or two input ports and there is no chaining module, hence the number of off-paths is at most one for each on-path. An SPC two-pattern test is a pair of two consecutive vectors that launches transitions at the port corresponding to the starting point of the on-path and sets stable two consecutive vectors for the other ports of the combinational block. When

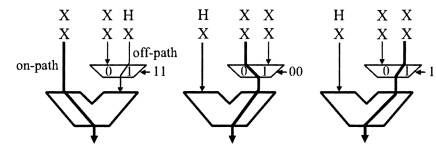

Fig. 1 Constraints of ATPG to generate SPC two-pattern tests.

SPC two-pattern tests are applied to a combinational block, the select signal of each MUX is fixed with an on-path or an off-path being selected. Amin [6] showed that while the select signal of a MUX is fixed, propagation of the signals from the selected input to the output is independent of the signals at the other input. Therefore the on-path is testable if SPC two-pattern tests can be applied to the starting points of the on-path and the off-path.

SPC two-pattern tests for combinational blocks can be generated by using a combinational test generation algorithm with constraints. To describe the constraints, we use the notation X and H. X denotes that it is possible to generate arbitrary vector and H means that the vector just before is held. In Fig. 1, we show an example of constraints for ATPG. XX for an on-path (a bold line in the figure) denotes that it is possible to generate arbitrary two vectors consecutively. XH denotes that the first vector is an arbitrary vector and the second vector is the same as the first one. This is the input constraint for off-path. As we mentioned above, for the inputs other than those on on-path, off-path and the select signal line of each MUX, we do not care generated vectors, hence we denote them as merely XX.

3.2 Quality of SPC Two-pattern Test

Smith [7] showed that a path delay fault is testable by a robust test if and only if there exits a robust single-input change (SIC) test for this fault, and Gharaybeh [8] showed that the same applies to non-robust tests. Their theorems show that there exist SIC robust tests for robust testable path delay faults and SIC non-robust tests for non-robust testable path delay faults. At gate level consideration, an SIC two-pattern test launches a transition for 1 bit of inputs of a combinational block, while an SPC two-pattern test can launch transitions for any bits of inputs of the corresponding port. Hence SPC two-pattern test can completely cover an SIC two-pattern test. In other words, there exists an SPC robust (resp. non-robust) test for a robust (resp. non-robust) testable path delay fault without loss of test quality.

The remaining testable path delay faults are FS path delay faults. To test these faults, transitions are needed at multiple inputs. A FS path delay fault that needs transitions for some inputs of only on-path can be tested using an SPC two-pattern test. However, faults that need transitions for some inputs of both an on-path and an off-path cannot be tested under the concept. We will experimentally examine how many FS faults become untestable.

3.3 SPC Two-pattern Testability

We define SPC two-pattern testability for RTL paths. To test an RTL path that does not pass through an operational module with two input ports, one control path and one observation path are sufficient to test the RTL path. Control paths are the paths to justify test patterns from PIs to each register and observation paths are the paths to propagate the responses to POs. If we consider only one control path, we need not care about timing conflict to justify test patterns. Timing conflict means that more than or equal to two values are required to the same PI at the same time. Hence it is certainly possible to generate a control path by using a *thru* function [10], A thru function is added to an operational module in order to propagate a value along a control path or an observation path without changing the value. The realization of a thru function is shown in Section 5. To test an RTL path $p \in P$ that passes through an operational module having two input ports, it is necessary to justify test patterns from a PI or PIs to appropriate registers by a pair of control paths C_1, C_2 and propagate test responses from an appropriate register to a PO by an observation path O_p, where C_1 is the path from a PI to the starting register of an on-path p, and C_2 is the path from a PI to the starting register of an off-path.

Definition 1: An RTL path p is SPC two-pattern testable if there exists a pair of control paths C_1 and C_2 that can apply SPC two-pattern tests to the combinational block and O_p that can observe the test responses.

3.3.1 Conditions for Control Paths

Here, to simplify the following discussion, we assume that there exists a thru function for each input port of every operational module in a data path. In the next section, we will propose an efficient DFT algorithm to add thru function to data paths. In order to support the application of SPC two-pattern tests with a pair of control paths C_1 and C_2, the difference between the sequential depths of C_1 and that of C_2 and/or the number of *hold* registers on C_1 and that on C_2 should be considered. The sequential depth of a control path C_i is the number of registers that appear on C_i and is denoted as $SD(C_i)$. Let EP_1 and EP_2 be the ending point of C_1 and that of C_2, respectively. If C_1 and C_2 are not

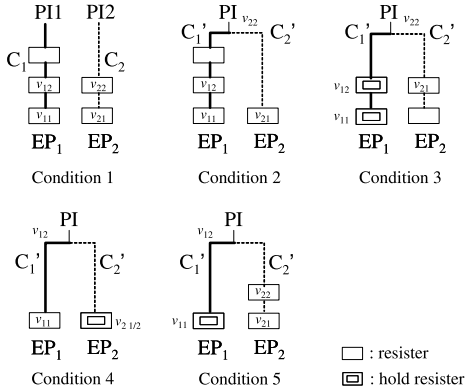

Fig. 2 Conditions for C_1 and C_2.

disjoint, let C_1' and C_2' be the paths from the diverging point of C_1 and C_2 to EP_1 and EP_2, respectively. In the following theorem, we show necessary and sufficient conditions for a pair of control paths C_1 and C_2 to support SPC two-pattern tests.

Theorem 1: A pair of control paths C_1 and C_2 can justify SPC two-pattern tests to their ending points EP_1 and EP_2 if and only if C_1 and C_2 satisfy one of the following five conditions.

(1) C_1 and C_2 are disjoint.
(2) $|SD(C_1') - SD(C_2')| \geq 2$
(3) There exist at least two hold registers on C_1'.
(4) There exists at least one hold register on C_2'.
(5) There exists at least one hold register on C_1' and $SD(C_2') - SD(C_1') = 1$

Examples of these conditions are shown in **Fig. 2**.

Proof: An arbitrary SPC two-pattern test (V_1, V_2) is represented as $V_1 = v_{11}\&v_{21}$ and $V_2 = v_{12}\&v_{22}$. v_{11} and v_{12} are applied to an on-path. v_{21} and v_{22} are applied to an off-path and they are the same value.

Sufficiency: Since we assume that there exists a thru function between each input and the output of every operational module, we have only to consider timing conflicts. If C_1 and C_2 satisfy Condition 1, it is obviously possible to justify any SPC two-pattern test from PIs to EP_1 and EP_2 (see Condition 1 of Fig. 2). With regard to Conditions 2, 3, 4 and 5, although C_1 and C_2 are not disjoint, it is also possible to justify any SPC two-pattern test without a timing conflict. In Condition 2, we first apply the first and the second partial vectors consecutively to the PI for the control path with higher sequential depth. Then we apply consecutively the remaining two vectors to the same PI. In Condition 3, we first load v_{11} and v_{12} into two hold registers on C_1' and hold the values, secondly we apply consecutively v_{21} and v_{22} to the PI. In Condition 4, we first load v_{21} into hold register on C_2' and hold the value. Then we apply v_{11} and v_{12} consecutively. In Condition 5, we first load v_{11} into hold register on C_1' and hold v_{11}, then we apply v_{21}, v_{22} and v_{12} consecutively.

Necessity: We assume that two control paths C_1 and C_2 do not satisfy any of the above five conditions. Such control paths satisfy all the following properties.

(1) C_1 and C_2 are not disjoint.
(2) $|SD(C_1') - SD(C_2')| < 2$
(3) The number of hold registers on C_1' is at most one.
(4) There is no hold register on C_2'.
(5) There is no hold register on C_1' if $SD(C_2') - SD(C_1') = 1$.

All the possible pairs of control paths C_1 and C_2 that satisfy all the above properties are as follows.

- C_1 and C_2 are not disjoint and $|SD(C_1') - SD(C_2')| = 1$ and there is no hold register on both C_1' and C_2'.
- C_1 and C_2 are not disjoint and $|SD(C_1') - SD(C_2')| = 0$ and there is no hold register on both C_1' and C_2'.
- C_1 and C_2 are not disjoint and $SD(C_1') - SD(C_2') = 1$ and there is only one hold register on C_1'.
- C_1 and C_2 are not disjoint and $|SD(C_1') - SD(C_2')| = 0$ and there is only one hold register on C_1'.

Any pair of control paths C_1 and C_2 described above can not guarantee SPC two-pattern test. Therefore five conditions are the only conditions for a pair of control paths C_1 and C_2 to justify SPC two-pattern tests from a PI or PIs to EP_1 and EP_2. □

Here we consider relaxation of the assumption of the number of input ports of an operational module. The following theorem shows the sufficient conditions for an operational module with n input ports.

Theorem 2: n control paths support the application of SPC two-pattern tests for an RTL path p if either of the following conditions is satisfied.

(1) Any pair of n control paths are disjoint.
(2) With regard to each pair of control paths for off-paths that are not disjoint, the mutually disjoint parts from the diverg-

ing point to both ending points cross at least one hold register.

The proof of this theorem is similar to that of Theorem 1.

As we mentioned in this subsection, to guarantee SPC two-pattern test, a register with hold function is needed even if the difference between sequential depths of C_1 and that of C_2 is zero. However to guarantee arbitrary two-pattern test in such case, we need more complex hardware element for DFT.

3.3.2 Conditions for Observation Paths

To observe a test response, the value captured at the ending register of an RTL path has to be propagated to a PO without changing its value. Fortunately, we need not care about timing conflict because only one observation path is sufficient to propagate the value. Hence to guarantee the propagation, it is sufficient to add a thru function to each operational module on the observation path.

4. The Conditions to Identify CUPs

We can obtain information about state transitions of a controller and control signals from the controller to a data path at each state by analyzing the RTL description of the circuit. By considering the timing of data transfer between registers and the structure of a data path, we identify RTL paths as control-dependent untestable paths (CUPs). Some control signals may depend on status signals. Status signals are determined depending on data in a data path. Such control signals are not determined uniquely by analyzing a controller part alone. In this paper, we eliminate such control signals during CUP identification.

Let P be a set of RTL paths in a data path. Now, we consider whether $p \in P$ is a CUP or not. Let R_s be the register that is the starting point of p, and let R_e be the register that is the ending point of p. Let C_{Rs} and C_{Re} be load enable signals of registers R_s and R_e, respectively. If the load enable signal of a register is equal to '1', the register loads a value, otherwise, holds its value. Note that in case the register does not have *hold* function, we assume that a load enable signal line is connected to the register, and the value of that signal is always '1'. In case the starting point of p is a PI or the ending point of p is a PO, the PI or the PO is treated as a register with no *hold* function. Let M_i and C_{M_i} $(1 \leq i \leq n)$ be a MUX on p and its select signal, respectively (n is the number of MUX on p). Let $C^k_{M_i}$ be a control value of C_{M_i} at time k. When M_i selects the input on p at k, the value of the select signal is denoted as $C^k_{M_i} = p_{M_i}$. Let S_i and S_j be states of the controller. S_i and S_j is said to be consecutive if there exists a direct transition from S_i to S_j. Let $(C^k_{M_i}, C^{k+1}_{M_i})$ be a select signal pair of consecutive two states.

Definition 2: An RTL path p is control-dependent untestable path (CUP) if either of following two conditions is satisfied for any consecutive two states.

(1) $(C^k_{Rs}, C^{k+1}_{Rs}) = (0,-) \vee (C^k_{Re}, C^{k+1}_{Re}) = (-,0)$ $\quad -$: don't care

(2) $\vee^n_{i=1}\{(C^k_{M_i}, C^{k+1}_{M_i}) \neq (-, p_{M_i})\}$

Theorem 3: All the gate-level paths corresponding to an RTL path p are non-robust untestable if p is CUP.

Proof: For the first condition of Definition 2, R_s does not launch a transition at S_i, or R_e does not capture the response at S_j. For the second condition, p is not selected at S_j and this prevents propagation of transitions from R_s to R_e. Therefore, p is non-robust untestable. □

5. DFT Method for RTL Data Path

In this section, we propose a DFT method that makes RTL paths except for CUPs in data paths SPC two-pattern testable.

5.1 DFT Element

Additional hardware elements of DFT are multiplexer (MUX), hold function and thru function. We use a MUX to make a new RTL path from a PI to a register. A hold function is added to a register for the purpose of holding the value according to need, and it is realized by adding a MUX just before the register to feedback a value from the output to the input. A thru function is explained briefly in Section 4. For a common module, such as adder or multiplier, it is realized by providing a constant value to the other input. It can be provided by adding a mask element. A mask element generates a constant depending on its control signal. For a more complex module or a module with one input port, we cannot realize the thru function by only providing a constant, then we deal with the thru function by bypassing the module using a MUX.

5.2 Algorithm for Adding DFT Elements

The flow of the proposed DFT algorithm is

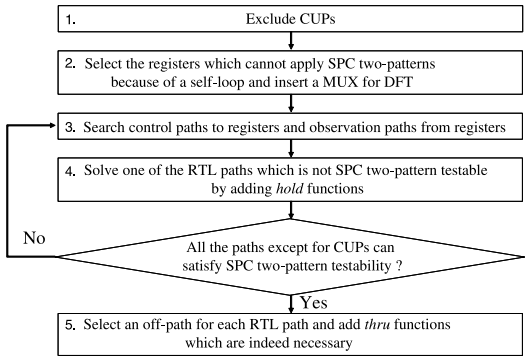

Fig. 3 Flow of our DFT algorithm.

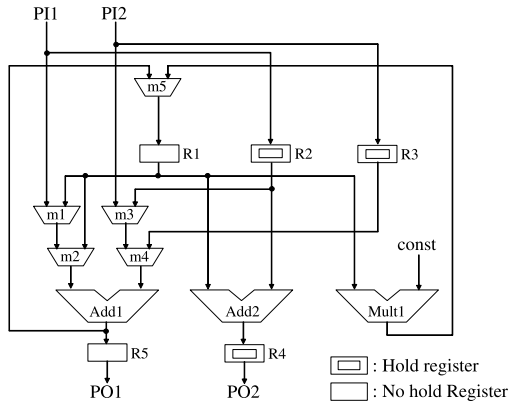

Fig. 4 LWF benchmark circuit.

shown in **Fig. 3**.

Step 1: We extract CUPs according to conditions of Theorem 3, and remove them from the consideration for test.

Step 2: There are some RTL paths that start at a register and go back to the same register. There are many cases where SPC two-pattern tests cannot be applied to such an RTL path because it is structurally difficult to satisfy the conditions of Theorem 1. Since it is only possible to make such an RTL path SPC two-pattern testable by adding MUX (hold function cannot solve this problem) and making a new control path from a PI, we first find such structures. To find RTL paths forming a loop we consider a circuit as a circuit graph consisting of four types of nodes, R, Op, Fo and M, and directed edges. The nodes of type R, Op, Fo and M correspond to a register, an operational module, a fanout and a MUX, respectively, and they are connected by directed edges corresponding to the signal lines of the circuit. We refer to the loop that starts at R-type node and go back to the same node without passing through any other R-type node as a self-loop.

We consider a self-loop. It is impossible to apply SPC two-pattern tests to the RTL path corresponding to the self-loop if there is no M-type node, which can be reached from a PI without passing through the self-loop, between the Op-type node and the R-type node. If one of the RTL paths that starts at R_i and passes through Op_j is not CUP, the RTL path should be modified into SPC two-pattern testable. Such an RTL path can be solved by inserting a MUX between Op_j and R_i, and adding a new path from a PI to the MUX. Here we consider the self-loop, $R1$-$m1$-$m2$-$Add1$-$m5$-$R1$ in **Fig. 4** as an example, and corresponding nodes in its circuit graph are named R_1, M_1, M_2, Op_1 and M_5, respectively. There is no M-type node between Op_1 and R_1 which can be reached from PI without passing through the self-loop. If one of the RTL paths starting at $R1$ is not CUP, a MUX is added to the place between Op_1 and R_1 then a new RTL path PI1-MUX-$R1$ is made. When there are some PIs in a circuit, we select the PI such that the pair of control paths is disjoint to satisfy the first condition of Theorem 1. However if there is only one PI in the circuit, we make a new RTL path from the PI. In this case, if the pair of control paths may not satisfy any conditions from second to fifth, hold function is added in Step 4.

Step 3: In this step, candidates for control and observation paths to each register are selected using heuristics. The decision of control and observation paths will be made in Step 4.

In order to reduce area overhead and test application time, control paths is selected as they form trees whose source nodes are PIs, accordingly each register is reachable from a PI via a control path with the minimum sequential depth. To search such control paths, we represent the data path as a port graph $G = (V, E)$ [10]. V is the set of all input ports and output ports of modules, and E is the set of all directed edges corresponding to the signal lines in the data path and relation between an input and an output of each module (we call the latter edge inside edge). We apply *breadth first search* (BFS) with respect to the number of registers to the port graph. From the result of the search, we obtain trees that contain the information of control paths with the minimum sequential depth from PIs to registers. The search ends when all the registers become reachable. In Refs. 6) and 10), to search control paths they also make use of BFS. In this paper,

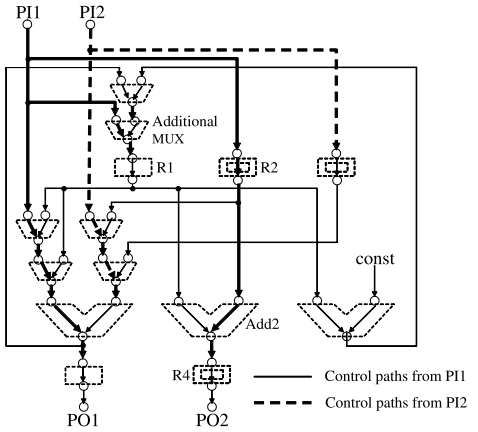

Fig. 5 The port graph and candidates of control paths for LWF.

we add a new condition for search which takes advantage of the feature of SPC two-pattern testability. Considering the conditions of Theorem 1, it is desirable that there exists a hold register on a control path. Therefore we choose a path starting at a hold register if there are some paths that can be chosen at the same sequential depth. **Figure 5** shows the port graph for LWF and candidates of control paths for each register in LWF.

Next we search observation paths with the minimum depth. The search from each register to a PO makes use of observation trees. Observation trees are made by performing the BFS from each PO on the port graph that is generated by reversing the direction of edges, then the BFS prioritize the path on a control tree to share thru function between control paths and observation paths if there is a branch.

Step 4: For one of the RTL paths that are not SPC two-pattern testable, we modify it into SPC two-pattern testable path by adding a hold function to its starting register. In this step, RTL paths that are not CUPs, i.e., the RTL paths need to be tested, and whose pairs of control paths have not yet been determined are dealt with. We first judge RTL paths one by one whether it satisfies one of the conditions of Theorem 1 or not. If the RTL path is SPC two-pattern testable, the pair of control paths generated in Step 3 is determined. However, if the RTL path has no pair of control paths satisfying any one of the five conditions at all, it is sufficient to add a hold function to one of the registers that can be the starting points of off-paths in order to satisfy condition 4 of Theorem 1. Among the registers, a hold function is added to the register with the smallest sequential depth. Consequently, more control paths share the hold function because a set of control paths forms trees. Here we consider testing of RTL path $R2$-$Add2$-$R4$ in Fig. 5. The control paths for $R2$ and $R1$ are $PI1$-$R2$ and $PI1$-Additional MUX-$R1$. The additional MUX was already added between $m5$ and $R1$ in Step 2. Since the pair of control paths cannot satisfy any conditions of Theorem 1, a hold function is added to $R1$. If a hold function is added, go back to Step 3 and make the control trees again for the modified circuit. Then only unsolved RTL paths will be target of Step 4 again.

Step 5: We consider how to realize shorter test application time when there are some choices of off-paths for testing an on-path. We first try to select an off-path having a control path with the minimum sequential depth among them and disjointed from the control path for on-path. If there does not exist such an off-path, that of the minimum depth is selected. We assumed that thru functions are available for all the input ports of all operational modules, however some of them may not be necessary. It is indeed necessary to add a thru function between an input port and an output port, corresponding to inside edges on control or observation paths, of an operational module. To realize a thru function, we first search a support path [10] considering timing conflict. A support path is a path from a PI to an input of an operational module, which can justify a constant. If there does not exist such a path, we add a mask element or a MUX for bypass to realize it.

6. Experimental Results

In this section, we evaluate the effectiveness of the proposed DFT method compared to the previous DFT method for HTPT [6] with regard to area overhead and test application time. The DFT method that guarantees HTPT has similar advantages to enhanced-scan approach and can reduce the area overhead and the test application time. The circuit characteristics of RTL benchmarks used in the experiments are shown in **Table 1**. Paulin, LWF are widely used circuits. RISC and MPEG ☆ are more practical

☆ These circuits were provided for the Joint Research (1997–2001) with Semiconductor Technology Academic Research Center (STARC).

Table 1 Circuit characteristics.

Circuit	BW	#PIs	# POs	# REGs	# RTL path	Area
Paulin	16	2	2	7	29	10,550
LWF	16	2	2	5	19	3,322
RISC	32	1	3	40	10,108	94,302
MPEG	8	7	16	241	651	77,554

Table 2 Results of DFT and test generation.

		Area overhead [%]		TG time [sec]		Test application time [cyc]		# CUPs
Circuit	BW	Proposed	HTPT	Proposed	HTPT	Proposed	HTPT	Proposed
Paulin	8	5.13	11.56	1,956	5,828	785,136	1,645,335	11
	16	3.30	7.43	-	-	-	-	11
LWF	8	7.43	15.25	33	35	38,913	74,792	3
	16	6.38	13.99	551	805	1,638,660	3,162,124	3
RISC	32	0.64	1.99	-	-	$3T_{ALU}$+2	$4T_{ALU}+2$	512
MPEG	8	4.64	9.35	-	-	$186T_M$+2,079	$186T_M$+2,016	0

and larger circuits designed by industry. In this experiment, we used the logic synthesis tool Design Compiler (Synopsys). To generate SPC two-pattern tests, we used the combinational test generation algorithm that supports constraints [11].

Our proposed method guarantees to detect all the faults in a circuit that are detectable in combinational logic blocks separated from the circuit. With regard to robust and non-robust path delay faults the fault coverage of our method is equal to that for HTPT if CUPs are not considered. The CPU time required for our proposed DFT method is as follows. For LWF and Paulin, their CPU times are about 0.1 seconds. For RISC, the CPU time is 3.94 seconds. Note that the CPU time for RISC does not include the time to identify CUPs. Because some control signals depend on status signals, we manually identified only control signals independent of status signals. **Table 2** shows the results of area overhead, test generation time, test application time and the number of CUPs. For all benchmark circuits, area overhead of the proposed DFT method is lower than that of the DFT method for HTPT. Note that there is no reduction of area overhead depending on removing CUPs from target of test in this experiment. The difference between area overhead of the proposed method and that of the previous one become large if there are many registers that are reached from the same PI and at the same sequential depth. For 8 bit Paulin, 8 bit LWF and 16 bit LWF, the test generation times are shorter than that of HTPT. Test generation time with the input constraint for single-port-change tends to become a little longer than that with no constraint. However, we remove CUPs from the target of test generation and the number of target faults is reduced. Hence the test generation time of our proposed method became shorter. For 16 bit Paulin, RISC and MPEG, we cannot evaluate the test generation time because the number of faults is extremely large.

For 8 bit Paulin, 8 bit LWF and 16 bit LWF, the proposed method can reduce test application time to about 50% of that for HTPT. The main reason is that the number of target faults can be reduced by removing CUPs. For Paulin and LWF, we judged eleven and three RTL paths as CUPs, respectively. The judged RTL paths correspond to 50% of the total gate-level paths in 8 bit Paulin and 22% of that in 8 bit and 16 bit LWF. However, even if all the path delay faults on CUPs are removed from the target of test generation, some of them on control paths may be activated during justification of test patterns. Analyzing the percentage of alleviation of over-testing is our future work. The other reason is that the previous method adds extra registers to these circuits. In such case, extra one cycle is necessary for loading data into such a register. To show the influence we also calculated test application time by assuming that RTL paths identified as CUPs are not removed from the target of test. For 8 bit Paulin, 8 bit LWF and 16 bit LWF, the test application times are 1,594,259 cycles, 49,916 cycles and 2,096,465 cycles, respectively. Those results are smaller than that for HTPT data path.

For 16 bit Paulin, RISC and MPEG, it is not practical to test all paths in the data path be-

cause the number of paths is extremely large. Therefore we consider the critical parts that affect the difference between test application time of the proposed method and that of previous one. For 16 bit Paulin, a combinational block composed of two multipliers is the critical part. In the proposed method, many RTL paths through the block are identified as CUPs. However, we cannot estimate the difference between the number of test patterns for the block in the proposed method and that in the previous method. Hence, we cannot perform symbolic analysis. For RISC, an ALU is critical part and its number of tests is denoted as T_{ALU} in the table. The proposed method can reduce 25% compared to the previous one. For MPEG, a sub circuit composed of 64 identical structures of modules is critical. The number of tests is denoted as T_M in the table. For both methods, the test application times are almost the same. For all circuit except for MPEG, since CUPs are identified, over-testing problem is alleviated.

In Section 3.2, we showed that SPC two-pattern tests can test a subset of FS path delay faults. Here, we show the number of FS path delay faults in three simple operational modules that can be tested by SPC two-pattern tests. For an adder and a subtracter, there is no FS path delay fault. All the faults in an adder or a subtracter can be robust or non-robust path delay faults. For an 8 bit multiplier, 947 of the total 49,328 FS path delay faults are tested. From these results, SPC two-pattern tests do not always test all the FS path delay faults of an operational module. On the other hand, if any two-pattern test can be applied, all the FS path delay faults are tested. For every RTL path in an HTPT data path, we can apply any two-pattern test. If it is necessary to test FS path delay faults of such an operational module that is SPC two-pattern test resistant, we can guarantee application of arbitrary two-pattern tests by applying our previous DFT method only for the module.

7. Conclusion

This paper proposed a concept of single-port-change (SPC) two-pattern testability and presented an efficient non-scan DFT method for data path. The proposed method can reduce area overhead and test application time compared to the previous DFT method for hierarchically two-pattern testability without losing the quality of test. Moreover, we alleviated over-testing by removing the control-dependent untestable paths from the consideration of test.

Acknowledgments The authors would like to thank Dr. Tomokazu Yoneda and Dr. Virendra Singh (Nara Institute of Science and Technology) for their valuable discussion and their cooperation on this experiment. This work was supported in part by 21st Century COE (Center of Excellence) Program (Ubiquitous Networked Media Computing) and in part by Japan Society for the Promotion of Science (JSPS) under Grants-in-Aid for Scientific Research B(2) (No.15300018) and for Young Scientists (B) (No.17700062).

References

1) Krstic, A. and Cheng, K.-T.: *Delay Fault Testing for VLSI Circuits*, Kluwer Academic Publishers (1998).
2) Savir, J. and Patel, S.: Scan-based transition test, *IEEE Trans. on CAD*, Vol.12, No.8, pp.1232–1241 (Aug. 1993).
3) Savir, J. and Patel, S.: Broad-side delay test, *IEEE Trans. on CAD*, Vol.13, No.8, pp.1057–1064 (Aug. 1994).
4) Devadas, B.I. and Stong, G.E.: Design for testability: Using scanpath techniques for path-delay test and measurement, *Proc. International Test Conf*, pp.365–374 (1991).
5) Murray, B.T. and Hayes, J.P.: Hierarchical test generation using pre computed tests for modules, *IEEE Trans. on Computer Aided Design*, Vol.9, No.6, pp.594–603 (June 1990).
6) Md. Altaf-Ul-Amin, Ohtake, S. and Fujiwara, H.: Design for hierarchical two-pattern testability of data paths, *IEICE Trans. on Information and Systems*, Vol.E85-D, No.6, pp.975–984 (June 2002).
7) Smith, G.L.: Model for Delay Faults Based Upon Paths, *Proc. International Test Conference*, pp.342–349 (Nov. 1985).
8) Gharaybeh, M.A., Bushnell, M.L. and Agrawal, V.D.: Classification and Test Generation for Path-Delay Faults Using Single Stuck-at Fault Tests, *Journal of Electronic Testing: Theory and Applications*, Vol.11, No.1, pp.55–67 (Aug. 1997).
9) Lai, W.-C., Krstic, A. and Cheng, K.-T.: On Testing the Path Delay Faults of a Microprocessor Using its Instruction Set, *Proc. 18th VLSI Test Symp.*, pp.15–20 (2000).
10) Wada, H., Masuzawa, T., Saluja, K.K. and Fujiwara, H.: Design for strong testability of RTL data paths to provide complete fault efficiency, *Proc. Int. Conf. on VLSI Design*, pp.300–305 (2000).

11) Singh, V., Inoue, M., Saluja, K.K. and Fujiwara, H.: Instruction-based delay fault self-testing of processor cores, *Proc. International Conference on VLSI Design 2004*, pp.933–938 (Jan. 2004).

(Received October 21, 2005)
(Accepted April 4, 2006)
(Online version of this article can be found in the IPSJ Digital Courier, Vol.2, pp.338–347.)

Yuki Yoshikawa received the B.E. degree in computer science from Kyoto Institute of Technology, Kyoto, Japan in 2003 and the M.E. degree in information science from Nara Institute of Science and Technology from Nara, Japan in 2005. Presently he is a Ph.D. candidate of the Graduate School of Information Science, Nara Institute of Science and Technology. His research interests are delay test and design for testability. He is a member of IEEE and IEICE.

Satoshi Ohtake received the B.E. degree in computer science from the University of Electro-Communications, Tokyo, Japan, in 1995 and the M.E. and Ph.D. degrees in information science from Nara Institute of Science and Technology, Nara, Japan, in 1997 and 1999, respectively. He was a Research Fellow of the Japan Society for the Promotion of Science from 1998 to 1999. Presently he is an Assistant Professor of the Graduate School of Information Science, Nara Institute of Science and Technology. His research interests are VLSI CAD, design for testability, and test pattern generation. He is a member of IEEE Computer Society and IEICE.

Michiko Inoue received her B.E., M.E, and Ph.D. degrees in computer science from Osaka University in 1987, 1989, and 1995 respectively. She worked at Fujitsu Laboratories Ltd. from 1989 to 1991. She is an associate professor of the Graduate School of Information Science, Nara Institute of Science and Technology (NAIST). Her research interests include distributed algorithms, parallel algorithms, graph theory and design and test of digital systems. She is a member of IEEE, IEICE, and Japanese Society for Artificial Intelligence.

Hideo Fujiwara received the B.E., M.E., and Ph.D. degrees in electronic engineering from Osaka University, Osaka, Japan, in 1969, 1971, and 1974, respectively. He was with Osaka University from 1974 to 1985 and Meiji University from 1985 to 1993, and joined Nara Institute of Science and Technology in 1993. In 1981 he was a Visiting Research Assistant Professor at the University of Waterloo, and in 1984 he was a Visiting Associate Professor at McGill University, Canada. Presently he is a Professor at the Graduate School of Information Science, Nara Institute of Science and Technology, Nara, Japan. His research interests are logic design, digital systems design and test, VLSI CAD and fault tolerant computing, including high-level/logic synthesis for testability, test synthesis, design for testability, built-in self-test, test pattern generation, parallel processing, and computational complexity. He is the author of Logic Testing and Design for Testability (MIT Press, 1985). He received the IECE Young Engineer Award in 1977, IEEE Computer Society Certificate of Appreciation Award in 1991, 2000 and 2001, Okawa Prize for Publication in 1994, IEEE Computer Society Meritorious Service Award in 1996, and IEEE Computer Society Outstanding Contribution Award in 2001. He is an advisory member of IEICE Trans. on Information and Systems and an editor of IEEE Trans. on Computers, J. Electronic Testing, J. Circuits, Systems and Computers, J. VLSI Design and others. Dr. Fujiwara is a fellow of the IEEE, a Golden Core member of the IEEE Computer Society, a fellow of the IEICE.