J Electron Test (2010) 26:151–164
DOI 10.1007/s10836-009-5135-1

RTL DFT Techniques to Enhance Defect Coverage for Functional Test Sequences

Hongxia Fang · Krishnendu Chakrabarty · Hideo Fujiwara

Received: 30 September 2009 / Accepted: 10 December 2009 / Published online: 6 January 2010
© Springer Science+Business Media, LLC 2009

Abstract Functional test sequences are often used in manufacturing testing to target defects that are not detected by structural test. However, they suffer from low defect coverage since they are mostly derived in practice from existing design-verification test sequences. Therefore, there is a need to increase their effectiveness using design-for-testability (DFT) techniques. We present a DFT method that uses the register-transfer level (RTL) output deviations metric to select observation points for an RTL design and a given functional test sequence. Simulation results for six $ITC'99$ circuits show that the proposed method outperforms two baseline methods for several gate-level coverage metrics, including stuck-at, transition, bridging, and gate-equivalent fault coverage. Moreover, by inserting a small subset of all possible observation points using the proposed method, significant fault coverage increase is obtained for all benchmark circuits.

Keywords DFT · Output deviations · RT-level · Test-point insertion · Unmodeled defects

Responsible Editor: P. Mishra

This research was supported in part by the Semiconductor Research Corporation under Contract no. 1588, and by an Invitational Fellowship from the Japan Society for the Promotion of Science. This paper is based on a preliminary version of an invited paper in *Proceedings of IEEE International High Level Design Validation and Test Workshop, 2009*, and a presentation at *the IEEE Workshop on RTL and High-Level Testing, 2009*.

H. Fang (✉) · K. Chakrabarty
Department of Electrical and Computer Engineering, Duke University, Box 90291, 130 Hudson Hall, Durham, NC 27708, USA
e-mail: hf12@ee.duke.edu

K. Chakrabarty
e-mail: krish@ee.duke.edu

H. Fujiwara
Graduate School of Information Science, Nara Institute of Science and Technology, Kansai Science City, Nara 630-0192, Japan
e-mail: fujiwara@is.naist.jp

1 Introduction

Very deep sub-micron (VDSM) process technologies are leading to increasing defect rates for integrated circuits (ICs) [1, 24]. Since structural test alone is not enough to ensure high defect coverage, functional test is commonly used in industry to target defects that are not detected by structural test [11, 28, 36]. An advantage of functional test is that it avoids overtesting since it is performed in normal functional mode. In contrast, structural test is accompanied by some degree of yield loss [32]. Register transfer (RT)-level fault modeling, design-for-testability (DFT), test generation and test evaluation are therefore of considerable interest [5, 26, 33, 35]. In RT-level design, a circuit's behavior is defined in terms of the flow of signals (or transfer of data) between registers and the logical operations performed on those signals.

A number of methods have been presented in the literature for test generation at RT-level. In [5], the authors proposed test generation based on a genetic algorithm (GA), targeting statement coverage as the quality metric. In [12, 15, 23, 31], the authors used pre-computed test sets for RT-level modules (adders, shifters, etc.) to derive test vectors for the complete design. In [40], the authors presented a spectral method

for generating tests using RT-level faults, which has the potential to detect almost the same number of faults as using gate-level test generation. In [19, 21], the authors proposed a fault-independent test generation method for state-observable finite state machines (FSMs) to increase the defect coverage.

To increase the testability of the complete design and to ease RT-level test generation, various DFT methods at RT-level have also been proposed. The most common methods are based on full-scan or partial scan. However, a scan-based DFT technique leads to long test application time and it is less useful for at-speed testing. On the other hand, non-scan DFT technique [6, 9, 13, 14, 30, 37] offer low test application time and they facilitate at-speed testing. In [6], non-scan DFT techniques are proposed to increase the testability of RT-level designs. In [30], the authors presented a method called orthogonal scan. It uses functional datapath flow for test data, instead of traditional scan-path flow; therefore, it reduces test application time. In [13], a technique was proposed to improve the hierarchical testability of the data path, which can aid hierarchical test generation. In [14], the authors presented a DFT technique for extracting functional control- and data-flow information from RT-level description and illustrated its use in design for hierarchical testability. This method has low overhead and it leads to shorter test generation time, up to 2–4 orders of magnitude less than traditional sequential test generation due to the use of symbolic test generation. In [37], the authors presented a method based on strong testability, which exploits the inherent characteristic of datapaths to guarantee the existence of test plans (sequences of control signals) for each hardware element in the datapath. Compared to the full-scan technique, this method can facilitate at-speed testing and reduce test application time. However, it introduces hardware and delay overhead. To reduce overhead, the authors proposed a linear-depth time-bounded testability-based DFT method in [9]. It ensures the existence of a linear-depth time expansion for any testable fault and experiments showed that it offers lower hardware overhead than the method in [37].

The goal of the above prior work on RTL DFT was to increase testability and ease RT-level test generation. To enhance the effectiveness of given functional test sequences, the focus in prior work was on functional test selection based on various metrics, such as transition input/output (TRIO) [22], toggle coverage [10], validation vector grade (VVG) [34], etc. DFT involving scan-chain insertion at RTL has also been studied [2, 20]. However, the use of RTL DFT to increase the defect coverage of existing functional test sequences for non-scan designs has largely been ignored. The generation of functional test sequences is a particularly challenging problem, since there is insufficient automated tool support and this task has to be accomplished manually or at best in a semi-automated manner. Therefore, functional test sequences for manufacturing test are often derived from design-verification test sequences [16, 18, 27] in practice. It is impractical to apply all such long verification sequences during time-constrained manufacturing testing. Therefore, shorter subsequences must be used for testing, and this leads to the problem of inadequate defect coverage. Therefore, we focus on RTL DFT to increase the effectiveness of these existing test sequences for non-scan designs.

In this paper, we address the problem of improving the defect coverage of given functional test sequences for an RT-level design. The proposed method adopts the RT-level deviation metric from [8] to select the most appropriate observation test points. The deviation metric at the gate-level has been used in [38] to select effective test patterns from a large repository of n-detect test patterns. It has also been used in [39] to select appropriate LFSR seeds for reseeding-based test compression. The deviation metric at RT-level has been defined and used in [8] for grading functional test sequences.

Simulation results for six $ITC'99$ circuits show that the proposed method outperforms two baseline methods for several gate-level coverage metrics, including stuck-at, transition, bridging, and gate-equivalent fault coverage. Moreover, by inserting a small subset of all possible observation points using the proposed method, significant fault coverage increase is obtained for all benchmark circuits. Since functional test sequences are used to target unmodeled defects, especially when they are used in conjunction with structural testing for modeled faults, we evaluate test effectiveness by multiple and different fault models.

The remainder of this paper is organized as follows. Section 2 presents the problem formulation. Section 3 describes the RT-level output-deviations metric and its prior application to functional test grading. Section 4 presents the proposed observation-point selection method based on RT-level deviations. The design of experiments and experimental results are reported in Section 5. Section 6 concludes the paper and outlines directions for future work.

2 Problem Formulation

We first formulate the problem being tackled in this paper.

Given:

- The RT-level description for a design and a functional test sequence $\mathcal{S}$;
- A practical upper limit n on the number of observation points that can be added.

Goal: Determine the best set of n observation points that maximizes the effectiveness of the functional test sequence $\mathcal{S}$.

Functional test sequences can be instruction-based for processors or application-based for application specific integrated circuit (ASIC) cores such as an MPEG decoder. They can be in the format of high-level instructions or commands, or in the format of binary bit streams.

To increase testability, we can insert an observation point for each register output. We can obtain the highest defect coverage by inserting the maximum number of observation points. However, it is impractical to do so due to the associated hardware and timing overhead. In fact, the number of observation points that can be added is limited in practice. For a given upper limit n, the challenge is to determine the best set of n observation points such that we can maximizes the defect coverage of the given functional test sequence. Our main premise is that RT-level output deviation can be used as a metric to guide observation-point selection.

3 Output Deviations at RT-level: Preliminaries

The RT-level output deviations metric has been defined and used in [8] to grade functional test sequences. The RT-level output deviations metric was used as a surrogate coverage metric to grade functional test sequences and test sequences with higher deviation values were found to provide higher defect coverage.

Before describing RTL output deviations, we introduce basic concepts. The first concept is the transition count (TC) of registers. Typically, there is dataflow between registers when an instruction is executed and the dataflow affects the values of registers. For any given bit of a register, if the dataflow causes a change from 0 to 1, it records that there is a $0 \rightarrow 1$ transition. Similarly, if the dataflow causes a change from 1 to 0, it records that there is a $1 \rightarrow 0$ transition. If the dataflow makes no change to this bit of the register, it records that there is a $0 \rightarrow 0$ transition or a $1 \rightarrow 1$ transition, as the case may be.

After a transition occurs, the value of the bit of a register can be correct or faulty (thus an error may be produced). With any transition of a register bit, a "confidence level" (CL) parameter is associated, which represents the probability that the correct transition occurs. The CL is not associated with the test sequence; rather, it provides a probabilistic measure of the correct operation of instructions at the RT-level. It can be estimated from low-level failure data or it can be provided by the designer based on the types of faults of interest. Low CL values can be assigned to registers that are to be especially targeted by functional test sequences for error observation and propagation. Experiments showed that small differences in the CL values have little impact on the effectiveness of grading functional test sequences using RT-level output deviations.

Without loss of generality, the CL values for $0 \rightarrow 0$ and $1 \rightarrow 1$ are assumed to be higher than that for the transitions $0 \rightarrow 1$ and $1 \rightarrow 0$ for a register bit. The CL values for $0 \rightarrow 1$ and $1 \rightarrow 0$ are assumed to be identical. The CL for a register can be described as a 4-tuple, e.g., $< 0.998, 0.995, 0.995, 0.998 >$, where the elements in the tuple correspond to the transitions $0 \rightarrow 0$, $0 \rightarrow 1$, $1 \rightarrow 0$, and $1 \rightarrow 1$, respectively. While different CL values can be used for the various registers in a design, without loss of generality, all registers are assumed to have identical CL values.

When an instruction is executed, there may be several transitions for the bits of a register. Therefore, an error may be manifested after the instruction is executed. The CL for instruction I_i, C_i, is defined as the probability that no error is produced when I_i is executed.

Similarly, since a functional test sequence is composed of several instructions, the CL for a functional test sequence is defined to be the probability that no error is observed when this functional test sequence is executed. For example, suppose a functional test sequence, labeled T_1, is composed of instructions I_1, I_2, ..., I_N, and let C_i be the CL for I_i, as defined above. The CL value for T_1, $C(T_1)$, is defined as:

$$C(T_1) = \prod_{i=1}^{N} C_i. \tag{1}$$

This corresponds to the probability that no error is produced when T_1 is executed. Finally, the deviation for functional test sequence T_1, $\triangle(T_1)$, is defined as $1 - C(T_1)$, i.e.,

$$\triangle(T_1) = 1 - \prod_{i=1}^{N} C_i. \tag{2}$$

Based on these definitions, output deviations can be calculated at RT-level for functional test sequences for a given design. Three contributors are considered in the calculation of deviations. The first is the TC of registers. Higher the TC for a functional test sequences, the more

likely is it that this functional test sequences will detect defects.

The second contributor is the observability of a register. The TC of a register will have little impact on defect detection if its observability is so low that transitions cannot be propagated to primary outputs. Therefore, each output of registers is assigned an observability value using a SCOAP-like measure [3]. (In SCOAP testability measure, a higher value of the measure for a signal indicates that it might be difficult to control or observe, i.e., it has lower controllability or lower observability.) The observability vector for a design, composed of the observability values of all registers, is used to model the observability of the the design.

We use the Parwan processor [29] as an example to illustrate the calculation of observability vector. The Parwan is an accumulator-based 8-bit processor with a 12-bit address bus. Its architectural block diagram is shown in Fig. 1a. Its dataflow diagram, representing all its possible functional paths, is shown in Fig. 1b. Each node represents a register. The IN and OUT nodes represent memory. A directed edge between registers represents a possible functional path between registers. For example, there is an edge between the AC node and the OUT node. This edge indicates that there exists a possible functional path from register AC to memory.

From the dataflow diagram, we can calculate the observability vector. First, we define the observability value for the OUT node. The primary output OUT has the highest observability since it is directly observable. Using SCOAP-like measure for observability, we define the observability value of OUT to be 0, written as $OUT_obs = 0$. For every other register node, we define its observability parameter as 1 plus the minimum of the observability parameters of all its fanout nodes. For example, the fanout nodes of register AC are OUT, SR, and AC itself. Thus the observability parameter of register AC is 1 plus the minimal observability parameter among OUT, SR, AC. That is, the observability parameter of register AC is 1. In the same way, we can obtain the observability parameters for MAR, PC, IR and SR. We define the observability value of a register as the reciprocal of its observability parameter. Finally, we obtain the observability vector for Parwan. It is simply $\left(\frac{1}{AC_obs}, \frac{1}{IR_obs}, \frac{1}{PC_obs}, \frac{1}{MAR_obs}, \frac{1}{SR_obs}\right)$, i.e., (1, 0.5, 1, 1, 0.5).

The third contributor to RT-level output deviation is the weight vector, which is used to model how much combinational logic a register is connected to. Each register is assigned a weight value, representing the relative sizes of its input cone and fanout cone. Obviously, if a register has a large input cone and a large fanout cone, it will affect and be affected by many lines and gates. Thus it is expected to contribute more to defect detection. In order to accurately extract this information, we need gate-level information to calculate the weight vector. We only need to report the number of stuck-at faults for each component based on the gate-level netlist. This can be easily implemented without gate-level logic simulation by a design analysis tool. Consider each component in Parwan. There are 248 stuck-at faults in AC, 136 in IR, 936 in PC, 202 in MAR, 96 in SR, 14690 in ALU, and 464 in SHU. From this information, we can easily calculate the weight vector (Table 1).

Based on the RT-level description of the design, we can determine the fanin cone and fanout cone of a register. For example, the AC register is connected to three components: AC, SHU and ALU. Given a set of

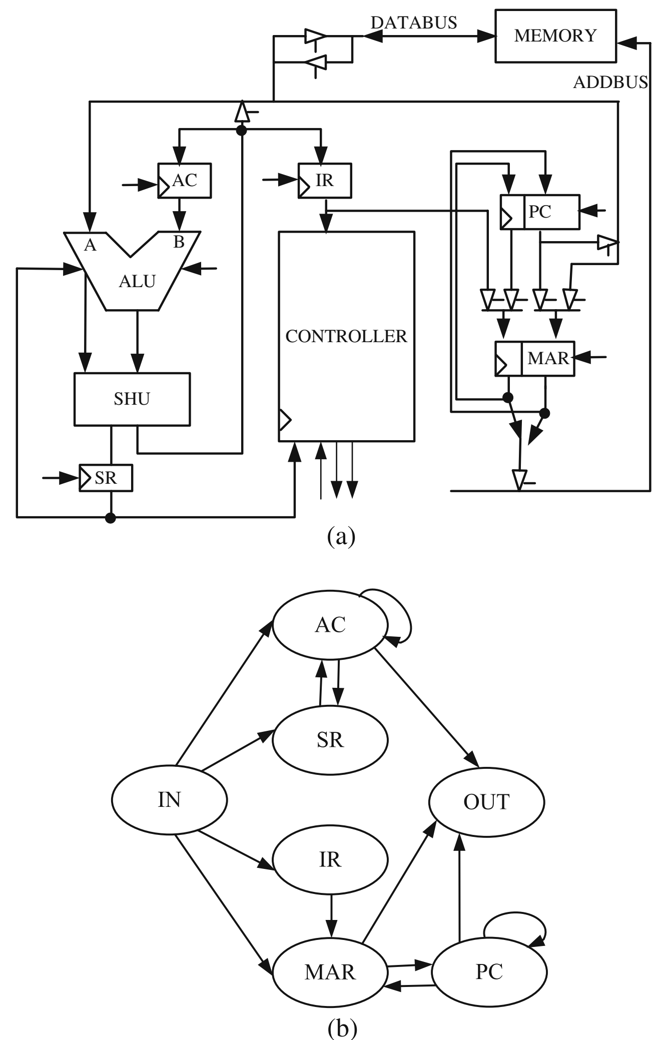

Fig. 1 **a** Architecture of the Parwan processor; **b** Dataflow graph of the Parwan processor (only the registers are shown)

Table 1 Weight vector for registers (Parwan)

	AC	*IR*	*PC*	*MAR*	*SR*
No. of faults affecting register	2172	338	936	202	2020
Weight value	1	0.1556	0.4309	0.093	0.930

registers $\{R_i\}, i = 1, 2, .., n$, let f_i be the total number of stuck-at faults in components connected to register R_i. Let $f_{max} = \max\{f_1, .., f_n\}$. We define the weight of register R_i as f_i/f_{max} to normalize the size of gate-level logic. Table 1 shows the numbers of faults affecting registers and weights of registers. We can see that $f_{max} = 2172$ for Parwan processor, which is the number of faults in AC. The weight of IR can be calculated as 338/2172, i.e., 0.1556. In this way, weights of other registers can also be obtained. Finally, we get the weight vector (1, 0.1556, 0.4309, 0.093, 0.930).

In order to calculate output deviations, first the effective TCs for registers should be obtained. Suppose register *Reg* makes N_1 $0 \to 1$ transitions for a functional test sequence TS. Then its effective $0 \to 1$ TC equals the product of N_1, the observability value of register *Reg*, and the weight of register *Reg*.

We use Parwan to illustrate how to calculate the effective TCs. For a functional test sequence TS, the TCs corresponding to TS are shown in Table 2. In Table 2, each row shows the TC for one register, while each column represents the transition type. For example, the value of third row and second column is 206, which implies that the $0 \to 0$ TC for IR is 206 when functional test sequence TS is executed.

By considering the weight vector and the observability vector, the TC of a register can be transformed to the effective TC. Table 3 shows the effective TC values for TS for the given observability vector and weight vector.

In Table 3, each row lists the effective TC for one register. The last row shows the aggregated effective TC for all registers. The columns indicate the various types of transitions.

The following example illustrates how to calculate the deviation for a functional test sequences TS. Suppose TS is composed of 50 instructions I_1, I_2 ..., I_{50}.

Table 2 TCs for TS

	$0 \to 0$	$0 \to 1$	$1 \to 0$	$1 \to 1$
AC	67	61	60	44
IR	206	67	66	37
PC	708	90	85	205
MAR	913	251	246	246
SR	67	11	14	12

Table 3 Effective TCs for test sequence TS

Register ID	Register name	$0 \to 0$	$0 \to 1$	$1 \to 0$	$1 \to 1$
R_1	*AC*	67	61	60	44
R_2	*IR*	16.03	5.21	5.13	2.88
R_3	*PC*	305.08	38.78	36.63	88.33
R_4	*MAR*	84.91	23.34	22.88	22.88
R_5	*SR*	31.16	5.115	6.51	5.58
Sum		504.18	133.45	131.19	163.67

For each instruction I_i, suppose the effective $0 \to 0$ TC for register R_k $(1 \le k \le 5)$ is R_{ki00}, the effective $0 \to 1$ TC for register R_k is R_{ki01}, the effective $1 \to 0$ TC for register R_k is R_{ki10}, and the effective $1 \to 1$ TC for register R_k is R_{ki11}. Given the CL vector (1, 0.98, 0.98, 1), the CL value C_i for instruction I_i can be calculated by considering all possible transitions and the different registers:

$$C_i = \prod_{k=1}^{5} \left(1^{R_{ki00}} \cdot 0.998^{R_{ki01}} \cdot 0.998^{R_{ki10}} \cdot 1^{R_{ki11}}\right). \tag{3}$$

Using the property of the exponents, whereby $x^a \cdot x^b = x^{a+b}$, Eq. 3 can be rewritten as

$$C_i = 1^{\sum_{k=1}^{5} R_{ki00}} \cdot 0.998^{\sum_{k=1}^{5} R_{ki01}} \cdot 0.998^{\sum_{k=1}^{5} R_{ki10}} \cdot 1^{\sum_{k=1}^{5} R_{ki11}}. \tag{4}$$

Let $S_{i00} = \sum_{k=1}^{5} R_{ki00}$, $S_{i01} = \sum_{k=1}^{5} R_{ki01}$, $S_{i10} = \sum_{k=1}^{5} R_{ki10}$, and $S_{i11} = \sum_{k=1}^{5} R_{ki11}$. Equation 4 can now be written as

$$C_i = 1^{S_{i00}} \cdot 0.998^{S_{i01}} \cdot 0.998^{S_{i10}} \cdot 1^{S_{i11}}. \tag{5}$$

Based on the deviation definition, the deviation for TS can be calculated as $\Delta(TS) = 1 - \prod_{i=1}^{50} C_i$, i.e.,

$$\begin{aligned}\Delta(TS) &= 1 - \prod_{i=1}^{50} \left(1^{S_{i00}} \cdot 0.998^{S_{i01}} \cdot 0.998^{S_{i10}} \cdot 1^{S_{i11}}\right)\\ &= 1 - 1^{\sum_{i=1}^{50} S_{i00}} \cdot 0.998^{\sum_{i=1}^{50} S_{i01}} \cdot 0.998^{\sum_{i=1}^{50} S_{i10}} \cdot 1^{\sum_{i=1}^{50} S_{i11}}\end{aligned} \tag{6}$$

Let $S^*_{00} = \sum_{i=1}^{50} S_{i00}$, $S^*_{01} = \sum_{i=1}^{50} S_{i01}$, $S^*_{10} = \sum_{i=1}^{50} S_{i10}$ and $S^*_{11} = \sum_{i=1}^{50} S_{i11}$, Eq. 6 can be rewritten as:

$$\Delta(TS) = 1 - 1^{S^*_{00}} \cdot 0.998^{S^*_{01}} \cdot 0.998^{S^*_{10}} \cdot 1^{S^*_{11}}. \tag{7}$$

Note that S^*_{00} is the aggregated effective $0 \to 0$ TC of all registers for all the instructions in TS. The parameters S^*_{01}, S^*_{10}, and S^*_{11} are defined in a similar way.

4 Observation-point Selection

In this section, we first define the new concept of RT-level internal deviations. Next, we analyze the factors that determine observation-point selection. Then we present the observation-point selection algorithm based on RT-level deviations. Finally, we introduce recent related work [25], which will be used in Section 5 for comparison.

4.1 RT-level Internal Deviations

The RT-level output deviation [8] is defined to be a measure of the likelihood that error is manifested at a primary output. Here we define the RT-level internal deviation to be a measure of the likelihood of error being manifested at an internal register node, which means error being manifested at one or more bits of register outputs. In the calculation of RT-level output deviation, a transition in a register is meaningful only when it is propagated to a primary output. On the other hand, in the calculation of RT-level internal deviation, we do not care whether a transition in a register is propagated to a primary output. The method for calculating internal deviations for register can also be used to calculate internal deviations for each bit of a register.

The calculation of RT-level internal deviation is similar to that of RT-level output deviation. In order to calculate RT-level internal deviations for a register, first we need to define the internal effective TCs. Suppose a register *Reg* makes N_1 $0 \rightarrow 1$ transitions for a functional test sequence TS. Then its internal effective $0 \rightarrow 1$ TCs equals the product of N_1 and the weight of register *Reg*. The observability value of register *Reg* does not contribute to its effective transition count. Take IR in the Parwan processor as an example. It has the observability value 0.5 and weight 0.1556. Since it makes 67 $0 \rightarrow 1$ transitions for functional test sequence TS, its internal effective $0 \rightarrow 1$ transition count is 0.1556×67, i.e., 10.43. In the same way, we can calculate the internal effective TCs of all registers for TS, as shown in Table 4.

Table 4 Internal effective TCs for test sequence TS

Register ID	Register name	$0 \rightarrow 0$	$0 \rightarrow 1$	$1 \rightarrow 0$	$1 \rightarrow 1$
R_1	AC	67	61	60	44
R_2	IR	32.06	10.43	10.26	5.76
R_3	PC	305.08	38.78	36.63	88.33
R_4	MAR	84.91	23.34	22.88	22.88
R_5	SR	62.32	10.23	13.02	11.16

After obtaining the internal effective transition counts, we can calculate the RT-level internal deviations. Suppose a functional test sequence TS is composed of m instructions I_1, I_2 ..., I_m. For each instruction I_i, suppose the internal effective $0 \rightarrow 0$ TCs for register R_k $(1 \leq k \leq t)$ is I_R_{ki00}, the internal effective $0 \rightarrow 1$ TCs for register R_k is I_R_{ki01}, the internal effective $1 \rightarrow 0$ TCs for register R_k is I_R_{ki10}, and the internal effective $1 \rightarrow 1$ TCs for register R_k is I_R_{ki11}. Let $I_S_{k00} = \sum_{i=1}^{m} I_R_{ki00}$, $I_S_{k01} = \sum_{i=1}^{m} I_R_{ki01}$, $I_S_{k10} = \sum_{i=1}^{m} I_R_{ki10}$, $I_S_{k11} = \sum_{i=1}^{m} I_R_{ki11}$. The internal deviation of register R_k for TS can be calculated for a given CL vector $(cl_{00}, cl_{01}, cl_{10}, cl_{11})$ as follows:

$$I_{dev}(R_k) = 1 - {cl_{00}}^{I_S_{k00}} \cdot {cl_{01}}^{I_S_{k01}} \cdot {cl_{10}}^{I_S_{k10}} \cdot {cl_{11}}^{I_S_{k11}}.$$

Note that I_S_{k00} is the aggregated internal effective $0 \rightarrow 0$ TCs of register R_k for all the instructions in TS. The parameters I_S_{k01}, I_S_{k10}, and I_S_{k11} are defined in a similar way. In this way, we can calculate the internal deviations of every register for TS. The method for calculating internal deviations for register can also be used to calculate internal deviations for each bit of a register.

4.2 Analysis of Factors that Determine Observation-point Selection

The selection of observation points is determined by three factors: RT-level internal deviations of registers, observability values of registers, and the topological relationship between registers. In this work, we only consider the insertion of observation points at outputs of registers.

For a register *Reg*, we have the following attributes attached with it: $I_{dev}(Reg)$, $O_{dev}(Reg)$, $obs(Reg)$, to represent its internal deviation, output deviation, and observability value, separately.

For two registers *Reg*1 and *Reg*2, when two attributes are close in value, we define the following observation-point-selection rules based on the third attribute:

Rule 1: If $I_{dev}(Reg1) > I_{dev}(Reg2)$, select *Reg*1;
Rule 2: If $obs(Reg1) < obs(Reg2)$, select *Reg*1;
Rule 3: If *Reg*1 is the logical predecessor of *Reg*2, select *Reg*2.

For Rule 1, the motivation is that if we select a register with higher I_{dev}, its observability will become 1. Thus, its O_{dev} will also becomes higher. The higher O_{dev} of this register will contribute more to the cumulative O_{dev} for the circuit. Since we have shown that the cumulative O_{dev} is a good surrogate metric for gate-level

fault coverage [8], we expect to obtain better gate-level fault coverage when we select a register with higher I_{dev}.

For Rule 2, when two registers do not have a predecessor/successor relationship with each other, obviously we should select the register with lower observability. For Rule 3, if we select $Reg1$, $obs(Reg1)$ will become 1 but this will not contribute to the increase of observability of $Reg2$; if we select $Reg2$, $obs(Reg2)$ will become 1 and $obs(Reg1)$ will also be increased due to the predecessor relationship between $Reg1$ and $Reg2$. Therefore, it is possible that the selection of $Reg2$ yields better results than the selection of $Reg1$, i.e., the cumulative observability after the insertion of observation point on $Reg2$ is higher than for $Reg1$.

Rule 2 and Rule 3 are in conflict with each other on the observability attribute. Rule 2 selects a register with lower observability while Rule 3 selects a register with higher observability. In this work, we assume that Rule 3 is given higher priority than Rule 2.

We use RT-level output deviations to guide the selection of observation points. We have determined that we should select a register with higher I_{dev}. Since O_{dev} is proportional to I_{dev} and obs, if I_{dev} factor contributes more to O_{dev}, we should select the register with higher O_{dev}. Also, by selecting a register with higher O_{dev}, we are implicitly satisfying the predecessor relationship rule: for two registers $Reg1$ and $Reg2$ whose I_{dev} values are comparable, if $Reg1$ is the predecessor of $Reg2$, we have $obs(Reg1) < obs(Reg2)$ and $O_{dev}(Reg1) < O_{dev}(Reg2)$. Then we will not select $Reg1$, which is in accordance with Rule 3.

4.3 RT-level Deviation Based Observation-point Selection

Based on the RT-level output deviations, we have developed a method for selecting best set of n (where n is a user-specified parameter) observation points for a given RT-level design and a given functional test sequence. In the selecting of observation-points, we target the specific bits of a register. The calculation of I_{dev}, O_{dev}, obs for a register is carried out for each bit of a register. The selection procedure is as following:

– Step 0: Set the candidate list to be all bits of registers that do not directly drive a primary output.
– Step 1: Derive the topology information for the design and save this information in a look-up table. Obtain the weight vector, observability vector, and TCs for each register bit, and calculate RT-level output deviations for each register bit.
– Step 2: Select a register bit with the highest output deviations as an observation point. Remove this selected register bit from the candidate list.
– Step 3: If the number of selected observation points reaches n, terminate the selection procedure.
– Step 4: Update the observability vector using the inserted observation point (selected in Step 2) and the topology information. Re-calculate output deviations for each register bit using the updated observability vector. Go to Step 2.

In Step 1, the topology information of the design can be extracted using a design analysis tool, e.g., Design Compiler from Synopsys. Topology information extracted here refers to information of the upstream registers for each register. It only needs to be determined once and it can be saved in a look-up table for subsequent use. In Step 4, after selecting and inserting an observation point, we need to update the observability vector because the observability of its upstream nodes will also be enhanced. There is no need to recompute TCs since these depend only on the functional test

Table 5 Value of k for each circuit

	$b09$	$b10$	$b12$	$b13$	$b14$	$b15$
k	48	20	135	147	300	20

Table 6 Gate-level fault coverage (stuck-at and transition) of the design before and after inserting all observation points

Circuit	Original design		Design with all observation points		
	SFC%	TFC%	#OP	SFC%	TFC%
$b09$	59.18	47.93	27	82.8	67.86
$b10$	36.89	20.19	14	69.03	45.67
$b12$	50.25	26.67	115	55.23	31.92
$b13$	35.9	23.33	43	70.83	44.02
$b14$	83.95	74.6	161	92.34	83.32
$b15$	9.91	5.35	347	23.29	11.36

Table 7 Gate-level metrics ($BCE+$ and GE score) of the design before and after inserting all observation points

Circuit	Original design		Design with all observation points		
	BCE+%	GE score	#OP	BCE+%	GE score
$b09$	45.58	121	27	70.13	173
$b10$	28.04	132	14	55.07	330
$b12$	29.91	889	115	33.52	1005
$b13$	23.11	257	43	47.12	483
$b14$	74.52	8601	161	81.23	8934
$b15$	4.4	806	347	10.63	1987

sequence, and they are not affected by the observation points. There is also no need to re-calculate the weight vector.

After the n observation points have been selected, they are inserted in the original RT-level design. The modified RTL design is synthesized to a gate-level netlist. To insert an observation point, we simply need to connect it directly to a new primary output. An alternative method is to use only one additional primary output and connect all observation points to this primary output through XOR gates (space compactor). By doing so, we can reduce the number of extra primary outputs to one. However, this method will lead to lower fault coverage due to error masking.

4.4 Observation-point Selection Based on Probabilistic Observability Analysis

An automatic method to select internal observation signals for design verification was proposed in recent work [25]. Since this method is also applicable for observation-point selection in manufacturing test, we take it as an example of recent related work and

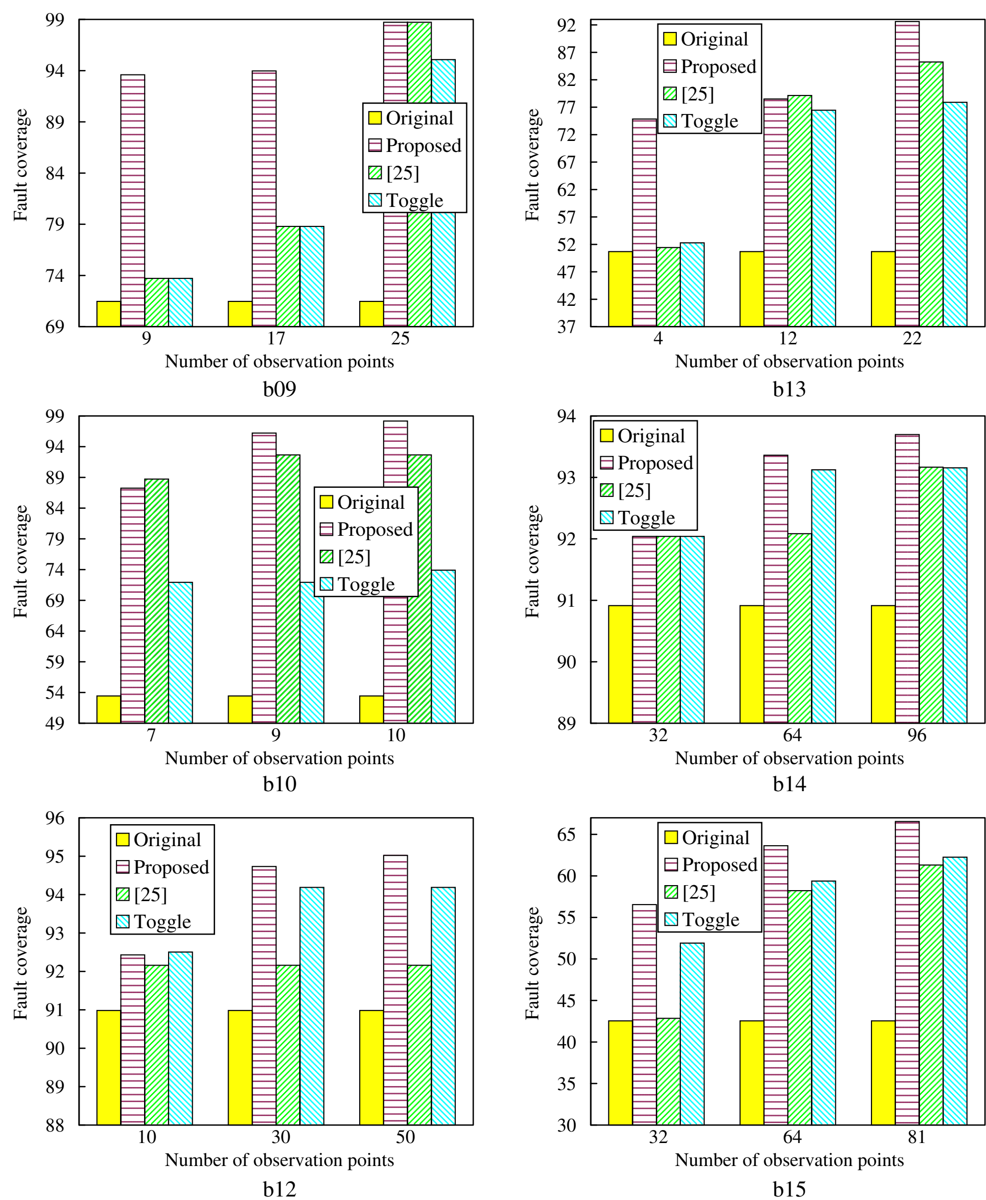

Fig. 2 Results on gate-level normalized stuck-at fault coverage

compare the results obtained by the proposed RT-level deviation based method to this method in Section 5.

5 Experimental Results

We evaluated the efficiency of the proposed RT-level observation-point selection method by performing experiments on six $ITC'99$ [5] circuits. These circuits are translated into Verilog format and are taken as the experimental vehicles. The functional test sequences are generated using the RT-level test generation method from [5]. Our goal is to show that the RT-level deviation-based observation-point selection method can provide higher defect coverage than other baseline methods. Here, defect coverage is estimated in terms of following gate level coverage metrics:

- stuck-at fault coverage;
- transition fault coverage;
- enhanced bridging fault coverage estimate ($BCE+$);
- gate-exhaustive (GE) score (GE score is defined as the number of observed input combinations of gates) [4, 17].

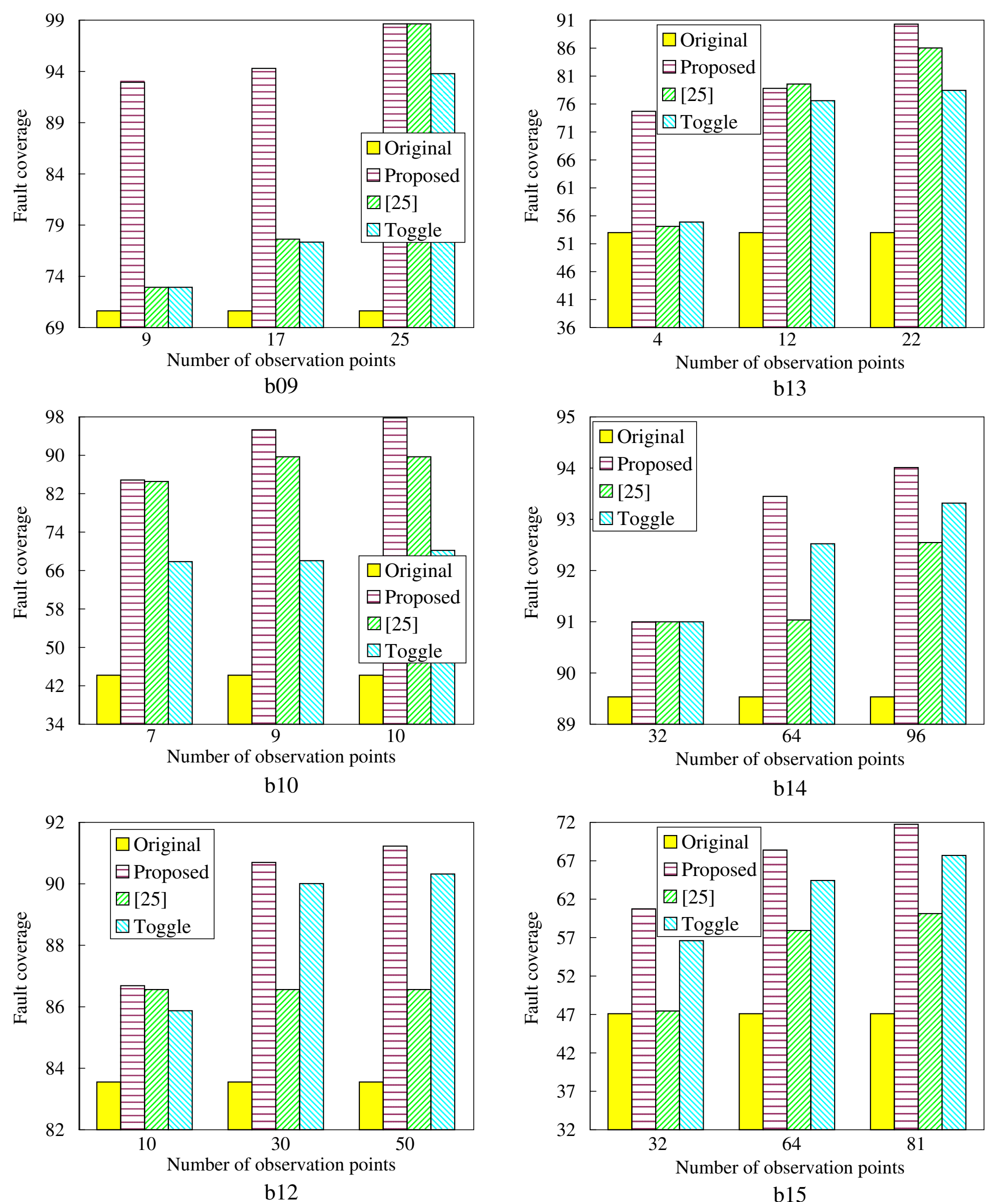

Fig. 3 Results on gate-level normalized transition fault coverage

Since functional test sequences are usually used to target unmodeled defects that are not detected by structural test, we considered metrics $BCE+$ and GE score, which are more effective for defect coverage, comparing to traditional stuck-at fault coverage and transition fault coverage. The GE score is defined as the number of the observed input combinations of gates. Here, "observed" implies that the gate output is sensitized to at least one of the primary outputs. We first compare the gate-level fault coverage for the original design to the design with all observation points inserted. Next we show the gate-level fault coverage for different observation-point selection methods.

5.1 Experimental Setup

All experiments were performed on a 64-bit Linux server with 4 GB memory. Synopsys Verilog Compiler (VCS) was used to run Verilog simulation and compute the deviations. The Flextest tool was used to run gate-level fault simulation. Design Compiler (DC) from Synopsys was used to synthesize the RT-level descriptions

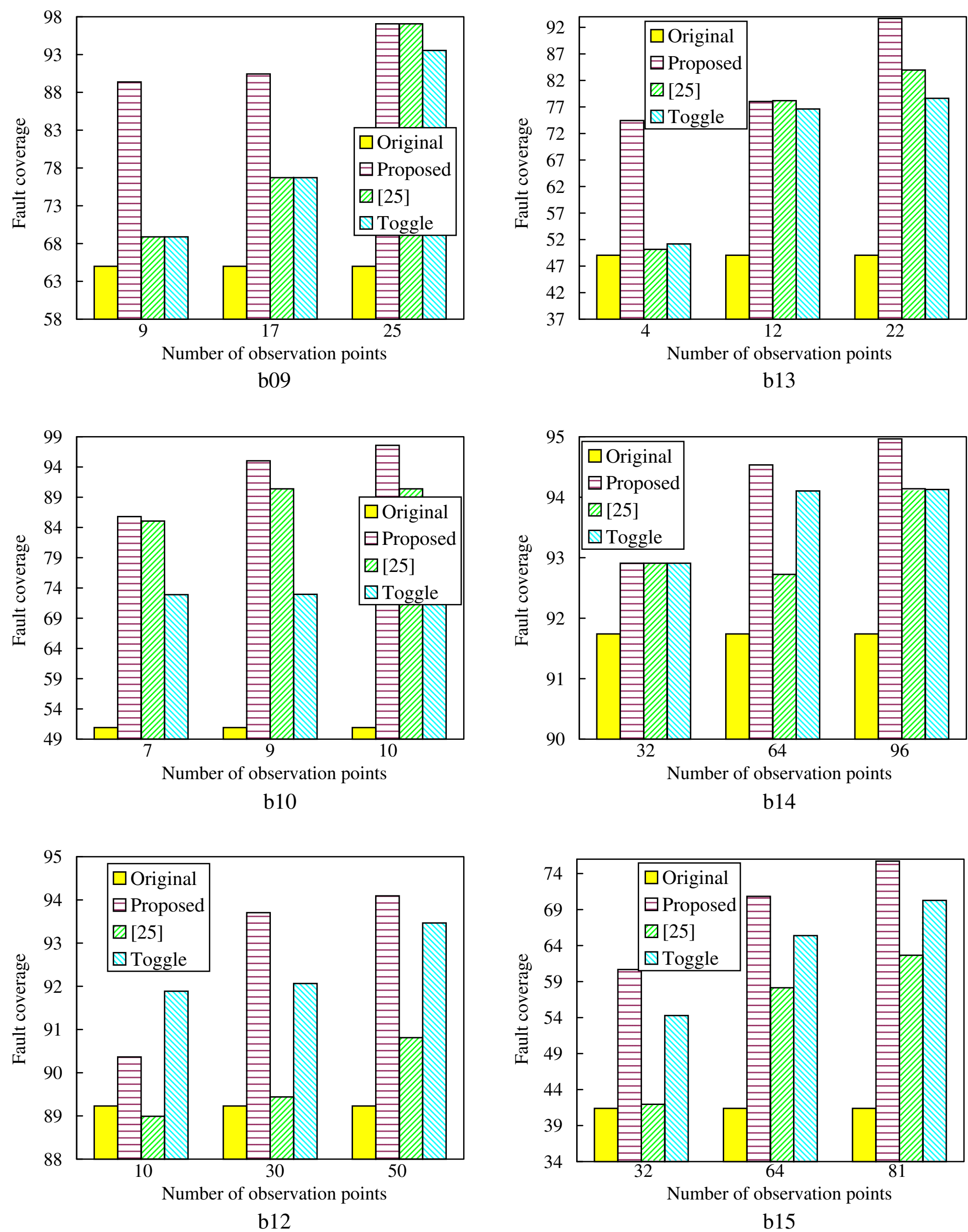

Fig. 4 Results on the gate-level normalized $BCE+$ metric

as gate-level netlists and extract the gate-level information for calculating the weight vector. For synthesis, we used the library for Cadence 180 nm technology. All other programs were implemented in C++ and Perl scripts.

5.2 *k*-toggle Coverage Based Observation-point Selection

RTL toggle coverage has been used as an approximation of gate-level fault coverage [7], and has been used for functional test selection [10]. Since RTL toggle coverage only checks whether the value of a wire/register toggles from 0 to 1 or toggles form 1 to 0, it often saturates too early during simulation. Therefore, we define a *k*-toggle coverage metric as follows.

k-toggle coverage: Suppose a wire/register bit toggles m times from 0 to 1 and toggles n times from 1 to 0 during simulation. If $m \geq k$ and $n \geq k$, the *k*-toggle coverage for this wire/register bit is defined to be 100%; If either $m \geq k$ or $n \geq k$, the *k*-toggle coverage for this wire/register bit is defined to be 50%; otherwise, the *k*-toggle coverage for this wire/register bit is defined to be 0.

For a register, its *k*-toggle coverage is defined to be the average *k*-toggle coverage of all bits. For example,

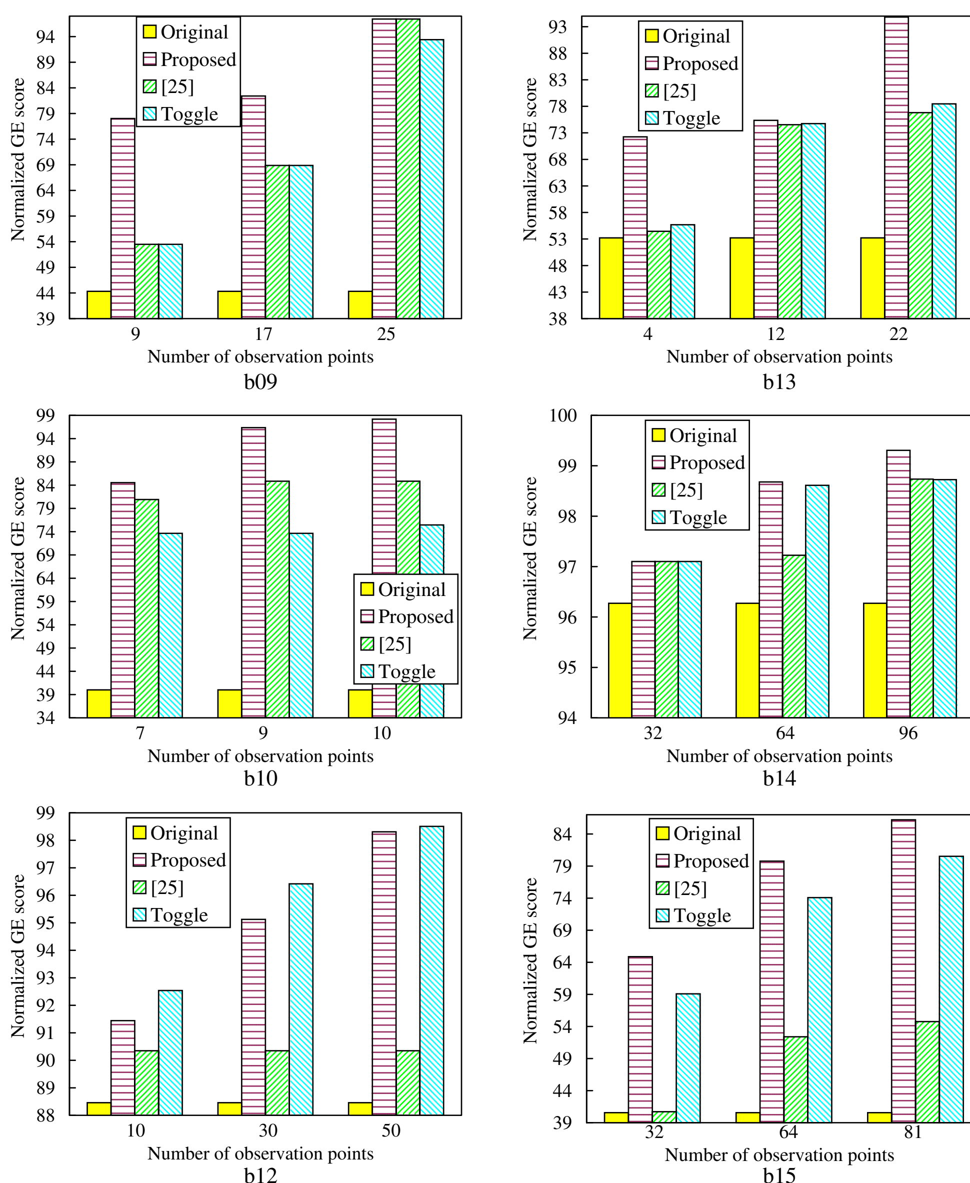

Fig. 5 Results on gate-level normalized GE score

suppose a register has three bits and the k-toggle coverage for each bit is 50%, 50%, 100%. Then, the k-toggle coverage for this register is calculated as $(50\% + 50\% + 100\%)/3$, i.e., 66.67%.

In order to evaluate the efficiency of the proposed method, we consider the observation-point selection based on the k-toggle coverage metric as a baseline. First, we obtain the k-toggle coverage for all the registers that are not directly connected to primary outputs (POs). Next, we select the registers with the highest k-toggle coverage as the observation points. In this work, k is set to be the average toggle counts for all register bits under the given functional test sequences. For example, if there are 3 register bits in a design and their toggles are recorded as follows: *reg*0 (0-to-1): 50 times; *reg*0 (1-to-0): 100 times; *reg*1 (0-to-1): 40 times; *reg*1 (1-to-0): 80 times; *reg*2 (0-to-1): 70 times; *reg*2 (1-to-0): 60 times. The average toggle count is calculated as: $(50 + 100 + 40 + 80 + 70 + 60)/(2 \times 3)$, i.e., 66.67. In this case, we set k to be 67 for this circuit. Table 5 lists the values of k for circuits used in the experiments.

5.3 Comparison of Gate-level Fault Coverage for the Original Design to the Design with all Observation Points Inserted

Tables 6 and 7 compare the gate-level metrics (stuck-at fault coverage, transition fault coverage, $BCE+$ and GE score) for the original design to the design will all observation points inserted. The parameters SFC%, TFC%, $BCE+$% indicate the gate-level fault coverage for stuck-at faults, transition delay faults and bridging fault estimate, respectively. #OP lists the number of observation points.

From these two tables, we can see that the gate-level fault coverage is not very high even when all observation points are inserted. There are two possible reasons for this: one reason is that the design suffers form low controllability. The other reason is that the quality of the given functional test sequences is not so effective for modeled fault. We can increase the gate-level fault coverage by improving the quality of functional test sequences or by inserting control points to the design. However, we focus here only on selection of observation points so that the given functional test sequences can be made more useful for manufacturing test. Therefore, it is of interest to determine the maximum gate-level fault coverage when all possible observation points are inserted, and to normalize the fault coverage to this maximum value when we evaluate the impact of inserting a subset of all possible observation points.

5.4 Comparison of Normalized Gate-level Fault Coverage for Different Observation-point Selection Methods

By considering the fault coverage of a design with all observation points inserted to be 100%, we normalize the fault coverage of designs with a smaller number of observation points. Similarly, the normalized GE score is obtained by taking the GE score of a design with all observation points inserted as the reference. In this section, we compare the normalized gate-level fault coverage and normalized GE score for different observation-point selection methods.

For each circuit, we select the same number of n (for various values of n) observation points using different methods. Results for normalized gate-level fault coverage and normalized GE score are shown in the Figs. 2, 3, 4 and 5. We compared the proposed method to [25] as well as the k-toggle coverage based observation-point insertion method. "Toggle" denotes the observation-point selection method based on k-toggle coverage in all the figures.

The results show that the proposed method outperforms the two baseline methods for all six circuits in terms of defect coverage. Since it is difficult to assess the real defect coverage, we estimated the gate-level fault coverage on stuck-at fault, transition fault, $BCE+$ and GE score metric. Let's see the result for $b15$ in Fig. 2. The x-axis represents the number of observation points. The y-axis represents the normalized gate-level stuck-at fault coverage. We can see that the proposed method can provide higher normalized stuck-at fault coverage than that provided by the method in [25] and the k-toggle coverage based method for the same number of observation points. This trend can be observed for all the circuits in all the figures. Also, by inserting a small fraction of all possible observation points using the proposed method, significant increase in fault coverage and GE score are obtained in all cases. For example, from the result for $b15$ in Fig. 2, we can see that by selecting and inserting 64 observation points using our method, the normalized stuck-at fault coverage can be increased from 47% to 67%. Recall that the number of all possible observation points for $b15$ is 417. Therefore, it means that we can obtain 20% increase in the normalized stuck-at fault coverage by only inserting about 15% observation points. For each circuit, it only takes a few seconds to calculate RT-level deviations and select observation points. These results highlight the effectiveness of the RT-level, deviation-based observation-point selection method.

6 Conclusion

We have presented an RT-level output deviations metric and shown how it can used to select and insert the observation points for a given RT-level design and a functional test sequence. This DFT approach allows us to increase the effectiveness of functional test sequences (derived for pre-silicon validation) for manufacturing testing. Experiments on six $ITC'99$ benchmark circuits show that the proposed RT-level DFT method outperforms two baseline methods for enhancing defect coverage. We have also shown that the RT-level deviations metric allows us to select a small set of the most effective observation points. As part of future work, we are extending this approach to the selection of control points.

References

1. Ahmed N, Tehranipoor M, Jayaram V (2006) Timing-based delay test for screening small delay defects. In: Design automation conference, pp 320–325
2. Aktouf C, Fleury H, Robach C (2000) Inserting scan at the behavioral level. IEEE Des Test Comput 17:34–42
3. Bushnell ML, Agrawal VD (2000) Essentials of electronic testing. Kluwer Academic, Norwell
4. Cho KY, Mitra S, McCluskey EJ (2005) Gate exhaustive testing. In: International test conference, pp 771–777
5. Corno F, Reorda MS, Squillero G (2000) RT-level ITC'99 benchmarks and first ATPG result. IEEE Des Test Comput 17:44–53
6. Dey S, Potkonjak M (1994) Non-scan design-for-testability of RT- level data paths. In: International conference on computer-aided design, pp 640–645
7. Drako D, Cohen P (1998) HDL verification coverage. In: Integrated system design
8. Fang H, Chakrabarty K, Jas A, Patil S, Trimurti C (2009) RT-level deviation-based grading of functional test sequences. In: VLSI test symposium, pp 264–269
9. Fujiwara H, Iwata H, Yoneda T, Ooi Y (2008) A nonscan design-for- testability method for register-transfer-level circuits to guarantee linear- depth time expansion models. IEEE Trans Comput-aided Des Integr Circuits Syst 27:1535–1544
10. Gangaram V, Bhan D, Caldwell JK (2006) Functional test selection using dynamic untestability analysis. In: Internaltional workshop on microprocessor test and verification
11. Gatej J et al (2002) Evaluating ATE features in terms of test escape rates and other cost of test culprits. In: International test conference, pp 1040–1049
12. Ghosh I, Fujita M (1999) Automatic test pattern generation for functional RTL circuits using assignment decision diagrams. In: Design automation conference, pp 43–48
13. Ghosh I, Raghunathan A, Jha NK (1995) Design for hierarchical testability of RTL circuits obtained by behavioral synthesis. In: IEEE international conference on computer design, pp 173–179
14. Ghosh I, Raghunathan A, Jha NK (1998) A design for testability technique for RTL circuits using control/dataflow extraction. IEEE Trans Comput-aided Des Integr Circuits Syst 17:706–723
15. Goloubeva O, Jervan G, Peng Z, Reorda MS, Violante M (2002) High-level and hierarchical test sequence generation. In: HLDVT, pp 169–174
16. Grossman JP, Salmon JK, Ho CR, Ierardi DJ, Towles B, Batson B, Spengler J, Wang SC, Mueller R, Theobald M, Young C, Gagliardo J, Deneroff MM, Dror RO, Shaw DE (2008) Hierarchical simulation-based verification of Anton, a special-purpose parallel machine. In: IEEE international conference on computer design, pp 340–347
17. Guo R, Mitra S, Amyeen E, Lee J, Sivaraj S, Venkataraman S (2006) Evaluation of test metrics: Stuck-at, bridge coverage estimate and gate exhaustive. In: VLSI test symposium, pp 66–71
18. Guzey O, Wang L-C (2007) Coverage-directed test generation through automatic constraint extraction. In: HLDVT, pp 151–158
19. Hosokawa T, Inoue R, Fujiwara H (2007) Fault-dependent/independent test generation methods for state observable FSMs. In: Asian test symposium, pp 275–280
20. Huang Y, Tsai C-C, Mukherjee N, Samman O, Cheng W-T, Reddy SM (2002) Synthesis of scan chains for netlist descriptions at RT-level. Journal of Electronic Testing: Theory and Applications (JETTA) 18:189–201
21. Inoue R, Hosokawa T, Fujiwara H (2008) A test generation method for state-observable FSMs to increase defect coverage under the test length constraint. In: Asian test symposium, pp 27–34
22. Kang J, Seth SC, Gangaram V (2007) Efficient RTL coverage metric for functional test selection. In: VLSI test symposium, pp 318–324
23. Kim H, Hayes JP (1998) High-coverage ATPG for datapath circuits with unimplemented blocks. In: International test conference, pp 577–586
24. Lin X et al (2006) Timing-aware atpg for high quality at-speed testing of small delay defects. In: Asian test symposium, pp 139–146
25. Lv T, Li H, Li X (2009) Automatic selection of internal observation signals for design verification. In: VLSI test symposium, pp 203–208
26. Mao W, Gulati RK (1996) Improving gate level fault coverage by RTL fault grading. In: International test conference, pp 150–159
27. Mathaikutty DA, Ahuja S, Dingankar A, Shukla S (2007) Model-driven test generation for system level validation. In: HLDVT, pp 83–90
28. Maxwell PC, Hartanto I, Bentz L (2000) Comparing functional and structural tests. In: International test conference, pp 400–407
29. Navabi Z (1997) VHDL: analysis and modeling of digital systems. McGraw-Hill, Hightstown
30. Norwood RB, McCluskey EJ (1996) Orthogonal scan: low overhead scan for data paths. In: International test conference, pp 659–668
31. Ravi S, Jha NK (2001) Fast test generation for circuits with RTL and gate-level views. In: International test conference, pp 1068–1077
32. Rearick J, Rodgers R (2005) Calibrating clock stretch during AC scan testing. In: International test conference, pp 266–273
33. Santos MB, Goncalves FM, Teixeira IC, Teixeira JP (2001) RTL-based functional test generation for high defects

coverage in digital systems. Journal of Electronic Testing: Theory and Applications (JETTA) 17:311–319
34. Thaker PA, Agrawal VD, Zaghloul ME (1999) Validation vector grade (VVG): a new coverage metric for validation and test. In: VLSI test symposium, pp 182–188
35. Thaker PA, Agrawal VD, Zaghloul ME (2000) Register-transfer level fault modeling and test evaluationtechniques for VLSI circuits. In: International test conference, pp 940–949
36. Vij AK (2002) Good scan=good quality level? well, it depends.... In: International test conference, p 1195
37. Wada H, Masuzawa T, Saluja KK, Fujiwara H (2000) Design for strong testability of RTL data paths to provide complete fault efficiency. In: IEEE international conference on VLSI design, pp 300–305
38. Wang Z, Chakrabarty K (2008) Test-quality/cost optimization using output-deviation-based reordering of test patterns. IEEE Trans Comput-aided Des Integr Circuits Syst 27:352–365
39. Wang Z, Fang H, Chakrabarty K, Bienek M (2009) Deviation-based LFSR reseeding for test-data compression. IEEE Trans Comput-aided Des Integr Circuits Syst 29:259–271
40. Yogi N, Agrawal VD (2006) Spectral RTL test generation for gate-level stuck-at faults. In: Asian test symposium, pp 83–88

Hongxia Fang received the B.S. degree in mathematics from Nankai University, Tianjin, China, in 2002 and the M.S. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2005. She is currently working toward the Ph.D. degree in electrical and computer engineering in the Department Electrical and Computer Engineering, Duke University, Durham, NC. Her current research is focused on design-for-testability (DFT), testing and diagnosis to target unmodeled defects in integrated circuits and multi-chip boards.

Krishnendu Chakrabarty received the B. Tech. degree from the Indian Institute of Technology, Kharagpur, in 1990, and the M.S.E. and Ph.D. degrees from the University of Michigan, Ann Arbor, in 1992 and 1995, respectively. He is now Professor of Electrical and Computer Engineering at Duke University. He is also a member of the Chair Professor Group (honorary position) in Software Theory at the School of Software, Tsinghua University, Beijing, China. Prof. Chakrabarty is a recipient of the National Science Foundation Early Faculty (CAREER) award, the Office of Naval Research Young Investigator award, the Humboldt Research Fellowship from the Alexander von Humboldt Foundation, Germany, and several best papers awards at IEEE conferences. His current research projects include: testing and design-for-testability of integrated circuits; digital microfluidics and biochips, circuits and systems based on DNA self-assembly, and wireless sensor networks. He has authored nine books on these topics (including two in press), published over 320 papers in journals and refereed conference proceedings, and given over 130 invited, keynote, and plenary talks.

Prof. Chakrabarty is a Fellow of IEEE, a Golden Core Member of the IEEE Computer Society, and a Distinguished Engineer of ACM. He is a 2009 Invitational Fellow of the Japan Society for the Promotion of Science (JSPS). He is recipient of the 2008 Duke University Graduate School Dean's Award for excellence in mentoring. He served as a Distinguished Visitor of the IEEE Computer Society during 2005–2007, and as a Distinguished Lecturer of the IEEE Circuits and Systems Society during 2006–2007. Currently he serves as an ACM Distinguished Speaker. He is an Associate Editor of IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on VLSI Systems, and IEEE Transactions on Biomedical Circuits and Systems. He also serves as an Editor of the Journal of Electronic Testing: Theory and Applications (JETTA). He is the Editor-in-Chief for IEEE Design & Test of Computers, and the Editor-in-Chief for ACM Journal on Emerging Technologies in Computing Systems.

Hideo Fujiwara received the B.E., M.E., and Ph.D. degrees in electronic engineering from Osaka University, Osaka, Japan, in 1969, 1971, and 1974, respectively. He was with Osaka University from 1974 to 1985 and Meiji University from 1985 to 1993, and joined Nara Institute of Science and Technology in 1993. Presently he is a Professor at the Graduate School of Information Science, Nara Institute of Science and Technology, Nara, Japan. His research interests are logic design, digital systems design and test, VLSI CAD and fault tolerant computing, including high-level/logic synthesis for testability, test synthesis, design for testability, built-in self-test, test pattern generation, parallel processing, and computational complexity. He has published over 350 papers in refereed journals and conferences, and nine books including the book from the MIT Press (1985) entitled "Logic Testing and Design for Testability." He received the IECE Young Engineer Award in 1977, IEEE Computer Society Certificate of Appreciation Awards in 1991, 2000 and 2001, Okawa Prize for Publication in 1994, IEEE Computer Society Meritorious Service Awards in 1996 and 2005, IEEE Computer Society Continuing Service Award in 2005, and IEEE Computer Society Outstanding Contribution Award in 2001 and 2009.

He served as an Editor of the IEEE Trans. on Computers (1998–2002), Journal of Electronic Testing: Theory and Application (1989–2004), Journal of Circuits, Systems and Computers (1989–2004), VLSI Design: An Application Journal of Custom-Chip Design, Simulation, and Testing (1992–2005), and several guest editors of special issues of IEICE Transactions of Information and Systems. He is currently an advisory member of IEICE Trans. on Information and Systems. Dr. Fujiwara is a fellow of the IEEE, a Golden Core member of the IEEE Computer Society, a fellow of the IEICE (the Institute of Electronics, Information and Communication Engineers of Japan) and a fellow of the IPSJ (the Information Processing Society of Japan).