IEEE 2001 Workshop on RTL ATPG & DFT, pp. 128-133, Nov. 2001.

# A design for hierarchical testability for RTL data paths using extended data flow graphs

Shintaro Nagai, Satoshi Ohtake and Hideo Fujiwara
Graduate School of Information Science, Nara Institute of Science and Technology
8916-5, Takayama, Ikoma, Nara 630-0101, Japan
E-mail: {*shinta-n, ohtake, fujiwara*}@is.aist-nara.ac.jp

## Abstract

*This paper proposes a non-scan DFT method for hierarchical testability of a register-transfer level data path using control vector sequences generated by an original controller. In hierarchical test generation, a test plan for each module in the data path is generated. The test plan consists of a control vector sequence that can justify any value to the inputs of the module under test from some primary inputs and can propagate its output value to a primary output. In order to generate a control vector sequence for a test plan from the original controller, we extract an* extended test control data flow graph *from the data path and the controller. In our proposed method, the area overhead for a hierarchically testable data path is smaller than our previous work since the area overhead for the test controller to supply such test plans to the data path is small. Furthermore, our proposed method can achieve 100% fault efficiency and at-speed testing.*

## 1 Introduction

Testing of large VLSI circuits is a hard problem. To ease complexity of testing, design-for-testability(DFT) methods have been proposed. The objectives of a DFT method are the following: (1)low area overhead, (2)short test generation time, (3)short test application time, and (4)high fault efficiency[1].

Recently, test generation and DFT methods for register-transfer level(RTL) data paths have been proposed[1]-[9]. These are based on the hierarchical test generation method in [10]. In the hierarchical test generation, test patterns and test plans for each combinational hardware element in the data path are generated. A test plan is a control vector sequence that can justify any value to the inputs of the combinational hardware element under test from some primary inputs, and that can propagate any value from the output of the combinational hardware element to a primary output. However, there does not always exist a test plan for each combinational hardware element. These methods[6]-[9] guarantee the applicability of tests in a hierarchical function. Among them, [5]-[9] are our previous work.

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

The DFT method introduced in [6] makes a data path strongly testable using hold functions of registers and thru functions of operational modules. This method can achieve 100% fault efficiency, and allows at-speed testing. Furthermore, the test generation time and the test application time is shorter than the full scan design. In this method, test plans can not be applied from the original controller to the data path. In a DFT method introduced in [7](strongly testable design method), some multiplexers and a test controller for providing test plans to the data path are added. However, since the area overhead of the test controller and some multiplexers is large, the total area overhead becomes larger than that of the full scan design.

To reduce the are overhead of the test controller, a new concept for testability of data paths called *fixed-control testability* is introduced in [8]. In [8], a DFT method(fixed-control testable design method) is also proposed to make a data path fixed-control testability using thru functions of operational modules, multiplexers and bypass registers. The fixed-control testable design method has advantages over the strongly testable design method. The total area overhead of this method can be reduced compared to that of strongly testable design method. However, the total area overhead of our previous work is still slightly larger than that of the full scan design.

The DFT method introduced in [9] for data paths using control vector sequences of the original controller guarantees the applicability of hierarchical test generation for each combinational hardware element using hold functions of registers and thru functions of operational modules. Also this DFT method can achieve 100% fault efficiency for the data path by adding functions supplying test plans to the hierarchical testable data path.

Genesis[1]-[3] have been proposed as a DFT method such that a test plan for each operational module can be composed by using of control vector sequences of the original controller. Firstly, Genesis generates a *test control data*

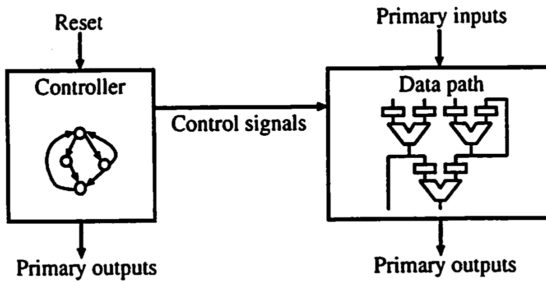


**Figure 1. An RTL controller/data path circuit.**

*flow*(TCDF) graph so as to extract control vector sequences. Secondly, Genesis performes hierarchical testability analysis(HTA) for each operational module using the TCDF graph. In the case that the test plan can not be composed by using a control vector sequence, Genesis guarantees the applicability of hierarchical test generation for each operational module by adding some multiplexers to the data path. The area overhead of Genesis is small, since the test plan for each operational module consists of control vectors of the DFT elements(multiplexers) and a control vector sequence generated by the original controller. However, Genesis can not always achieve 100% fault efficiency, since a test plan for both multiplexers and additional multiplexers of the data path are not generated.

In this paper, we propose a design method for hierarchical testability of data paths using control vector sequences of the original controller without adding any function to it. This method guarantees the applicability of hierarchical test generation for each combinational hardware element using thru functions of operational modules, constant value generators and multiplexers. This method can achieve 100% fault efficiency, and allows at-speed testing. Furthermore, from our experimental results, the area overhead of this method is as small as that of Genesis.

## 2 RTL controllers and data paths

In RTL description, a VLSI circuits generally consists of a controller and a data path as shown in Figure 1. The former is represented by a state transition graph and the latter is represented by hardware elements (e.g. registers, multiplexers and operational modules) and lines.

In this paper, we consider the following restrictions for an RTL circuits.

**A1:** The controller has only a reset signal input.

**A2:** All signal lines in the data path have the same bit width.

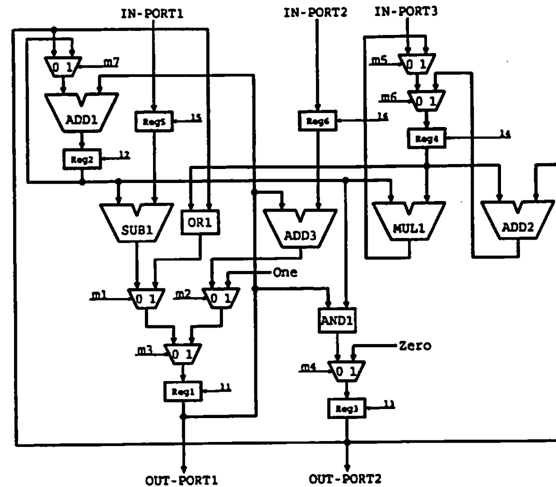


**Figure 2. The data path of *Tseng*.**

| Input reset | State | | Outputs | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PS | NS | l1 | l2 | l3 | l4 | l5 | l6 | m1 | m2 | m3 | m4 | m5 | m6 | m7 |
| 1 | Any | S1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | S1 | S2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | S2 | S3 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | S3 | S4 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | S4 | S5 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | S5 | S1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 3. The state transition table of *Tseng*.**

**A3:** An hardware element has only one or two data inputs and only one data output.

## 3 Hierarchical test generation

Hierarchical test generation is an efficient technique for generating test patterns of very large data paths. In hierarchical test generation, test generation of each combinational hardware element $M$ proceeds as follows:

**Step1:** Test patterns are generated for $M$ using a combinational ATPG tool at logic level.

**Step2:** A test plan is generated for $M$ at RT level. The test plan is a control vector sequence that can justify the test pattern obtained at Step1 to the input of $M$ from some primary input and can propagate its output responses from the output of $M$ to a primary output.

*Strong Testability*[6] is proposed as a characteristic of data paths that guarantees the applicability of the hierarchical test generation.

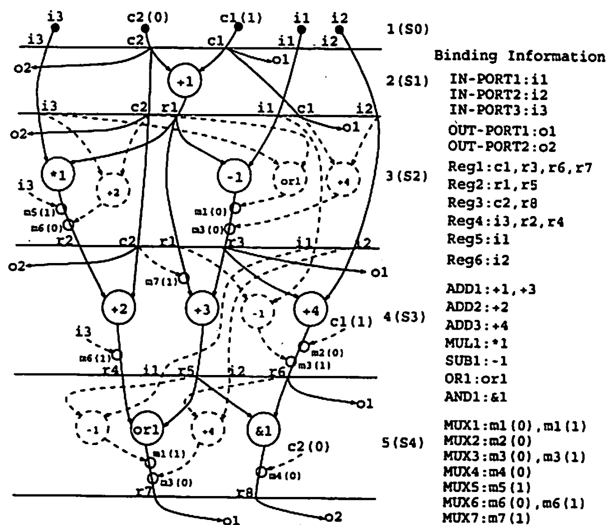


**Figure 4. An ETCDF graph of *Tseng*.**

**Definition 1 (Strong testability)** *A data path is strongly testable iff there exists a test plan for each combinational hardware element M that makes it possible to apply any pattern to M and to observe any response of M.* □

A strongly testable data path has the following advantages:

- Test pattern generation time is short since a combinational ATPG tool can be applied to each combinational hardware element individually.
- 100% fault efficiency can be achieved for the whole data path, since each combinational hardware element $M$ is a circuit of small size and the strong testability guarantees complete controllability and complete observability of $M$.

## 4 An extended test control data flow graph

We introduce a new control data flow graph called *extended test control data flow* (ETCDF) graph defined as follows.

An ETCDF graph $G = (V, E, c)$ of an RTL circuit is a directed graph, where $V$ is a set of nodes representing operations, $E$ is a set of edges representing data transfer flows and $c : V \mapsto N$(natural numbers) defines control steps. Node $v \in V$ corresponds to a primary input, a primary output, an operational module, or a multiplexer of the data path. If edge $e(v_i, v_j) \in E$ satisfies $c(v_i) \neq c(v_j)$, then $e$ corresponds to a register between two data path elements corresponding to $v_i$ and $v_j$ and data signal lines connecting them. Otherwise, $e$ corresponds to a data signal line. A control step denotes counts of clock cycles applied to the controller since the reset signal is applied. Notice that the reset signal is applied at the first control step and after that the reset signal is never applied. For example, the data path and the controller of the RTL circuit Tseng are shown in Figure 2 and 3, respectively and an ETCDF of the circuit is shown in Figure 4. In the ETCDF, large circles denote nodes corresponding to operational modules and small circles denotes nodes corresponding to multiplexers.

Nodes that appear at a control step are classified into two types :*real nodes* and *dummy nodes*. A node of type real is active at the control step, i.e., care values are provided to the hardware element corresponding to the node and the response of the element will be loaded to some register or propagated to some primary output. A node of type dummy is inactive at the control step, i.e., no care values are provided to the hardware element corresponding to the node or the response of the element will not be loaded to any register nor propagated to any primary output.

Edges on the ETCDF are also classified into two types: *real edges* and *dummy edges*. An edge of type real connects two real nodes. An edge of type dummy connects between a real node and a dummy node, or connects between two dummy nodes. For example, in the ETCDF shown in Figure 4, nodes and edges drawn by normal line are real nodes and real egdes, respectively. Also, nodes and edges drawn by dotted line are dummy nodes and dummy edges, respectively.

Notice that, although circuit functions related to dummy nodes do not affect the outside of the circuit, these circuit functions have potentiality to make test plans.

## 5 A design for hierarchical testability for RTL data paths

In this section we propose DFT method for hierarchical testability for RTL data paths. Given an RTL circuit, our proposed DFT method guarantees the applicability of hierarchical test generation of the data path by adding thru functions of operational modules, constant value generators and multiplexers to the data path. In our DFT method, instead of modifying the controller, We add a test register which stores control vectors for DFT elements added to the data path (see Figure 5). In our approach, all hardware elements except for the DFT elements are controlled by the original controller during test application. The DFT elements are controlled from the test register. Our proposed DFT method proceeds as follows:

**Step1:** Given an RTL circuit, for each combinational hardware element in the data path, we analyze whether a test plan which can be composed only of a control vector sequence of the original controller exists.

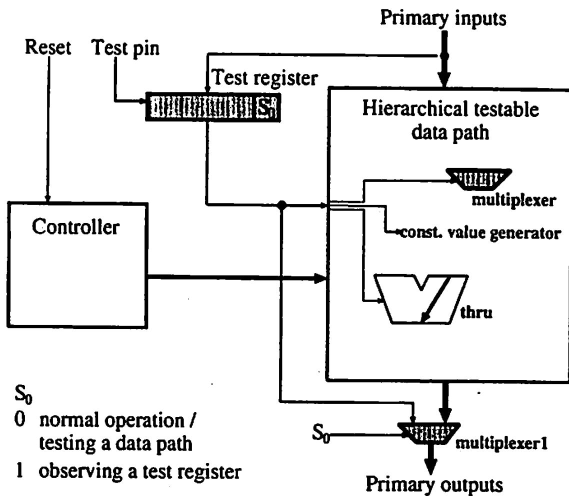


**Figure 5. Architecture of test plan application.**

**Step2:** For each combinational hardware element whose test plan does not exist, a test plan is made by adding DFT elements such that the test plan can be composed of a control vector sequence of the original controller and a vector for those DFT elements.

Notice that, our DFT method also support DFT elements.

In Step1, to extract control vector sequences, we first generate an ETCDF graph from the data path and the controller. Next, we analyze whether a test plan exists for each combinational hardware element by analyzing the controllability for each incoming edge and observability for an outgoing edge of a node corresponding the combinational hardware element under test in the ETCDF graph.

In Step2, we add thru functions of operational modules, constant value generators, or multiplexers to the data path such that the additional DFT elements are controlled using one control vector while testing a hardware element.

The way of application of test plans is shown in Figure 5. In our proposed DFT method, a hardware for controlling the additional DFT elements is a register(test register) since the additional DFT elements are controlled using one control vector while testing a hardware element. Therefore, the test plans can be supplied at low area overhead. To observe the values for controlling the additional DFT elements, we add a multiplexer at primary outputs of the data path. The test pin in Figure 5 is an additional pin for load/hold signal of the test register.

## 6 Experimental results

In this section, we evaluate effectiveness of our proposed method by experiments. Circuit characteristics of RTL benchmark circuits used in the experiments is shown in Table1. In our experiments, we used AutoLogicII (MentorGraphics) as a logic synthesis tool with its sample libraries to synthesize those circuits. In Table 1, the column "Area(*gate*)" denotes the total circuit size before DFT in gates. The columns "Controller" and "Data path" denote the characteristics of controller parts and data path parts, respectively: The columns "#PI", "#PO", and "Area(*gate*)" denote the numbers of primary inputs, primary outputs, and circuit size in gates, respectively. The columns "#State"and "#Control" in "Controller" denote the numbers of states and of control outputs, respectively. The columns "#Reg.", "#Mod." and "#MUX" in "Data path" denote the numbers of registers, operational modules and multiplexeres in data paths, respectively.

The hardware and pin overheads of Genesis[3](Genesis), the DFT method of [7](Strong), the DFT method of [8](Fix) and our proposed method(This) are shown in Table 2. Genesis can not guarantee 100% fault efficiency. In Table 2, the columns "DP","TC", and "MUX" denote the hardware overheads of the data path, the test controller, and the multiplexer added at the primary outputs of the data path, respectively. In our proposed method, the hardware and pin overheads for the whole circuit is as small as Genesis, and is smaller than the DFT methods of both Strong and Fix.

Test generation results are shown in Table 3. In our proposed method, the test generation time is longer than Strong because the number of combinational hardware elements for test generation of our proposed method is larger than Strong. The test application time of our proposed method are longer than the DFT methods of both Strong and Fix. The objective of Strong and Fix is that the test application time is short. Since the main objective of our proposed method is that a test plan for each combinational hardware element is composed of a control vector sequence, the test application time of our proposed method becomes longer than that of Strong and Fix.

Though the test generation and test application time of our proposed method are longer than those of Strong and Fix, we can see that our proposed method is better than both Strong and Fix in the context of hardware overhead from this experimental results. Furthermore, our proposed method can guarantee 100% fault efficiency which Genesis can not, and the hardware overhead of our proposed method is as small as Genesis.

## 7 Conclusion

This paper presented an approach to deign for hierarchical testability for RTL data paths using extended test control data flow graphs. Our proposed method can achieve 100% fault efficiency and allows at-speed testing. Furthermore, the hardware overhead of our proposed method is smaller

**Table 1. Circuit charcteristics.**

| Circuits | Area(*gate*) | Controllers | | | | | Data paths | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | #PI | #PO | #State | #Control | Area(*gate*) | #PI | #PO | #Reg. | #Mod. | #MUX | Area(*gate*) |
| LWF | 1986 | 1 | 0 | 4 | 8 | 58 | 32 | 32 | 5 | 3 | 5 | 1924 |
| JWF | 6875 | 1 | 0 | 8 | 38 | 200 | 80 | 80 | 14 | 3 | 25 | 6672 |
| Paulin | 24966 | 1 | 0 | 6 | 16 | 124 | 64 | 64 | 7 | 4 | 11 | 24834 |
| Tseng | 15033 | 1 | 0 | 5 | 13 | 95 | 96 | 64 | 6 | 7 | 7 | 14930 |

**Table 2. Hardware overheads.**

| Circuits | Area overheads(%) | | | | | | | | | | | | | | | | Pin overheads(#) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Genesis* | | | | Strong | | | | Fix | | | | This | | | | Genesis | Strong | Fix | This |
| | | DP | TC | MUX | | DP | TC | MUX | | DP | TC | MUX | | DP | TC | MUX | | | | |
| LWF | — (0) | — (0) | — (0) | — (0) | 34.5 | 5.7 | 24.3 | 4.5 | 30.5 | 5.6 | 20.4 | 4.5 | 7.3 | 3.8 | 2.3 | 1.2 | 0 | 3 | 4 | 1 |
| JWF | — (0) | — (0) | — (0) | — (0) | 32.8 | 1.6 | 26.4 | 4.8 | 37.7 | 6.0 | 26.5 | 5.2 | 2.1 | 1.1 | 0.7 | 0.4 | 0 | 3 | 4 | 1 |
| Paulin | — (1.3) | — (1.0) | — (0.2) | — (0.1) | 8.0 | 2.0 | 5.4 | 0.6 | 6.1 | 2.3 | 3.1 | 0.7 | 2.5 | 1.6 | 0.6 | 0.3 | 1 | 3 | 4 | 1 |
| Tseng | — (2.0) | — (1.7) | — (0.2) | — (0.1) | 11.5 | 3.6 | 6.8 | 1.1 | 12.8 | 5.8 | 5.8 | 1.2 | 2.4 | 1.5 | 0.6 | 0.3 | 1 | 3 | 4 | 1 |

—( ): DFT results only for operational modules of the data path.
*: Genesis can not achieve 100% fault efficiency.

**Table 3. Test generation results.**

| Circuits | Test generation time(sec.) | | | | Test application time(cyc.) | | | |
|---|---|---|---|---|---|---|---|---|
| | Genesis*1 | Strong*2 | Fix*2 | This | Genesis*1 | Strong | Fix | This |
| LWF | — (0.38) | 0.83 | 0.85 | 0.78 (0.38) | — (78) | 229 | 229 | 196 (78) |
| JWF | — (0.38) | 0.84 | 1.13 | 0.85 (0.67) | — (78) | 720 | 742 | 934 (78) |
| Paulin | — (2.44) | 3.08 | 3.12 | 5.02 (4.51) | — (1520) | 1034 | 1245 | 1922 (1538) |
| Tseng | — (2.83) | 3.65 | 3.98 | 3.78 (3.26) | — (1124) | 1061 | 1157 | 1533 (1274) |

—: In Genesis, multiplexers of the data path are not tested.
*1, ( ): Test generation results only for operational modules of the data path.
*2: The test plan generation time is included to the test generation time.

than that of our previous work. One of our future works is to extend our proposed method to a DFT method that can deal with RTL circuits with a controller that has primary inputs and status inputs.

## Acknowledgments

Authors would like to thank Dr. Michiko Inoue of Nara Institute of Science and Technology for her valuable discussions. This work was sponsored in part by NEDO (New Energy and Industrial Technology Development Organization) through the contract with STARC (Semiconductor Technology Academic Research Center) and supported in part by Foundation of Nara Institute of Science and Technology under the Grant for Activity of Education and Research.

## References

[1] S.Bhatia and N.K.Jha,"Genesis:A behavioral synthesis system for hierarchical testability,"in *Proc. of EDTC*, pp.272-276, Feb. 1993.

[2] I.Ghosh, A.Raghunathan and N.K.Jha,"Design for hierarchical testability of RTL circuits obtained by behavioral synthesis," *IEEE Trans. on CAD*, VOL.16, NO.9, pp.1001-1014, Sep. 1997.

[3] I.Ghosh, A.Raghunathan and N.K.Jha,"A Design for testability technique for RTL circuits using control/data flow extraction," *IEEE Trans. on CAD*, VOL.17, NO.8, pp.706-723, Aug. 1998.

[4] I.Ghosh, N.K.Jha and S.Bhawmik,"A BIST scheme for RTL controller-data paths based on symbolic testability analysis," in *Proc. of DAC*, pp.554-559, June 1998.

[5] S.Ohtake, T.Masuzawa and H.Fujiwara,"A non-scan approach to DFT for Controllers Achieving 100% Fault Efficiency," Journal of Electronic Testing: Theory and Applications, Vol.16, No.5, pp.553-566, Oct. 2000.

[6] H.Wada,K.K.Saluja,T.Masuzawa,H.Fujiwara,"Design for strong testablility of RTL data paths to provide complete fault efficiency," in *Proc. of 13th International Conf. on VLSI Design*, pp.300-305, Jan. 2000.

[7] S.Ohtake, H.Wada, T.Masuzawa and H.Fujiwara,"A non-scan DFT method at register-transfer level to achieve complete fault efficiecy," in *Proc. of ASP-DAC*, pp.599-604, 2000.

[8] S.Ohtake, S.Nagai, H.Wada and H.Fujiwara,"A DFT method for RTL circuits to achieve complete fault efficiency based on fixed-control testability," in *Proc. of ASP-DAC*, pp.331-334, 2001.

[9] K.Suzuki,M.Inoue and H.Fujiwara,"Design for testability of a data path using functions of its original controller," Technical Report of IEICE.,(FTS2000-86), pp.1-8, Feb. 2001 (in Japanese).

[10] B.T. Murray and J.H. Hayes,"Hierarchical test generation using pre computed tests for modules," *IEEE Trans. on CAD*, VOL.9, NO.6, pp.594-603, June 1990.