IEEE the 8th Asian Test Symposium (ATS'99), pp. 5-12, Nov. 1999.

A Method of Test Generation for Weakly Testable Data Paths Using Test Knowledge Extracted from RTL Description

Satoshi Ohtake Michiko Inoue Hideo Fujiwara

Graduate School of Information Science, Nara Institute of Science and Technology
8916-5, Takayama, Ikoma, Nara 630-0101, Japan

E-mail: {*ohtake, kounoe, fujiwara*}@*is.aist-nara.ac.jp*

Abstract *Weak testability[1] is a testability measure for register-transfer level (RTL) data paths. If a data path satisfies weak testability, for each hardware element of the data path, there exist paths from some primary inputs to the element to justify some values on its output and paths from the hardware element to some primary outputs to propagate some values on its output. For a weakly testable data path, a sequential ATPG tool can generate a test sequence with high fault efficiency in short test generation time[1]. In this paper, we introduce a notion called* test knowledge*, use of which by commercial ATPG tools can further decrease the test generation time and increase the fault efficiency in weakly testable data paths. This test knowledge is information which is relevant to structure of weakly testable data paths, and also easy to find. Use of this test knowledge to facilitate the test generation and increase the testability of data path is established in experimental results.*

1. Introduction

Testing of large VLSI circuits is a well-known hard problem. It is necessary to reduce the cost of testing and to enhance the quality of testing. The cost of testing is estimated by test generation time and test application time. The quality of testing is estimated by fault efficiency[1]. For combinational circuits, combinational automatic test pattern generation (ATPG) tools can generally generate tests with high fault efficiency in reasonable time[2]. On the other hand, high fault efficiency can not be achieved in sequential circuits due to enormous time taken by sequential ATPG tools.

To ease the complexity of test generation, design-for-testability (DFT) techniques are followed. The most commonly used DFT techniques for sequential circuits are scan-based approaches[2]. These techniques can make ATPG easier and can improve fault efficiency. However, these techniques have main disadvantage that at-speed testing[3] is impossible. To avoid such the problems of scan techniques, scientists often proposed some non-scan approaches. Moreover, any technique followed at gate level faces the problems arising out of huge number of elements and high complexities of the circuits at that level. Thus test generation and DFT techniques were recently proposed at register-transfer level (RTL). The main advantage of consideration of DFT at RTL is as follows: complexity of DFT at RTL is much smaller than that at logic level because the number of elements under consideration at RTL is much smaller than that at logic level. Furthermore, since testability is considered at RTL and then logic synthesis is applied, we do not need to take consideration into testability at logic level and we can avoid such an undesirable result that the area and performance optimized in logic synthesis are spoiled by applying DFT at logic level.

In this paper, we consider the testing problems for data paths at RTL. In RTL description, a VLSI circuit generally consists of a controller and a data path. The former is represented by a state transition graph and the latter is represented by interconnection of registers and operational modules. A sequential circuit synthesized from an RTL data path preserves the circuit structure of the RTL data path. Some non-scan DFT techniques for RTL data paths which allow at-speed testing were proposed by Dey et al.[4] and Takabatake et al.[1]. The latter is our previous work. The paper defined a testability measure called *weak testability* for RTL data paths and presented a DFT technique which introduced *thru operation* for some operational modules to make a given data path weakly testable. In literature [1], experimental results showed that hardware overhead of the DFT is very low. If a data path satisfies weak testability, for each hardware element of the data path, there exist paths from some primary inputs to the hardware element to justify some values on its output and paths from the hardware element to some primary outputs to propagate some values on its output. In literature [1], experimental results using a common commercial sequential ATPG tool also showed that a sequential circuit synthesized from weakly testable data path offers low test generation time and high fault efficiency in comparison to that from general data path. However, since an ATPG tool does not know justification/propagation paths considered at RTL, it has to search justification/propagation paths at logic level.

In this paper, given a weakly testable data path, we dis-

[1]Fault efficiency is the ratio of the number of faults detected or proved redundant to the total number of faults.

cuss a method of test generation for the weakly testable data path. In case of a given data path does not satisfy weak testability, we can make it weakly testable by the DFT method in [1]. For a weakly testable data path, we propose to provide some pre-information to commercial ATPG tools using which the tools can further decrease test generation and application time with maintaining or achieving high fault efficiency in weakly testable data path. This pre-information is called *test knowledge*. It is basically an early information of behavior or structure of circuits in the design process. Use of such kind of early information was also proposed in [5, 6]. In literature [6], Vishakantaiah et al. proposed a method of test generation using test knowledge extracted from RTL description written in VHDL. In our proposed method, we find test knowledge early from the structure of the data path and using the test knowledge, a test generation algorithm can easily find paths to activate a fault and propagate an error of the fault. Thus we expect the reduction in test generation time by ATPG tools. The presented method has the following advantages:

1. Test generation time can be reduced.
2. Test application time can be reduced.
3. Fault efficiency can be improved.
4. It is applicable to commercial sequential ATPG tools (need not modify your ATPG tools).

This paper is organized as follows: Section 2 gives the definition of weak testability and shows properties of weakly testable data paths. In Section 3, we define test knowledge and propose a method of test generation. Section 4 presents some heuristics to estimate the effectiveness of the test knowledge. Section 5 shows experimental results of our method.

2. Preliminaries

In RTL description, a VLSI circuit generally consists of a controller and a data path. In this paper, we consider only data paths. Inputs of a data path of a VLSI can be classified as follows: *primary inputs* which are primary inputs of the VLSI and *control inputs* which are from a controller of the VLSI. Similarly, outputs of the data path of the VLSI can be classified as follows: *primary outputs* which are primary outputs of the VLSI and *status outputs* which are to the controller of the VLSI. In this paper, we assume that values on control inputs of a data path of a VLSI can be controlled directly from outside of the VLSI and values on status outputs of the data path can be observed directly from outside of the VLSI.

A data path consists of *hardware elements* and *lines*, where a hardware element is a primary input, a primary output, a control input, a status output, a register, a multiplexor, or an operational module, and a line connects two hardware elements with some bit width. Inputs of hardware elements of a data path can be classified into *data inputs* and *control inputs*. Data inputs of a hardware element are connected directly/indirectly from primary inputs of the data path. Control inputs of a hardware element are connected directly from control input of the data path. Examples of control inputs of hardware elements are load enable signals of registers, selection signals of multiplexers and selection signals of functions of operational modules. Similarly, outputs of hardware elements of a data path can be classified into *data outputs* and *status outputs*. Data outputs of a hardware element are connected directly/indirectly to primary outputs of the data path. Status outputs of a hardware element are connected directly to status outputs of the data path.

In this section, we introduce testability measure for data paths *weak testability* and discuss the properties of data paths which satisfy weak testability.

2.1. Weak Testability

Definition 1 *Thru operation* [1]
Let M be an operational module and F_M be a set of functions which M provides. An operational module M is *thru module* if the following holds.

$$\exists f_{th} \in F_M, \exists i, f_{th}(X_1, \cdots, X_i, \cdots, X_n) = X_i.$$

The above operation is called *thru operation on X_i* and the input X_i is called a *thru input*. □

If an input X is a thru input, X is denoted by $\hat{X}$. Let IN_H denote a set of inputs of an hardware element H. Let H_1 and H_2 be hardware elements, and X be an input of a hardware element. Let $H_1 \rightarrow X$ (resp. $H_1 \rightarrow H_2$) means that there is a line connecting the output of H_1 and X (resp. some input of H_2).

In the rest of this paper, we consider that a multiplexer is an operational module whose every input is thru input.

Here, we introduce weak controllability/observability of a hardware element. Intuitively, weak controllability (resp. observability) of a hardware element H means that *some* value (not necessarily *any*) on the output of H can be justified (resp. propagated) from primary inputs (resp. outputs). Weak controllability is defined recursively from the primary inputs; a hardware element is weakly controllable, if all its inputs or at least one of its thru inputs are weakly controllable.

Definition 2 *Weak controllability* [1]
A set of weakly controllable hardware elements is the minimum set H_{wc} satisfying the following conditions.

1) For any primary input $I, I \in H_{wc}$.
2) For any register $H, \exists H_{wc} \in H_{wc}[H_{wc} \rightarrow H] \Rightarrow H \in H_{wc}$.
3) For any operational module M,

$$\forall X \in IN_M \; [\exists H_{wc} \in H_{wc} \; [H_{wc} \rightarrow X]] \vee$$
$$\exists \hat{X} \in IN_M \; [\exists H_{wc} \in H_{wc} \; [H_{wc} \rightarrow \hat{X}]]$$
$$\Rightarrow M \in H_{wc}.$$ □

It can be considered that some value of some input of a hardware element is propagated (in a weak sense) to its output either if the input is thru input or if all other inputs are weakly controllable. Weak observability is defined recursively from the primary outputs; a hardware element is weakly observable, if its output values can be propagated (in a weak sense) to an output of some weakly observable hardware element.

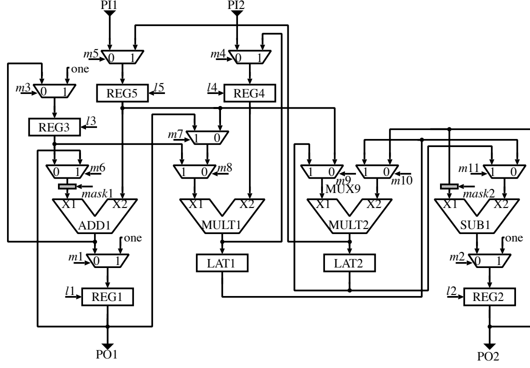

Figure 1: A weakly testable data path (PAULIN).

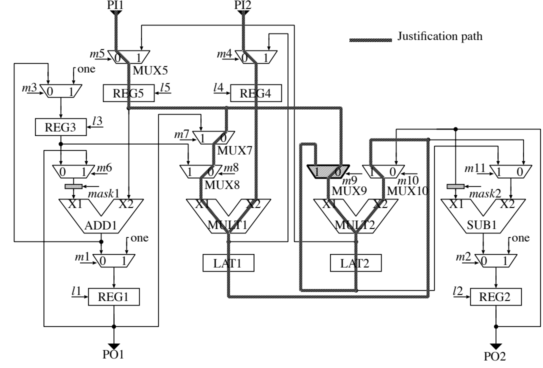

Figure 2: A set of weakly controllable paths of MUX9.

Definition 3 *Weak observability* [1]
A set of weakly observable hardware elements is the minimum set H_{wo} satisfying the following conditions.
1) For any primary output O, $O \in H_{wo}$.
2) For any hardware element H,

$$\begin{aligned} &\exists H_{wo} \in H_{wo}[\\ &\quad \exists X \in IN_{H_{wo}} \; [H \rightarrow X \wedge \forall X' \in IN_{H_{wo}} - \{X\} \\ &\qquad [H \rightarrow X' \vee \exists H_{wc} \in H_{wc} \; [H_{wc} \rightarrow X']]] \vee \\ &\quad \exists \hat{X} \in IN_{H_{wo}} \; [H \rightarrow \hat{X}]] \\ &\Rightarrow H \in H_{wo}. \end{aligned}$$

□

Definition 4 *Weak testability* [1]
A data path *DP* is weakly testable iff all registers in *DP* are weakly controllable and weakly observable. □

It is obvious from the definition that, if all registers in a data path are weakly controllable, all registers are also weakly observable, that is, the data path is weakly testable.

Example 1 *Weakly testable data path*
An example of weakly testable data paths is as shown in Figure 1. It was transformed from the data path of Paulin in [7] by the DFT method of [1]. This includes mask elements, where a mask element realizes a thru operation. In the figure, ADD, MULT, and SUB denote operational modules adder, multiplier, and subtracter, respectively. REG and LAT denote register with load enable signal and without that, respectively. Hardware elements which have control signals from $m1$ to $m11$ denote multiplexers. Similarly, hardware elements which have control signals *mask*1 and *mask*2 denote mask elements of ADD1 and SUB1, respectively.

2.2. Properties of Weakly Testable Data Paths

Let H be a hardware element of a weakly testable data path. We define a set of paths in the weakly testable data path which makes H weakly controllable called a *set of weakly controllable paths* of H and that which makes H weakly observable called a *set of weakly observable paths* of H.

First, we define a *set of weakly justification paths*. We consider that weakly controllability of an input X of a hardware element and weakly controllability of the output of a hardware element H which has connection $H \rightarrow X$ are equivalent.

Definition 5 *A set of weakly justification paths*
Let X be an input of H. Let J be a set of some paths from primary inputs to X and let H_J be a set of hardware elements on J. Let DP_J be a data path which consists of H_J and J. Here, we allow DP_J to have hardware elements which have inputs connected from no hardware element. We assume that these such inputs are not weakly controllable. If J is a minimal set of paths that makes X weakly controllable on DP_J, J is called a *set of weakly justification paths* of X. □

Definition 6 *A set of weakly controllable paths*
Let $X_i (i = 1, 2, \ldots, n)$ be an input of H where n is the number of inputs of H. Let J_i be a set of weakly justification path of X_i. Then $\bigcup_{i=1}^{n} J_i$ is called a *set of weakly controllable paths* of H. □

Definition 7 *A set of weakly observable paths*
A path from the output of H to primary outputs is called a *propagation path* of H. Let P be a propagation path of H. Let m be the number of operational modules on P such that whose input on P is not thru input, and let $PH_i (i = 1, 2, \ldots, m)$ be such an operational module. Let n be the number of inputs of PH_i which are not on P and $X_{ij} (j = 1, 2, \ldots, n)$ be such an input of PH_i. Let J_{ij} be a set of weakly justification paths of X_{ij}. A set of paths P and $\bigcup_{i=1}^{m} \bigcup_{j=1}^{n} J_{ij}$ is called a *set of weakly observable paths* of H. □

Example 2 *A set of weakly controllable path*
We consider the weakly testable data path shown in Figure 1. A set of weakly controllable paths of a multiplexer MUX9 is shown in Figure 2.

Example 3 *A set of weakly observable path*
We consider MUX9 in Figure 1. A set of weakly observable paths of MUX9 is shown in Figure 3.

From Definition 4, for any hardware element H in a weakly testable data path, it is guaranteed that there exists a set of weakly controllable paths of H and a set of weakly observable paths of H. Notice that, in weak testability, control

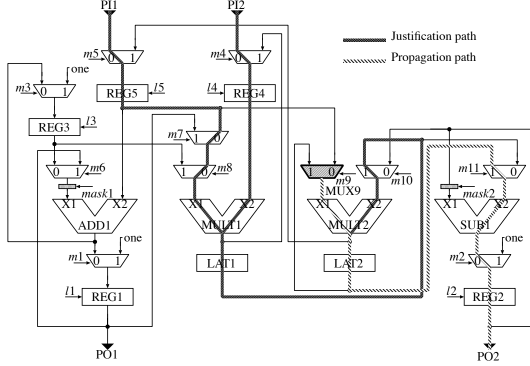

Figure 3: A set of weakly observable paths of MUX9.

timing of load enable signals of registers, selection signals of multiplexers and selection signals of functions of operational modules and dependencies between values on inputs and that on output of a operational modules are not considered. Therefore, it is not guaranteed that any value on any input of H is justified from primary inputs and any value on the output of H is propagated to some primary output (if it can be guaranteed, the data path is *strongly testable*[8]). However, for weakly testable data paths, experimental results in [1] showed that high fault efficiency can be achieved with short test generation time and hardware overhead to make weakly testable is very low.

3. Test Knowledge to Assist ATPG

3.1. Overview of Our Approach

For a sequential circuit synthesized from a weakly testable data path, most faults in the circuit can be activated from primary inputs and errors of these faults can be propagated to primary output using a set of weakly controllable paths and a set of weakly observable paths. However, in normal test generation for these faults, an ATPG tool has to search justification/propagation paths at logic level without using sets of weakly controllable paths and sets of weakly observable paths considered at RTL. If we can inform ATPG tools of existence of these paths called *test knowledge*, we can expect that test generation time and test application time can be decreased. In test generation of a fault of the sequential circuit, to inform an ATPG tool of appropriate test knowledge, we consider to fix some values on control inputs of the sequential circuit during generating a test of the fault.

3.2. Test Knowledge

Inputs of a data path are of two types, primary inputs and control inputs. These control inputs of the data path are connected to control inputs of hardware elements of the data path. If we fix some values on the control inputs, paths which propagate data from primary inputs to primary outputs of the data path can be activated topologically. Applying a value on control inputs of a data path to activate a

Table 1: Coordinate intersection.

$\cap$	0	1	χ	ψ
0	0	ψ	0	ψ
1	ψ	1	1	ψ
χ	0	1	χ	ψ
ψ	ψ	ψ	ψ	ψ

χ = don't care, ψ = can't specify

path from a hardware element to another hardware element is called *forming* the path defined as follows.

Definition 8 *Path formation*
For a hardware element H of a weakly testable data path, let C and O be a set of weakly controllable paths of H and a set of weakly observable paths of H, respectively. Let $p_i (i = 1, 2, \ldots, n)$ be a path in $C \cup O$ where n is the number of paths in $C \cup O$. Let $V_i = [c_{i1}, c_{i2}, \ldots, c_{im}]$ be a vector of values of control inputs of the data path where m is the number of control inputs of the data path. If c_{ij} is a value of a selection signal for some thru input which exists on p_i, c_{ij} is the value ('0' or '1') to enable the thru operation. Otherwise, c_{ij} is a don't care 'χ'. The vector V_i is called an *activating vector* of p_i. Then applying the activating vector V_i to the control inputs of the data path is called *forming* the path p_i. □

Example 4 *An activating vector*
In Figure 2, we consider a path p as follows: PI1 $\rightarrow$ MUX5 $\rightarrow$ REG5 $\rightarrow$ MUX7 $\rightarrow$ MUX8 $\rightarrow$ MULT1 $\rightarrow$ LAT1 $\rightarrow$ MUX10 $\rightarrow$ LAT2 $\rightarrow$ MUX9. The activating vector of p is $[m1, m2, m3, m4, m5, m6, m7, m8, m9, m10, m11, l1, l2, l3, l4, l5, mask1, mask2] = [\chi, \chi, \chi, \chi, 0, \chi, 0, 0, \chi, 1, \chi, \chi, \chi, \chi, \chi, \chi, \chi, \chi]$.

For each hardware element H in a weakly testable data path, there exist a set of weakly controllable paths C of H and a set of weakly observable paths O of H as mentioned above. However, some paths in $C \cup O$ may not be formed simultaneously due to the *conflict of control values*. First, we define *intersection of activating vectors* as follows.

Definition 9 *Intersection of activating vectors*
Let H be a hardware element of a weakly testable data path and let C and O be a set of weakly controllable paths of H and a set of weakly observable paths of H, respectively. Let p_i and p_j be two paths in $C \cup O$ and let V_i and V_j be an activating vector of p_i and that of p_j, respectively. The intersection $V_i \cap V_j$ formed from the respective coordinate intersections in Table 1 is said to be the *intersection of activating vectors* V_i and V_j. □

Notice that, in Table 1, ψ denote that we can not specify a value on the coordinate.

Then conflict of control values are defined as follows.

Definition 10 *Conflict of control values*
Let H be a hardware element of a weakly testable data path and let C and O be a set of weakly controllable paths of H and a set of weakly observable paths of H, respectively. Let p_i and p_j be two paths in $C \cup O$ and let $V_i = [c_{i1}, c_{i2}, \ldots, c_{im}]$ and $V_j = [c_{j1}, c_{j2}, \ldots, c_{jm}]$ be an activating vector of p_i and that of p_j, respectively, where m is the number of control inputs of the data path. If a coordinate intersection $c_{ik} \cap c_{jk} (k = 1, 2, \ldots, m)$ is ψ, we say that a *conflict of control values* occurs on k-th coordinate of V_i and V_j. □

Therefore, in general, for a test generation of a fault of H, we can not fix some values on control inputs which is required to form paths in $C \cup O$ during generating tests of the fault. We define a *test knowledge* of H giving consideration to this point as follows.

Definition 11 *Test knowledge*
For a hardware element H in a weakly testable data path, let C be a set of weakly controllable paths of H and let O be a set of weakly observable paths of H. Let $p_i(i = 1,2,\ldots,n)$ be a path in $C \cup O$ and let $V_i(i = 1,2,\ldots,m)$ be an activating vector of p_i where n and m be the number of paths in $C \cup O$ and that of control inputs of the data path, respectively. Let K is a vector of values of control inputs of the data path such that $\bigcap_{i=1}^{m} V_i$. Such K is said to be a *test knowledge* of H with respect to C and O. □

In general, a test knowledge of H with respect to C and O contains '0', '1', 'χ', and 'ψ'. In our test generation method, for faults in H, we fix each value of a coordinate which is '0' or '1' of the test knowledge on control inputs of the weakly testable data path during generating tests of these faults.

Example 5 *Test knowledge*
We consider a hardware element MUX9 in Figure 1. We suppose that a set of weakly controllable paths and a set of weakly observable paths of MUX9 are selected as Figures 2 and 3, respectively. In this case, a test knowledge of MUX9 is as follows: $[m1, m2, m3, m4, m5, m6, m7, m8, m9, m10, m11, l1, l2, l3, l4, l5, mask1, mask2] = [\chi, 0, \chi, 0, 0, \chi, 0, 0, \psi, 1, 1, \chi, \chi, \chi, \chi, \chi, \chi, 1]$.

For each hardware element H of a weakly testable data path, since there exists one or more sets of weakly controllable paths of H and two or more sets of weakly observable paths of H, we can derive two or more test knowledges of H with respect to a pair of a set weakly controllable paths and a set of weakly observable paths. It is conceivable that selection of a test knowledge of H affects test generation time and test application time of faults in H and quality of tests of these faults. Therefore, we propose some heuristic measures to estimate test knowledge in Section 4 and show effectiveness of these heuristic measures by experiments in Section 5.

3.3. A Method of Test Generation

In our method, we assume that a weakly testable data path and its synthesized circuit are given. If only a data path which is not weakly testable is given, we can make it weakly testable by the DFT method of [1]. If only a data path described in behavioral level, we also make it weakly testable data path using the method of high-level synthesis for testability proposed in [9]. In the case of synthesized circuit is not given, we can derive it by synthesizing the given weakly testable data path using logic synthesis tool. We assume that there exists a one-to-one mapping between a hardware element of the given data path and subcircuit of its synthesized circuit and such mapping is known. The method consists of the following steps.

Step 1 *Extract test knowledge*
For each hardware element H of a weakly testable data path, first, we extract a set of weakly controllable paths C of H and a set of weakly observable paths O of H. Then we derive a test knowledge of H with respect to C and O.

Step 2 *Make fault lists*
For each hardware element H of the weakly testable data path, we make a fault list which include a subset of faults of the subcircuit corresponding to H in a given synthesized circuit. Notice that, for some faults, since time required to generate tests of these faults or to identify that these faults are redundant may be long, we do not deal with some faults in this step. These faults are considered at the last step of this method. Details of this is discussed in Section 4.

Step 3 *Generate tests using test knowledge*
For each hardware element H of the weakly testable data path, we generate tests of faults listed in the fault list corresponding to H for the given synthesized circuit using the test knowledge of H. We iterate the following steps till when trial of test generation for all faults is finished.

Step 3. 1 For faults listed in a fault list which is never tried to generate tests, we try to generate tests using test knowledge corresponding to the fault list. The test knowledge is informed to a sequential ATPG tool by fixing control values of the test knowledge to some control inputs during test generation of these faults.

Step 3. 2 We perform fault simulation for faults in remaining fault lists which are not tried to generate tests using test sequences generated at the previous step. We except faults detected at this fault simulation from remaining fault list.

Step 4 *Generate tests without test knowledge*
For each fault which could not be generated tests at previous step and faults which are not included at Step 2, we try to generate tests again without using test knowledge.

Notice that, in this method, we restrict search space of sequential ATPG by informing test knowledge to sequential ATPG tools. It is generally conceivable that faults which could not be generated tests at the third step contain some testable faults. Therefore, we need to perform the forth step.

4. Estimating Effectiveness of Test Knowledge

In general, for each hardware element H of a weakly testable data path, there exist two or more test knowledges of H as mentioned above. In this section, to select an effective test knowledge, we propose five heuristic measures to estimate effectiveness of these test knowledges. We discuss these heuristic measures in order of effectiveness which is obtained from our preliminary experiments. In our preliminary experiments and proper experiments, we used a logic synthesis tool AutoLogicII (MentorGraphics) with sample libraries of MentorGraphics and a sequential ATPG tool TestGen (SunriseTestSystems)[10] on Ultra2 (SunMicrosystems) workstation.

Heuristic Measure 1 *Thru operation of operational module under test generation*
Weak testability of data paths is defined for fault free data

Table 2: Faults disable thru operation.

	TGt	%FC	%FE	TAt
Try	361.84	83.33	86.98	46
Don't try	1.16	83.33	100.00	46

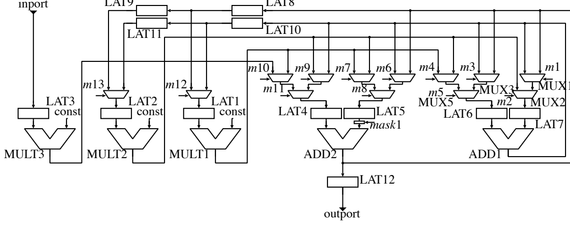

Figure 4: A weakly testable data path (4IIR).

paths. Suppose H is a hardware element in a weakly testable data path under test generation with a test knowledge with respect to a set of controllable paths C, where H has a thru input X and C includes a path to some other input X' of H via the input X. We call such a path a *loop*. We consider that, in test generation of H under such test knowledge, we have to justify values on X' by way of H itself and some fault f in H may cause to fail in such justification. In this case, it is conceivable that time required to generate test of f or to identify that f is redundant becomes long. Therefore, if there exists a test knowledge which makes no loop, we select the test knowledge. Otherwise, we do not try to generate tests for these faults at the third test generation step of our method and try them at the forth step.

Preliminary Experiment 1 *An experimental basis of Heuristic Measure 1*
Here, we consider generating tests for faults in a mask element of ADD1 of the weakly testable data path shown in Figure 1. In this case, the thru operation is needed to justify values on input X1 of ADD1. However, if there exists a fault f of the mask element which disables the thru operation, we can not justify values on the input X1. Table 2 shows test generation results of the third step of our method for the mask element of ADD1. In the table, rows "Try" and "Don't try" denote that all faults in the mask element were tried and all faults except for faults which disable the thru operation in the mask element, respectively. Notice that, in row "Don't try", we assume that faults which are not tried are untestable faults. Columns "TGt", "%FC", "%FE", and "TAt" denote test generation time in second, fault coverage, fault efficiency, and the length of a test generated, respectively. From this results, it is conceivable that time required to generate a test of f or to identify that f is redundant is long. Therefore, we had better to ignore faults which disable the thru operation at the third step.

Heuristic Measure 2 *Reconvergent fanout*
A pair of paths p_1 and p_2 which are from a hardware element H_1 to the other hardware element H_2 such that all registers on these paths do not have load enable signals and p_1 and p_2 have the same number of registers is said to be a *same depth reconvergent fanout*. It is conceivable that a test knowledge makes some same depth reconvergent fanout, the number of values which can be justified from primary inputs on the output of the reconvergent element becomes small. Therefore, if there exists a test knowledge which does not make reconvergent fanout, we select such a test knowledge.

Table 3: Reconvergent fanout.

	TGt	%FC	%FE	TAt
Exists	0.28	98.96	100.00	30
Doesn't exist	0.19	100.00	100.00	42

Preliminary Experiment 2 *An experimental basis of Heuristic Measure 2*
Here, we consider test generation for faults in ADD1 in weakly testable data path shown in Figure 4. The weakly testable data path was transformed from the 4th order IIR cascade filter in [1] by the DFT method of [1]. Table 3 shows test generation results of ADD1. Rows "Exists" and "Doesn't exist" denote test generation result using test knowledge which forms a same depth reconvergent fanout as follows: ADD2 $\rightarrow$ MUX1 $\rightarrow$ MUX2 $\rightarrow$ LAT7 $\rightarrow$ ADD1 and ADD2 $\rightarrow$ MUX3 $\rightarrow$ MUX5 $\rightarrow$ LAT6 $\rightarrow$ ADD1, and that using test knowledge which does not form same depth reconvergent fanouts. The lower result is better than upper one. Therefore, if there exists a test knowledge which does not form same depth reconvergent fanouts, we had better to use it.

Heuristic Measure 3 *Function of operational modules*
Since data are propagated by way of operational modules on paths formed by a test knowledge, some value may not be justified/propagated from/to primary inputs/outputs by their functions. Therefore, for a hardware element H, we select a set of weakly controllable paths of H which can justify the maximum number of values on the inputs of H and a set of weakly observable paths of H which can propagate the maximum number of values on the output of H. And then, we derive a test knowledge from them.

Heuristic Measure 4 *Conflict of control values*
Suppose H is a hardware element of a weakly testable data path and C and O are a set of weakly controllable paths of H and a set of weakly observable paths of H, respectively. If conflicts of control values occur on some control inputs of activating vectors V_i and V_j of paths p_i and p_j in $C \cup O$, respectively, since such values can not be fixed on these control inputs, we can not reduce search space of ATPG corresponding to these such control input. Therefore, to derive a test knowledge which has the largest number of specified values, we select an appropriate pair of a set of weakly controllable paths and a set of weakly observable paths.

Heuristic Measure 5 *Cost of testing*
It is conceivable that test generation time for data paths depends on the number of clock cycles necessary to justify/propagate values of registers from/to primary inputs/outputs. Test application time (the length of tests) is also depends on such the number of clock cycles. In our approach, such the number of clock cycles is depend on test knowledge. Here, we introduce a measure to estimate time required to generate/apply tests called *weak testability cost* as

Table 4: Weak testability cost.

Cost	TGt	%FC	%FE	TAt
9	0.22	100.00	100.00	64
8	0.19	100.00	100.00	42

Table 5: Characteristics of benchmark circuits.

circuit	|bit|	#add	#sub	#mult	#mux	#reg(#L/H)	#th
4IIR	10	2	0	3	13	12(0)	1
Paulin	32	1	1	2	11	7(5)	2

follows.

Definition 12 *Weak testability cost*

Suppose H is a hardware element in a weakly testable data path and C and O are a set of weak controllable paths of H and a set of weak observable paths of H, respectively. Let n and m be the number of paths in C and O, respectively, and let $p_{Ci}(i = 1,2,\ldots,n)$ and $p_{Oj}(j = 1,2,\ldots,m)$ be paths in C and O, respectively. Let $D_{p_{Ci}}$ and $D_{p_{Oj}}$ be the number of registers on p_{Ci} and p_{Oj}, respectively. Let D be $max(\{D_{p_{Ci}}\}) + max(\{D_{p_{Oj}}\})$. Such D is said to be *weak testability cost* of the test knowledge of H with respect to C and O. □

We select a test knowledge which has the smallest weak testability cost.

Preliminary Experiment 3 *An experimental basis of Heuristic Measure 5*

Here, we consider test generation for faults in ADD1 in the weakly testable data path shown in Figure 4. Table 4 shows test generation results of ADD1 using two test knowledges which have different weak testability cost each other. From this results, smaller one is better in test generation time and test application time. Therefore, we had better to use test knowledge which has the smallest weak testability cost.

For a hardware element of a weakly testable data path, we select a test knowledge of the hardware element satisfying these five measures possibly as the above order.

5. Experimental Results

We show experimental results with two benchmarks the 4th order IIR cascade filter[1] (4IIR for short) and the data path of Paulin[7] (PAULIN for short). We made these data paths weakly testable using the DFT method of [1]. Weakly testable 4IIR and weakly testable PAULIN are shown in Figures 4 and 1, respectively.

Circuit characteristics of these weakly testable data paths are shown in Table 5. Columns "|bit|", "#add", "#sub", "#mult", "#mux" "#reg" and "#th" show bit width of data paths, the numbers of adders, subtracters, multipliers, multiplexers, registers, and thru operations which are added by the DFT method, respectively. In column "#reg", a number in parenthesis shows the number of registers which have load enable signals.

Tables 6 and 7 show test generation results of synthesized 4IIR and PAULIN, respectively. We experimented normal test generation (without test knowledge) and our method using test knowledge. In this experiment, to estimate heuristic measure, we extract test knowledge exactly by hand. In our method, we informed TestGen of test knowledge as test generation constraint. To evaluate effectiveness of test knowledge, we did not perform Step 3.2 described in Section 3.3. For both those tables, columns "MODULE", "#Flt", "Normal TG", and "Our TG" show name of hardware element of the data path, the number of faults in each modules, results of normal test generation, and that of our method, respectively. In columns "Normal TG" and "Our TG" of both these tables, columns "TGt (*sec.*)", "%FC", "%FE", and "TAt (cyc.)" show test generation time in second, fault coverage (%), fault efficiency (%), and the number of clock cycles required for test application (the length of tests), respectively. In our results, we used five measures discussed in Section 4 to select test knowledge. Notice that, column "%FE" in "Our TG" shows fault efficiency in case that test knowledges were applied as test generation constraints. In this case, the ATPG tool might identify that some faults are redundant under fixing some values of the test knowledge on control inputs of the synthesized circuits. Therefore, we had to perform Step 4 described in Section 3.3. For both tables, the result of Step 4 is shown in row "(Step 4)". Row "(Step 4)" of column "#Flt" shows the number of faults which had to be retried to generate tests. Row "TOTAL" shows the total amount.

Table 6: Test generation results of 4IIR.

MODULE	Flt	Normal TG (without test knowledge)				Our TG (using test knowledge)			
	(#)	TGt (*sec.*)	FC (%)	FE (%)	TAt (*cyc.*)	TGt (*sec.*)	FC (%)	FE (%)	TAt (*cyc.*)
ADD1	96	0.23	100.00	100.00	36	0.19	100.00	100.00	42
ADD2	96	0.26	100.00	100.00	52	0.16	100.00	100.00	30
MULT1	86	0.36	100.00	100.00	68	0.18	100.00	100.00	36
MULT2	86	0.37	100.00	100.00	86	0.18	100.00	100.00	42
MULT3	86	12.27	100.00	100.00	32	0.09	100.00	100.00	12
MUX1	80	0.83	100.00	100.00	184	0.33	100.00	100.00	100
MUX2	80	0.44	100.00	100.00	96	0.24	100.00	100.00	72
MUX3	80	14.49	98.75	98.75	116	0.43	100.00	100.00	124
MUX4	80	28.26	100.00	100.00	166	0.29	100.00	100.00	76
MUX5	80	0.44	100.00	100.00	88	0.38	100.00	100.00	112
MUX6	80	0.28	100.00	100.00	48	0.32	100.00	100.00	94
MUX7	80	10.58	100.00	100.00	146	0.31	100.00	100.00	92
MUX8	80	0.44	100.00	100.00	100	0.22	100.00	100.00	60
MUX9	80	0.52	100.00	100.00	132	0.21	100.00	100.00	56
MUX10	80	135.45	87.50	87.50	184	0.21	87.50	100.00	60
MUX11	80	142.97	87.50	88.75	110	0.17	87.50	100.00	52
MUX12	80	0.59	100.00	100.00	132	0.34	100.00	100.00	108
MUX13	80	0.68	100.00	100.00	156	0.36	100.00	100.00	122
LAT1	60	0.24	66.67	100.00	48	0.20	66.67	100.00	36
LAT2	60	0.23	66.67	100.00	56	0.17	66.67	100.00	42
LAT3	60	2.18	66.67	100.00	20	0.11	66.67	100.00	12
LAT4	60	7.10	66.67	100.00	20	0.13	66.67	100.00	18
LAT5	60	0.16	66.67	100.00	24	0.18	66.67	100.00	32
LAT6	60	0.30	66.67	100.00	50	0.18	66.67	100.00	42
LAT7	60	0.26	66.67	100.00	50	0.19	66.67	100.00	28
LAT8	60	0.25	66.67	100.00	48	0.18	66.67	100.00	36
LAT9	60	0.26	66.67	100.00	56	0.20	66.67	100.00	42
LAT10	60	0.48	66.67	100.00	70	0.21	66.67	100.00	60
LAT11	60	0.28	66.67	100.00	52	0.24	66.67	100.00	60
LAT12	60	0.15	66.67	100.00	18	0.12	66.67	100.00	24
MASK1	60	123.07	83.33	83.33	38	0.18	83.33	100.00	38
(Step 4)	(270)	–	–	–	–	371.71	0.00	89.26	0
TOTAL	2270	484.42	88.07	98.67	2482	378.61	88.11	98.72	1760

In the results of 4IIR, for MUX10, MUX11 and MASK1, use of test knowledge can drastically decrease test genera-

Table 7: Test generation results of PAULIN.

MODULE	Flt	Normal TG (without test knowledge)				Our TG (using test knowledge)			
	(#)	TGt (*sec.*)	FC (%)	FE (%)	TAt (*cyc.*)	TGt (*sec.*)	FC (%)	FE (%)	TAt (*cyc.*)
ADD1	316	1.49	100.00	100.00	62	1.10	100.00	100.00	44
MULT1	8260	102.27	100.00	100.00	164	18.30	100.00	100.00	120
MULT2	8260	48.70	99.98	99.98	248	19.84	100.00	100.00	142
SUB1	448	27.36	100.00	100.00	74	1.35	100.00	100.00	38
MUX1	196	0.91	100.00	100.00	38	0.96	100.00	100.00	30
MUX2	196	1.20	100.00	100.00	50	1.28	100.00	100.00	50
MUX3	196	1.63	100.00	100.00	64	1.91	100.00	100.00	76
MUX4	256	16.58	100.00	100.00	118	3.57	100.00	100.00	80
MUX5	256	16.95	100.00	100.00	88	4.64	100.00	100.00	68
MUX6	256	1.67	100.00	100.00	80	86.98	100.00	100.00	122
MUX7	256	58.79	100.00	100.00	121	13.45	100.00	100.00	64
MUX8	256	66.70	100.00	100.00	116	24.48	100.00	100.00	88
MUX9	256	10.99	100.00	100.00	144	2.81	100.00	100.00	56
MUX10	256	16.61	100.00	100.00	74	4.31	100.00	100.00	70
MUX11	256	24.95	100.00	100.00	94	1.65	100.00	100.00	64
REG1	448	0.95	78.57	100.00	42	1.08	78.57	100.00	44
REG2	448	1.31	78.57	100.00	48	1.35	78.57	100.00	50
REG3	448	1.58	78.57	100.00	60	2.01	78.57	100.00	80
REG4	448	59.30	78.12	99.55	88	3.09	78.57	100.00	78
REG5	448	1.81	78.57	100.00	56	1.68	78.57	100.00	40
LAT1	192	2.59	66.67	100.00	38	1.17	66.67	100.00	30
LAT2	192	1.34	66.67	100.00	40	0.95	66.67	100.00	22
MASK1	192	1.23	100.00	100.00	54	1.16	83.33	100.00	46
MASK2	192	7.00	100.00	100.00	70	24.07	83.33	100.00	42
(Step 4)	(672)	–	–	–	–	1.82	9.52	100.00	78
TOTAL	22928	473.91	97.34	99.99	2031	225.01	97.34	100.00	1544

tion time. Furthermore, for MUX3, use of test knowledge can improve not only test generation time but also fault coverage. In the result of our method, test generation time is dominated by Step 4. However, since faults were not detected at Step 4, only 6.90 seconds was needed to reach 88.11% fault coverage which is higher than fault coverage of normal test generation.

In the results of PAULIN, for MULT2 and REG4, our method can improve not only test generation time but also fault coverage. However, for MUX6 and MASK1 and MASK2, test generation time is increased and fault coverage is decreased, respectively. From further analysis of this results, the cause of this is that our method does not consider constant inputs "one" of MUX1 and MUX3 of weakly testable PAULIN. By giving consideration to constant inputs, we can expect that such the problems can be avoided.

Form these results for both circuits, our method could reduce test generation time and test application time for most hardware elements. And also could reduce total test generation time and total test application time. Moreover, our method could improve fault coverage for 4IIR and could enhance fault efficiency for both circuits. Therefore, it is conceivable that test knowledges for most hardware elements are effective and our test generation method is efficient.

6. Conclusion

This paper defined novel test knowledge extracted from RTL description and also proposed a method of test generation for weakly testable data paths using the test knowledge. The method is applicable for commercial sequential ATPG tools. Test knowledge estimated by proposed heuristic measures can reduce search space of sequential ATPG tools effectively. Experimental results showed that the test knowledge and the method are effective: test generation time and test application time can be reduced and fault efficiency is improved.

Our future works are to propose an algorithm to generate the test knowledge from RTL description and to perform experiments for large benchmarks.

Acknowledgments Satoshi Ohtake was under Japan Society for the Promotion of Science (JSPS) research fellowship. This work was supported in part by Semiconductor Technology Academic Research Center (STARC) under the Research Project. Authors would like to thank Toshimitsu Masuzawa of Nara Institute of Science and Technology and Tomoo Inoue of Hiroshima City University for their valuable discussions and Debesh Kumar Das of Jadavpur University, India for his helpful comments.

References

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

[2] H. Fujiwara: *Logic Testing and Design for Testability*, The MIT Press, 1985.

[3] P. C. Maxwell, R. C. Aitken, V. Johansen and I. Chiang: "The effect of different test sets on quality level prediction: when is 80% better than 90%?," in *Proc. of International Test Conference*, pp. 358–364, 1991.

[4] S. Dey and M. Potkonjak: "Non-scan design-for-testability of RT-level data paths," in *Proc. of the International Conference on Computer-Aided Design*, pp. 640–645, 1994.

[5] D. Brahme and J. A. Abraham: "Knowledge based test generation for VLSI circuits," in *Proc. of the International Conference on Computer-Aided Design*, pp. 292–295, 1987.

[6] P. Vishakantaiah, J. Abraham and M. Abadir: "Automatic test knowledge extraction form VHDL (ATKET)," in *Proc. of the 29th Design Automation Conference*, pp. 273–278, 1992.

[7] I. Ghosh, A. Raghunathan and N. K. Jha: "A design for testability technique for RTL circuit using control/data flow extraction," in *Proc. of the International Conference on Computer-Aided Design*, pp. 329–336, 1996.

[8] H. Wada, T. Masuzawa, K. K. Saluja and H. Fujiwara: "A non-scan DFT method for RT level datapaths to provide complete fault efficiency (in Japanese)," Technical report, IEICE, FTS97–44, 1997.

[9] M. Inoue, T. Higashimura, K. Noda, T. Masuzawa and H. Fujiwara: "A high-level synthesis method for weakly testable data paths," in *Proc. of the 7th Asian Test Symposium*, pp. 40–45, 1998.

[10] Sunrise Test Systems, Inc.: *Sunrise Reference Manual Version 2.3*, 1996.