Design for Testability Using Register-Transfer Level Partial Scan Selection

Akira Motohara Sadami Takeoka Toshinori Hosokawa Mitsuyasu Ohta,
Yuji Takai Michihiro Matsumoto Michiaki Muraoka

Semiconductor Research Center
Matsushita Electric Industrial Co., Ltd.
3-1-1, Yagumo-Nakamachi, Moriguchi, 570 Japan
Tel: +81-6-906-4933
Fax: +81-6-906-3851
motohara@vdrl.src.mei.co.jp

Abstract - An approach to top down design for testability using register-transfer level(RTL) partial scan selection is described. We propose a scan selection technique based on testability analysis for RTL design including data path circuits and control circuits such as state machines. Registers and state machines which make gate level ATPG difficult are identified by the scan selection technique based on RTL testability analysis effectively. Experimental results for actual circuits are also presented.

1. Introduction

Design for testability (DFT) is one of the most important design techniques for large and complex VLSI chips. DFT techniques at gate level are widely used. The most popular one is scan design[1] which reduces the complexity of test generation to a combinational test generation problem. Partial scan design approaches[2-6] which can reduce area and performance penalty are also actively studied and are used in the application areas where requirement of chip cost and/or performance is severe.

Recently, many papers have been published concerning DFT at register transfer (RT) or higher levels aiming at reduction of circuit size and CPU time for DFT and/or automatic test pattern generation (ATPG). Late 1980's, the early research on DFT at register transfer level (RTL) was started. TDES[7] is a knowledge based DFT system at RTL which handles RTL primitives such as registers, ALU's and memories, and DFT schemes such as scan and BIST. TDES can estimate testability, chip area, performance, etc., for a circuit under consideration, with/without assuming some DFT schemes in the circuit. TDES helps designers to make decision concerning DFT solutions at RTL. However, it is difficult to extend the techniques used in TDES to general VLSI chips containing complex control circuits. Another research group proposed an approach to DFT at RTL[8]. One of the most significant features of this method is that behavior of each component is taken into account and a test flow is obtained if a designer specifies the behavior of the circuit. However it takes a long time to specify the behavior of a complex component, in general.

DFT of state machines is another area which has been actively studied. DFT methods for a single finite state machine and a sequential circuit containing several finite state machines are described in [9,10]. They are called logic synthesis for testability (LSFT) since DFT takes place during the logic synthesis phase by eliminating redundancies of the circuit. Disadvantages of LSFT are: 1) large CPU time is needed to handle large state machines; 2) in a practical design, state machines are not always described explicitly. These methods are extended to general RT circuits introducing partitioning and redundancy elimination[11]. This work seems more practical than previous ones since general RT circuits including data path circuits can be handled and translated to fully sequentially testable circuits. However, it still has the following problems: 1) computing time has been reduced in comparison with the previous work, but it is expected that inhibitive CPU time is necessary to do LSFT for large and complex circuits; 2) this approach doesn't work if the circuit contains logic blocks which are manually designed at gate level; and 3) even if a circuit is fully testable, ATPG is still difficult for large circuits.

ADEPT[12] is another type of DFT at RTL which uses VHDL descriptions. ADEPT reads VHDL and extracts data path circuits and control circuits automatically. Then a testability measure called testability sequence range (TSR) is calculated. Based on TSR, ADEPT searches scan insertion points. If scan insertion is required in a register in a data path circuit, ADEPT directly replaces it with a scan register, and if scan insertion is required at a pseudo register placed at the interface between a data path circuit and a control circuit, the control circuit is synthesized and gate level DFT takes place to find a scan insertion point in the control circuit. ADEPT is more practical than the other work in a sense that it is based on a typical ASIC design flow including VHDL, a logic synthesis tool, and a sequential ATPG and that it focuses on sequential depth which makes sequential ATPG difficult.

Unfortunately, ADEPT is a data path oriented approach and is not so effective for circuits containing large control circuits.

PHITS[13,14] aims at DFT in a high-level synthesis environment. PHITS reduces/eliminates sequential depth and large feedback loops by data path allocation and partial scan insertion. PHITS is also data path oriented and no special idea on DFT for control logic circuits is presented.

As summarized above, DFT at RTL or higher levels for sequential circuits including complex state machines remains unsolved.

In this paper, we introduce an approach to DFT at RTL for sequential circuits including state machines. In order to deal with state machines efficiently, we introduced controllability/observability analysis at RTL. DFT techniques using gate level testability measures have been studied and concluded that they are not successful in achieving high fault coverage[15]. However, we started this work for the following reasons,

1) In sequential ATPG procedure, more than two memory elements belonging to a functional units such as registers and state machines are often required to be justified at a time. AT RTL, state machines and registers are explicitly described and recognized as functional units while gate level memory elements are scattered over the circuit.
2) As discussed in [6], if the circuit is modified so that the test sequence which causes state transition between initial and final states of sequential ATPG can be easily obtained, ATPG results can be also improved. Complex state machines can be identified at RTL.

Purposes of this work are:

1) define an RTL circuit model which is efficiently handled by ESDA tools and a DFT method for RTL design including data path circuits and control circuits such as state machines;
2) present a design flow with DFT at both RT and gate levels; and
3) show the experimental results which support effectiveness of our approach.

This paper is organized as follows. In Section 2, the circuit model we deal with is defined. In Section 3, a DFT method at RTL is described. In Section 4, experimental results are reported. Finally, our work is concluded in Section 5.

2. Definition

One major part of this work is to develop an ESDA tool which enables practical functional design and DFT at RTL. We developed a graphical functional design system Bchart. In this section, we present an outline of Bchart system, the design flow using Bchart, and RTL circuit model.

2. 1. Bchart

Fig. 1. shows the software architecture of Bchart. Bchart consists of the following five subsystems.

(1) function diagram editor
(2) diagram consistency checker
(3) interactive simulator
(4) HDL generator
(5) testability analyzer

The function diagram editor is a graphic editor which allows users to do function design entry and modification. The function diagram editor handles three types of hardware models, control circuits represented by state transition diagrams, data path circuits represented by register transfer diagrams, and combinational circuits represented by truth tables or boolean expressions. The following graphic windows dedicated to the hardware models are used in the function diagram editor.

(1) **state machine (SM) window**: state transition diagram,
(2) **register transfer (RT) window**: register transfer diagram,
(3) **boolean expression (BE) window**: boolean expression, and
(4) **condition table (CT) window**: truth table.

The circuit model is discussed in Section 2.3 in detail.

The diagram consistency checker checks static errors in the function diagram such as an undefined variable and an unused variable.

The interactive simulator enables designers to run function simulation on the function diagram. Each variable's status is shown in RT windows and active states are highlighted in SM windows. Simulation proceeds step by step in both forward and reverse time direction. Designers can efficiently verify the function diagram using this powerful debugging tool.

The HDL generator translates the function diagram to an HDL file. Users can specify a language type and a logic syn-

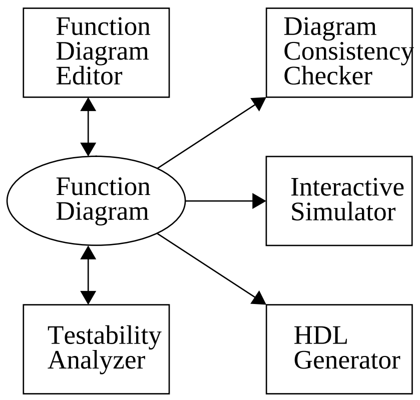


Fig. 1. Software Architecture of Bchart

thesis tool. Currently, VHDL/Verilog HDL and Mentor AutoLogic/Synopsys Design Compiler are supported. The HDL generator generates an HDL file so that the specified logic synthesis tool can correctly handle the HDL file. So the HDL file generated by Bchart is always correctly translated to a gate level net list by the specified logic synthesis tool. This is very useful in comparison with the HDL based design, where designers are often required to modify the HDL file several times in order to obtain a satisfactory synthesis result.

The testability analyzer plays an important role in generating RTL circuits which is made easily testable after logic synthesis. In Section 3, the testability analyzer is introduced in detail.

2. 2. Full Top Down Design Flow

Fig. 2. illustrates a typical design flow of ASIC using Bchart. In Fig. 2, rectangles and ovals represent design tools and design data, respectively. The function diagram edited using the function diagram editor is translated to a HDL file by the HDL generator. Then the HDL file is translated to a net list by a logic synthesis tool. For the net list, ATPG tool generates a test pattern. At the function diagram entry phase, the testability analyzer performs DFT at RTL. The gate level DFT can be used to improve testability further.

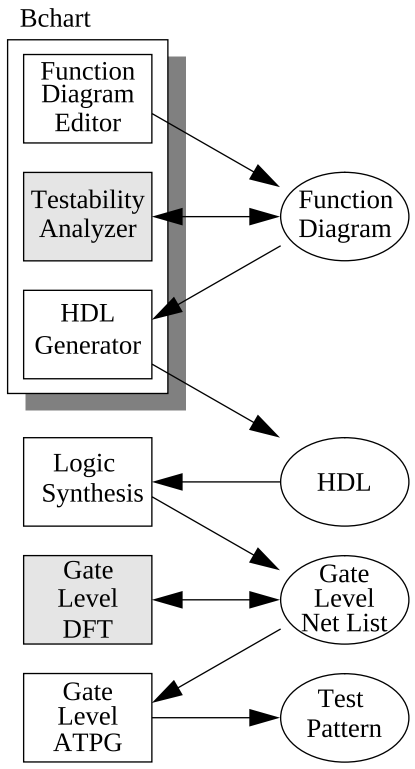


Fig. 2. Top Down Design Flow Using Bchart

2. 3. Circuit Model

In this section, the circuit model we deal with is discussed. The circuit under consideration consists of several subcircuits. Each subcircuit is represented in a graphic window as described in the previous section. Each subcircuit is categorized into the following three types.

(1) state machine
(2) data path
(3) combinational circuit

Signal communication between the subcircuits is carried out using variables. A variable has one or more bits of information. A variable which has one bit of information is defined as a **single bit variable**, and a variable which has two or more bits of information is called a **multiple bit variable**.

2. 3. 1. State Machine

Each state machine has two or more independent states defined by independent single bit variables. At any time, any two states can not be active at the same time in a state machine. A state transition occurs synchronously when a clock signal defined by a single bit variable changes from zero to one. Only one clock signal can be used in a state machine. Destination of state transition from a state is defined by a state transition edge which has information of a destination state. The condition of a state transition is defined by a single bit variable. If one state has two or more destinations, any two state transition conditions can not be true at the same time. If no state transition condition is true, no state transition occurs. The “else” condition is defined as a single bit variable which is true if and only if all of the other state transition conditions of a state are false. A state machine must have only one reset state. A state transition to the reset state occurs asynchronously when a reset signal defined by a single bit variable is true. A state machine handled by Bchart is Moore type and only state signals can be referred by other subcircuits. An example of a state machine described in the SM window is shown in Fig. 3.

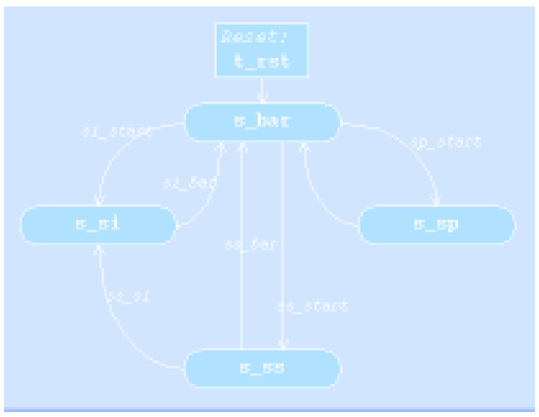


Fig. 3. State Machine

Note that a Mealy type state machine can be described with a Moore type machine and a condition table which holds relation between current states/state transition condition signals and corresponding output signals of the Mealy machine. The condition table is discussed in Section 2.3.3.

2. 3. 2. Data Path

A data path is represented by data transfers between facilities. Each data transfer has information of a source facility, a destination facility, and an optional transfer condition defined by a single bit variable. If a data transfer has no transfer condition, a default transfer condition which is always true is assumed.

An example of a data path subcircuit described in a RT window is shown in Fig. 4.

The following facilities are used in Bchart.

(1) A **register** is a single or multiple bit variable with memory. A register receives data on a fanning-in data transfer synchronously when a clock signal defined by a single bit variable changes from zero to one. If a set or reset condition defined by a single bit variable exists, the register state is set or reset asynchronously when a set or reset signal is true.

(2) A **terminal** is a single or multiple bit variable without memory. A terminal receives data on a fanning-in data transfer asynchronously when a state of the fanning-in data transfer changes.

(3) An **I/O pin** is a port used for signal communication with the outside world of the circuit.

(4) A **connect** is a port used for signal communication with the other data path subcircuits.

(5) A **standard operation unit** such as arithmetic unit (adder, subtracter, multiplier, etc.) or boolean unit (AND, OR, NOT, etc.) can be a facility.

(6) A **memory** such as RAM, ROM can be a facility.

(7) A **submodule** is used in order to describe the subcircuit in hierarchy.

2. 3. 3. Combinational Circuit

A combinational circuit which has one output can be described in a condition table (CT) window or a boolean expression (BE) window. The output of a combinational circuit is a single bit variable and is defined as a label. Labels are used as a control signal such as a state transition condition signal or a data transfer condition signal.

Examples of combinational circuits described in a CT and BE windows are shown in Fig. 5 and 6, respectively.

3. DFT AT RTL

DFT at RTL is performed by the testability analyzer. The testability analyzer has the following functions.

3. 1. Testability Analysis

The testability analyzer calculates 0-controllability(0-CTY), 1-controllability(1-CTY), and observability(OTY) for each bit of each variable on the circuit model discussed in the previous section. The calculating rules for each facility in register transfer subcircuits and combinational subcircuits are similar to the combinational controllability/observability

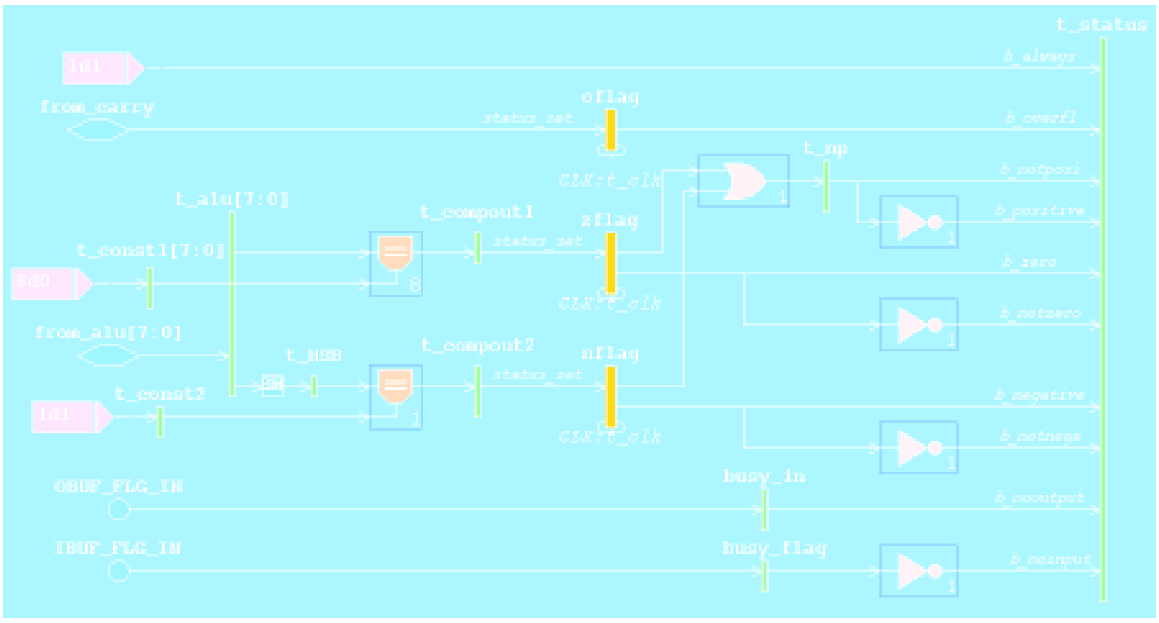
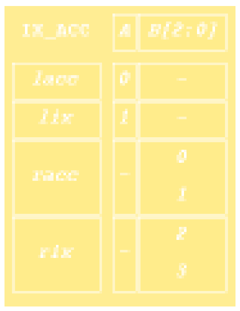
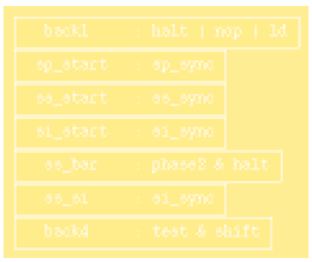


Fig. 4. Data Path



IX_ACC	A	B[2:0]
lacc	0	-
lix	1	-
racc	-	0 1
rix	-	2 3

Fig. 5. Condition Table



```
back1     : halt | nop | ld
sp_start  : sp_sync
ss_start  : ss_sync
si_start  : si_sync
ss_bar    : phase2 & halt
ss_si     : si_sync
back4     : test & shift
```

Fig. 6. Boolean Expression

calculation rule of SCOPE[16]. Instead of adding cost 1 for each gate level primitive in SCOPE, a constant value defined for each facility is added when 0/1-CTY and OTY is calculated. This constant value is called testability penalty. Testability penalty is defined so that sequential facilities such as registers and state machines have large numbers and sequential depth is reflected on 0/1-CTY and OTY.

3. 2. Testability Analysis for State Machines

0/1-CTY and OTY of each bit of each input pin and output pin of a state machine is calculated based on its function. As described in the previous section, each state corresponds to the single bit output pin and the reset signal, the clock signal, and the state transition condition signals are single bit input pins.

0/1-CTY of each state corresponding to an output pin is calculated according to the following rules:

$$C1(RS) = C1(R), \qquad (1)$$
$$C0(S) = C1(R), \qquad (2)$$
$$C1(NS) = C1(CS) + C1(C) + Cck, \qquad (3)$$
$$C0(CS) = C1(CS) + C1(C) + Cck, \qquad (4)$$

where C0(A), C1(A) are 0-CTY, 1-CTY of signal line A, respectively. RS, S, R, C, CS, NS and Cck are defined as follows:

RS: reset state,
S: non-reset state,
R: reset signal,
C: state transition condition,
CS: current state,
NS: next state reached from CS on condition C,
$$Cck = C0(R) + C0(CK) + C1(CK), \qquad (5)$$
where CK is the clock signal.

When 0/1-CTY of at least one of the input pins of a state machine is improved, above rules are estimated and the output pins whose 0/1-CTY are improved are given new values.

OTY calculation rules are as follows:

$$Ob(R) = \min(i)\{C1(Si) + \min\{Ob(Si), Ob(RS)\}\}, \qquad (6)$$
$$Ob(CK) = \min(i, j)\{C1(CSi) + C1(Cij) + \min(j)\{Ob(CSi), Ob(NSij)\} + C0(R)\}, \qquad (7)$$
$$Ob(Cij) = C1(CSi) + \&2(k\ !\ bj)\{C0(Cik)\} + \min\{Ob(NSij), Ob(ESi)\} + Cck, \qquad (8)$$
$$Ob(PSij) = \min(j)\{C1(Cij) + \&2(k\ !\ bi)\{C0(Ckj)\} + Ob(CSj)\} + Cck, \qquad (9)$$

where Ob(A) is OTY of signal line A. S*i*, C*ij*, CS*i*, NS*ij*, ES*ij*, and PS*ij* are defined as follows:

S*i*: non-reset state,
C*ij*: state transition condition,
CS*i*: current state,
NS*ij*: next state reached from CS*i* on condition C*ij*,
ES*i*: next state reached on "else condition" if "else condition" exists, current state otherwise,
PS*ij*: previous state from which state transition to CS*j* occurs on condition C*ij*,

3. 3. Scan Selection Algorithm

Features of our DFT at RTL are as follows:

(1) The unit of scan selection is a **memory element** at RTL, namely a register or a state machine. If a register corresponding to a multiple bit variable is selected, it is replaced with a scan register. In the same way, if a state machine is selected, it is replaced with a state machine consisting of a combinational circuit and a scan register. A scan register is defined as a register whose state can be shifted in and shifted out by scan operation. We use a term **scan element** to represent a scan register or a state machine with a scan register.

(2) Scan selection is performed based on testability analysis. A **memory element cost** of a memory element is defined as a sum of 0-CTY and 1-CTY of output pins and OTY of input pins of the memory element. A **total cost** of a circuit is defined as a sum of memory element cost of all memory elements in the circuit. In our scan selection algorithm, a set of memory elements which effectively reduce the total cost by replacing them with scan elements are selected. Fig. 7 shows the scan selection algorithm written in a pseudo code.

In Fig. 7, ***select_scan()*** is executed with two parameters, namely the number of scan flip flops ***numscan*** and the number of candidates ***numcand*** for each scan selection. This program repeatedly calls testability analysis subprogram ***calc_tm()*** and ***worst_ff()*** which selects ***numcand*** memory elements having the largest memory element cost and makes a memory element set candidate, and selects one memory element in candidate, until scan counts comes to ***numscan***. ***calc_tm()*** returns the total cost of the circuit. ***set_scan()*** and ***unset_scan()*** are used to

```
select_scan(numscan, numcand)
{
    for (i = 0; i < numscan; i += scan_bits) {
        total = calc_tm();
        candidate = worst_ff(numcand);
        best_gain = 0;
        for (each ff in candidate) {
            set_scan(ff);
            temp = calc_tm();
            unset_scan(ff);
            bits = bit_count(ff);
            if ( $\frac{total - temp}{bits}$ > best_gain) {
                best_gain = $\frac{total - temp}{bits}$;
                best_ff = ff;
                scan_bits = bits;
            }
        }
        set_scan(best_ff);
    }
}
```

Fig. 7. Scan Selection Algorithm

assume scan for a memory element and to cancel assumed scan, respectively. ***bit_count()*** counts the number of flip flops in a memory element.

4. Experiments

We did the following experiments:

(1) RTL DFT(RDFT)
(2) Gate level DFT-1(GDFT-1)
(3) Gate level DFT-2(GDFT-2)

Gate level DFT-1 is the method discussed in [6]. Gate level DFT-2 in experiment (3) is the same method as RTL DFT, namely scan flip flops are selected based on gate level controllability and observability. The purpose of experiment (3) is to compare it with (1), and (2).

We performed the above experiments on ISCAS'89 benchmark circuits[17] for gate level DFT and the RTL circuits summarized in Table 1. Cir1 is a 8 bit microprocessor[18]. Cir2 is a bus controller which contains few primary input and output pins and complex state machines and counters in it. Cir3 is an image processing "data path oriented" circuit.

HDL and the design tools used in our experiments are as follows:

HDL: Verilog HDL
ESDA: Bchart

TABLE 1
Profile of RTL Circuits

Circuit		Cir1	Cir2	Cir3
RTL	State Machine	2	2	0
	Primary In	16	3	18
	Primary Out	12	4	7
	Register	24	22	65
Gate Level	Primary In	40	3	40
	Primary Out	45	4	55
	Primitive	560	511	629
	Flip Flop	64	126	236

Logic synthesis: Synopsys Design Compiler
Sequential ATPG: Mint[19]

Table 2 shows the results. In Table 2, "Fault Coverage", "DFT Time", and "ATPG Time" represent fault coverage, CPU time on Sun SPARCstation 20 for scan selection and test generation, respectively. "No Scan" is the ATPG results for the original circuits. "Partial Scan (20%)" shows the above four DFT experimental results where 20% of flip flops are replaced with scan flip flops. Both "numscan" and "numcand" in Fig. 7 are set to the number of final scan flip flops.

ATPG and gate level DFT results for two ISCAS'89 circuits which have many flip flops and are hard to test are used to show the ATPG capability and effectiveness of gate level DFT. The results show that our gate level tools are comparable with the previously published most successful ATPG and DFT tools such as [5] from the point of view of fault coverage. One problem is that our gate level DFT tool uses sequential ATPG to find 1-Hamming distance states and very long CPU time which is comparable to the ATPG time for the original circuits is consumed.

RDFT achieves higher fault coverage than GDFT-1 in much shorter CPU time. In comparison with GDFT-2 where scan flip flops are selected according to the gate level SCOPE values, effectiveness of scan selection is much better. This fact supports that our testability analysis which directly calculates the difficulty of state transition reflects the difficulty of sequential ATPG at gate level. For Cir3, GDFT-2 resulted in a good fault coverage. This is the case where even gate level SCOPE values are good enough in finding scan flip flops since it contains no state machines. In other cases, GDFT-2 seems less practical.

The experimental results can be summarized as follows:

(1) Proposed method(RDFT) offers high fault coverage and short ATPG time.
(2) In comparison with ATPG based gate level DFT(GDFT-1)[6] which achieves highest fault coverage among published work, high fault coverage is obtained with a DFT time which is more than two orders of magnitude shorter.
(3) DFT based on gate level testability measure(GDFT-2) are not very effective.

TABLE 2
EXPERIMENTAL RESULTS

Circuit	Method	No Scan	Partial Scan (20%)		
			RDFT	GDFT-1	GDFT-2
Cir1	Fault Coverage (%)	88.85	92.21	91.42	91.14
	DFT Time (sec)	0	17	11534	4
	ATPG Time (sec)	3373	2077	5169	3428
Cir2	Fault Coverage (%)	18.06	91.05	89.88	28.15
	DFT Time (sec)	0	12	10389	4
	ATPG Time (sec)	1684	223	1436	1443
Cir3	Fault Coverage (%)	88.28	96.56	96.27	96.13
	DFT Time (sec)	0	109	27622	13
	ATPG Time (sec)	4443	272	884	3228
S1423	Fault Coverage (%)	88.05	—	95.78	89.50
	DFT Time (sec)	0		5121	2
	ATPG Time (sec)	8204		890	1378
S5378	Fault Coverage (%)	77.95	—	98.85	81.88
	DFT Time (sec)	0		18242	20
	ATPG Time (sec)	24435		441	1783

5. CONCLUSION

In this paper we described an approach to design for testability at RTL. Major contributions are as follows:

(1) We defined an RTL circuit model which enables efficient description in an ESDA tool and testability analysis which leads to effective partial scan selection for RTL design including data path circuits and control circuits such as state machines.
(2) We introduced a method of partial scan selection at RTL which selects critical registers and state machines based on RTL testability analysis.
(3) We did experiments to verify the effectiveness of our DFT approach.

According to the experimental results, our gate level DFT achieves high fault coverage comparable with the previously published most successful DFT methods, and DFT at RTL resulted in higher fault coverage than gate level DFT at much shorter CPU time.

REFERENCE

[1] M. Abramovici, M. A. Breuer, and A. D. Friedman, “Digital Systems Testing and Testable Design,” Computer Science Press, 1990.
[2] H.-K. T. Ma, S. Devadas, A. R. Newton, and A. Sangiovanni-Vincentelli, “An Incomplete Scan Design Approach to Test Generation for Sequential Machines,” Proc. Int. Test Conf., pp.730-734, Sep. 1988.
[3] K.-T. Cheng and V. D. Agrawal, “A Partial Scan Method for Sequential Circuits with Feedback,” IEEE Trans. Comp., pp.544-548, Apr. 1990.
[4] M. Abramovici, J. J. Kulikowski, and R. K. Roy, “The Best Flip-flops to Scan,” Proc. Int. Test Conf., pp.166-173, Oct. 1991.
[5] V. Chickermane and J. H. Patel, “A Fault Oriented Partial Scan Design Approach,” Proc. Int. Conf., Computer-Aided Design, pp.400-403, Nov. 1991.
[6] S. Takeoka, A. Motohara, T. Hosokawa, and M. Ohta, “Scan Flip-Flop Selection Based on State Transition for Automatic Partial Scan Insertion,” Proc. SASIMI’92, pp.282-291, Apr. 1992.
[7] M. Abadir and M. A. Breuer, “A Knowledge-Based System for Designing Testable VLSI Chips,” IEEE Design & Test of Comp., pp.56-68, Aug. 1985.
[8] M. Crastes de Paulet, M. Karam, and G. Saucier, “Testability Expertise and Test Planning from High-Level Specifications,” Proc. Int. Test Conf., pp.692-699, Aug. 1989.
[9] S. Devadas, H.-K. T. Ma, and A. R. Newton, “A Synthesis and Optimization Procedure for Fully and Easily Testable Sequential Machines,” IEEE Trans. CAD., Vol.8, pp.1100-1107, Oct. 1989.
[10] S. Devadas, H.-K. T. Ma, and A. R. Newton, “Redundancies and Don’t Cares in Sequential Logic Synthesis,” Journal of Electronic Testing: Theory and Applications, Vol.1, pp.15-30, Feb. 1990.
[11] A. Gohsh, S. Devadas, and A. R. Newton, “Sequential Logic Synthesis for Testability using Register-Transfer Level Descriptions,” proc. Int. Test Conf., pp.274-283, Oct. 1990.
[12] V. Chickermane, J. Lee, and J. H. Patel, “Design for Testability Using Architectural Descriptions,” Proc. Int. Test Conf., pp.752-761, Oct., 1992.
[13] T.-C. Lee, N. K. Jha, and W. H. Wolf, “Behavioral Synthesis of Highly Testable Data Paths under the Non-Scan and Partial Scan Environments,” Proc. 30th ACM/IEEE Design Automation Conf., pp.292-297, June 1993.
[14] T.-C. Lee, N. K. Jha, and W. H. Wolf, “A Conditional Resource Sharing Method for Behavioral Synthesis of Highly Testable Data Path,” Proc. Int. Test Conf., pp.744-753, Oct. 1993.
[15] V. Chickermane and J. H. Patel, “An optimization based approach to the partial scan design problem,” Proc. Int. Test Conf., pp.377-386, 1990.
[16] L. H. Goldstein, “Controllability and Observability Analysis of Digital Circuits,” IEEE Trans. Circuits and Systems, Vol. CAS-26, No.9, pp.685-693, Sep. 1979.
[17] F. Brglez, D. Bryan, and K. Kozminski, “Combinational Profile of Sequential Benchmark Circuits,” Proc. Int. Symp. Circuits and Systems, pp. 1929-1934, May 1989.
[18] H. Kanbara, “KUE-CHIP: A Microprocessor for Education of Computer Architecture and LSI Design,” Proc. IFIP Workshop on Design & Test of ASICs, pp.15-18, 1990.
[19] A. Motohara, T. Hosokawa, M. Muraoka, H. Maekawa, K. Kayashima, Y. Shimeki, and S. Shin, “A State Traversal Algorithm Using a State Covariance Matrix,” Proc. 30th ACM/IEEE Design Automation Conf., pp.97-101, June 1993.