WRTLT'11

IEEE Twelfth International Workshop on

RTL AND HIGH LEVEL TESTING

In conjunction with

VSI, India and MNIT, Jaipur

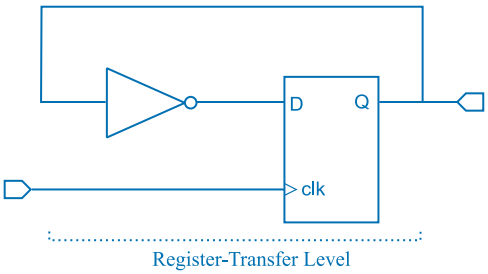
at MNIT, Jaipur, India

November 25–26, 2011

IEEE

PROCEEDINGS

Welcome Message



Welcome to WRTLT 2011, the twelfth annual IEEE International Workshop RTL and High Level Testing being held at Jaipur, November 25-26, 2011. The Steering Committee's vision in organising this annual technical meeting, to be held each winter, is to complement the broader International Conference on Asian Test Symposium and Asian Test Week by bringing into sharper focus key challenges and design issues related to RTL and High Level Testing of aggressively scaled and extremely complex microelectronic systems.

The scaling of CMOS technology has now clearly come up against physical limits of material properties and lithography, raising many new challenges that must be overcome to ensure IC quality and reliability. Unfortunately, there appears to be no obvious alternate technology that can replace End-of-Roadmap CMOS over the next decade. Therefore, the many testing challenges resulting from increasing defect rates, manufacturing variations, soft errors, wear out, etc. need to be addressed by innovative new design and test methodologies at the higher level of abstractions, if device scaling is to continue on track as per Moore's Law to 10nm and beyond. A key objective of this workshop is to provide an informal forum for spirited creative discussion and debate of these issues. The aim is to encourage the presentation and discussion of truly innovative and "out-of-the-box" ideas to address these many challenges that may not yet have been fully developed for presentation at more formally reviewed conferences.

We are very pleased to introduce WRTLT-11 program. On a day and a half, we will have a keynote address, a panel discussion and six sessions where 19 technical papers will be presented. The workshop begins on November 25 with a keynote address by Prof. Masahiro Fujita (University of Tokyo, Japan). On November 25-26, we will have 19 papers accepted to WRTLT. These papers are organized into a total of six sessions entitled High Level Verification and Testing, Delay Testing, IEEE Standards 1149.1 & P1687, Compaction, Secure & Dependable Design, Processor Testing, ATPG Acceleration, and Delay Measurement & Testing. The workshop is concluded by a panel discussion through our panel of experts. A tasteful visit to the pink city of Jaipur has also been arranged for the participants. We are sure that they will enjoy the natural shine and hospitality of Rajasthan.

Any technical workshop is the result of the work of many dedicated volunteers; WRTLT-11 has certainly been a team effort. We would like to wholeheartedly thank the Steering Committee, the Organizing Committee and the Program Committee. We also wish to thank all the authors who submitted their research ideas to WRTLT-11, and all the program participants for their contribution at the workshop. Finally, thanks are owed to the IEEE Computer Society Test Technology Technical Council for its technical sponsorship and support. We are also thankful to Malaviya National Institute of Technology, Jaipur (MNIT), VSI, and DIT, Government of India for supporting and sponsoring the meeting.

We hope that you will find WRTLT-11 enlightening and thought provoking, and overall a rewarding and enjoyable experience.

Rubin Parekhji (TI, India)
M.S.Gaur (MNIT, India)
General Co-Chairs

Michiko Inoue (NAIST, Japan)
Virendra Singh (IISc, India)
Program Co-Chairs

Technical Program-WRTLT 2011

Day-1 (Friday, 25th November)

Time	Event
8:30 to 9:00	Registration, Coffee
9:00 to 9:15	Opening Ceremony
9:15 to 9:50	Keynote Speaker 1 : Masahiro Fujita Topic: Structural and Functional Test Generation from RTL/Behavioural Description
9:50 to 11:05	Session - 1 *High Level Verification and Testing* *Session Chair: Zainalabedeen Navabi*
11:05 to 11:20	Tea Break
11:20 to 13:00	Session - 2 *Delay Testing* *Session Chair: Chia Yee Ooi*
13:00 to 14:00	Lunch
14:00 to 19:00	Social Event
19:00 to 21:00	Banquet
End of Day - 1	

Day-2 (Saturday, 26th November)

Time	Event
8:30 to 9:10	Keynote Speaker 2 : Hans Joachim Wunderlich Mixed - level Techniques for Test, Validation and Evaluation of Digital Systems
9:10 to 10:25	Session-3 *IEEE Standards 1149.1 & P1687, Compaction* *Session Chair : Joan Figueras*
10:25 to 10:45	Tea Break
10:45 to 11:15	Invited Talk Speaker : Jaan Raik Topic : Applications of RTL Test Generation : Past, Present and Future
11:15 to 12:30	Session - 4 *Secure & Dependable Design* *Session Chair : Ilia Polian*
12:30 to 13:30	Lunch
13:30 to 14:45	Session - 5 *Processor Testing, ATPG Acceleration* *Session Chair : Anzhela Mastrosova*
14:45 to 15:05	Tea Break
15:05 to 16:20	Session - 6 *Delay Measurement & Testing* *Session Chair : Ioana Vatajelu*
16:20 to 16:35	Tea Break
16:35 to 17:35	Panel Discussion
17:35 to 17:55	Vote of thanks
End of Day - 2 and End of WRTLT-11	

Table of Contents

Technical Program Committee, Organizing Committee, Steering Committee	5
Keynote & Invited Speakers	7
Nov 25 (Friday) :	
Opening Ceremony	
***Keynote Address 1:* Structural and Functional Test Generation from RTL / Behavioural Description** *Prof. Masahiro Fujita* *The University of Tokyo, Japan*	
Session I • High Level Verification and Testing **Chair : Zainalabedeen Navabi, University of Tehran, Iran**	
Built-in Self-Test for Functional Register - Transfer Level using Assignment Decision Diagram *Norlina Binti Paraman, Chia Yee Ooi, Ahmad Zuri Bin Sha'ameri, University Teknologi Malaysia, Malaysia & Hideo Fujiwara, Osaka Gakuin University, Japan*	9
A Binding Method for Hierarchical Testing Using Results of Test Environment Generation *Hiroaki Fujiwara, Toshinori Hosokawa, Ryoichi Inoue, Nihon University, Japan and Hideo Fujiwara, Osaka Gakuin University, Japan*	16
Testability Challenges of Mixed-Signal SoC with Integrated Power Management: Unified Top Level Design, Verification and Test Methodology *Lakshmanan Balasubramanian and R. K. Mittal, Texas Instruments (India) Pvt. Ltd., Bengaluru, India*	23
Session II • Delay Testing **Chair : Chia Yee Ooi, University Teknologi Malaysia, Malaysia**	
Additional Path Delay Fault Detection with Adaptive Test Data *Kohei Miyase, Hiroaki Tanaka, Kazunari Enokimoto, Xiaoqing Wen and Seiji Kajihara, Kyushu Institute of Technology, Iizuka, Japan*	31
On the Optimality of K Longest Path Generation *Ilia Polian,Jie Jiang, Matthias Sauer, University of Passau, Germany, Alexander Czutro and Bernd Becker, Albert-Ludwigs-University, Germany*	35
Path Selection for High-Quality Small Delay Defect Testing *Dong Xiang, School of Software, ISinghua University, China*	41
Robust PDFs Testing of Combinational Circuits based on Covering BDDs *Anzhela Matrosova, Ekaterina Nikolaeva, Sergey Ostanin, Tomsk State University, Russia, Virendra Singh, Indian Institute of Science, India*	47
Social Event	

Nov 26 (Saturday) :

Keynote Address 2: **Mixed-level Techniques for Test, Validation and Evaluation**
Hans - Joachim Wunderlich
Institut für Technische Informatik, Germany

Session III • IEEE Standards 1149.1 & P1687, Compaction
Chair : Joan Figueras, UPC, Spain

Extending BS-1149.1 for Interconnect Online BIST 53
Somayeh Sadeghi-Kohan, Ghazaleh Vazhbakht, Parisa Shaafi Kabiri and Zainalabedin Navabi, University of Tehran, Iran

A Study of Instrument Reuse and Retargeting in P1687 61
Farrokh Ghani Zadegan, Urban Ingelsson, Erik Larsson & Gunnar Carlsson, Linköping University, Ericsson

A Test Compaction Oriented Don't Care Identification Method 69
Hiroshi Yamazaki, Motohiro Wakazono, Toshinori Hosokawa, Nihon University, Japan & Masayoshi Yoshimura, Kyushu University, Fukuoka, Japan

Session IV • Secure & Dependable Design
Chair : Ilia Polian, University of Passau, Germany

SR-Quasi- Equivalents: Yet Another Approach to Secure and Testable Scan Design 77
Hideo Fujiwara, Osaka Gakuin University, Japan, Hideo Tamamoto and Katsuya Fujiwara, Akita University, Japan

Gracefully Degradable 3D On-Chip Networks Using an Optimized Rerouting Mechanism 83
Ali Shahabi, Reza Nakhjavani, Saeed Safari and Zainalabedin Navabi, University of Tehran, Iran

Self-Calibration using Functional BIST for Transient-Fault-Tolerant Sequential Circuits in Severe Electromagnetic Environment 90
Masayuki Arai, Aromhack Saysanasongkham, Kenta Imai, Yoshifumi Koyama and Satoshi Fukumoto, Tokyo Metropolitan University, Japan

Session V • Processor Testing, ATPG Acceleration
Chair : Anzhela Matrosova, Tomsk State University, Russia

On the Functional Test of Branch Prediction Units 94
Ernesto Sanchez and Matteo Sonza Reorda, Politecnico di Torino, Italy

Hierarchical Instruction Level Self Testing of Embedded Processor Cores 97
Parisa Sha'afi Kabiri and Zainalabedin Navabi, University of Tehran, Iran

An Approach to Hardware SAT Solver for Test Generation Based on Instance Similarity 105
Hideyuki Ichihara, Fumiyuki Hafuri, Kenji Ueda, Toshiya Mukai, Tsuyoshi Iwagaki and Tomoo Inoue, Hiroshima City University, Japan

Session VI • Delay Measurement & Testing
Chair : Ioana Vatajelu, UPC, Spain

Reconfigurable Array-Based Area-Efficient Test Structure for Standard Cell Characterization *Bishnu Prasad Das and Hidetoshi Onodera, Kyoto University, Japan*	110
ESDQL : A Metric for Evaluating Small Delay Defect Coverage *Xuefeng Zhu, Huawei Li and Xiaowei Li, Chinese Academy of Sciences, China*	116
Selection of the Flip-Flops for Partial Enhanced Scan Techniques *Ruslan Mukhamedov, Sergey Ostanin, Anzhela Matrosova, Alexey Melnikov, Tomsk State University, Tomsk, Russia and Virendra Singh, Indian Institute of Science, India*	122

Panel Discussion

Topic : Test Sign off at RTL : Will it be a reality
Moderator : Jaan Raik, Tallin Institute of Technology, Estonia

Panelist

Nilanjan Mukherjee (Mentor Graphics, USA)
Kazumi Hatayama (NAIST, Japan)
Parb Verma (Apache, USA)
Adit Singh (Auburn Univ, USA)

Author Index 127

Technical Program Committee

- *Chia Yee Ooi (UTM, Malaysia)*
- *Anzhela Matrosova (TSU, Russia)*
- *Tomokazu Yoneda (NAIST, Japan)*
- *Huawei Li (CAS, China)*
- *Seiji Kajihara (KIT, Japan)*
- *Kazumi Hatayama (NAIST, Japan)*
- *Ilia Polian (U.Passau, Germany)*
- *Erik Larsson (Linkoeping Univ, Sweden)*
- *Masaki Hashizume (Univ. of Tokushima, Japan)*
- *Tomoo Inoue (Hiroshima Univ, Japan)*
- *Yasuo Sato (KIT, Japan)*
- *Satoshi Ohtake (Oita Univ, Japan)*
- *Hideyuki Ichihara (HCU, Japan)*
- *Xioqiang Wen (KIT, Japan)*
- *Dong Xiang (Tsinghua Univ, China)*

Organizing Committee

General Co-Chairs
Rubin Parekhji (TI, India)
MS Gaur (MNIT, India)

Program Co-Chairs
Michiko Inoue (NAIST, Japan)
Virendra Singh (IISc, India)

Finance Chair
Vijay Laxmi (MNIT, India)

Publication Chair
Susanta Chakravarty (BESU, India)

Publicity Chair
Chia Yee Ooi (UTM, Malaysia)
Zhiqiang You (Hunan Univ, China)

Local Organization Chair
Lava Bhargava (MNIT, India)

Registration Chair
Jaynarayan Tudu (IISc, India)
Ashok Sihag (GBU, India)

Web Site Chair
Pawan Kumar (IISc, India)

Steering Committee

- *Tomoo Inoue (Hiroshima Univ, Japan) –* ***Chair***
- *Dong Xiang (Tsinghua Univ, China) –* ***Vice-Chair***
- *Hideo Fujiwara (Osaka Gakuin Univ, Japan)*
- *Masaki Hashizume (Univ. of Tokushima, Japan)*
- *Toshinori Hosokawa (Nihon Univ, Japan)*
- *Kazuhiko Iwasaki (Tokyo Metropolitan Univ, Japan)*
- *Erik Larsson (Linkoeping Univ, Sweden)*
- *Xiaowei Li (Chinese Academy of Sciences, China)*
- *Alex Orailoglu (UCSD, US)*
- *Kewal Saluja (Univ. of Wisconsin, Madison, US)*
- *Virendra Singh (IISc, India)*
- *Hideo Tamamoto (Akita Univ, Japan)*
- *Dafang Zhang (Hunan Univ, China)*

Keynote and Invited speakers

Keynote Speaker

Prof. Masahiro Fujita
Topic: Structural and Functional Test Generation from RTL / Behavioural Description

Biography:
Prof. Masahiro Fujita received his Ph.D. from the University of Tokyo in 1985. He is a Professor in VLSI Design and Education Center (VDEC) at the University of Tokyo. Prior to joining the University of Tokyo in 2000, he was Director of CAD in Fujitsu Laboratories of America for 6 years. He has done innovative works in the areas of digital design verification, synthesis, and testing. He has co-authored 6 books, and has over 150 publications. He has participated and chaired many prestigious conferences in CAD and VLSI designs. His current research interests include synthesis and verification in higher level design stages, hardware/sfotware co-designs and digital/analog co-designs.

Keynote Speaker

Hans - Joachim Wunderlich
Topic: - Mixed-level Techniques for Test, Validation and Evaluation of Digital Systems.

Biography:
Prof. Hans - Joachim Wunderlich received a Diploma in Mathematics from the University of Freiburg in 1981 and the Dr. rer. nat. (Ph.D.) from the University of Karlsruhe in 1986. Since 1991 he has been a full Professor and since 2002 he has been the director of the Institute of Computer Architecture and Computer Engineering at the University of Stuttgart. He is editor of various international journals and program committee member of a variety of IEEE conferences on design and test of electronic systems. In 2009 he became an IEEE Fellow.

Invited Speaker

Jaan Raik

Topic: - Applications of RTL Test Generation: Past, Present and Future

Register - Transfer Level (RTL) is a well defined design representation, which elevates the abstraction level from logic to that of arithmetic functions. Different from higher levels, RTL preserves the notion of circuit topology and cycle-accurate timing. RTL test pattern generation became a hot research topic in the '90s. Although the field is mature and established, RTL testing is very much alive as several challenges remain. We still are unable to handle large sequential designs. In addition, there are future applications which will be addressed in this talk. For example, RTL test generation may become another formal engine competing with SMT (SAT Modulo Theory) solvers in different areas, like in verification etc. Such an engine would be useful not only for designs described at RTL but also for more abstract descriptions that can be easily converted to RTL. Moreover, RTL test generation can be applied to advanced design optimization in synthesis of sequential circuits.

Biography:

Jaan Raik defended his MSc and Phd at Tallinn University of Technology (TUT) in 1997 and 2001, respectively. He is employed as a senior research fellow at TUT. Currently, he co-ordinates European FP7 co-operative research project DIAMOND. He was the local organization chair of ETS'11 and serves as the general chair of DDECS'12. His research interests include high-level test pattern generation, verification and debug, and network-on-chip testing. He has co-authored more than 100 papers.

Built-in Self-Test for Functional Register-Transfer Level using Assignment Decision Diagram

Norlina Binti Paraman[1], Chia Yee Ooi[1], Ahmad Zuri Bin Sha'ameri[1] and Hideo Fujiwara[2]
[1]Faculty of Electrical Engineering
University Teknologi Malaysia
81300 Skudai Johor, Malaysia
{norlina, ooichiayee, zuri}@fke.utm.my
[2]Faculty of Informatics
Osaka Gakuin University
Osaka 564-8511 JAPAN
fujiwara@ogu.ac.jp

Abstract **-** *Built-in Self-Test (BIST) currently has high area overhead and long test application time. In this paper, we propose a new BIST technique at functional register-transfer-level (RTL) which is based on the test-per-clock scheme. We modify the original register into multiple input signature register (MISR) at behavioural descriptions to reduce area overhead. We test all combinational blocks simultaneously. This method results in lower area overhead, shorter test application time and comparably high fault coverage compared to the previous method.*

Keywords **:** Built-in Self-Test (BIST); multiple input signature register (MISR); reduced BIST-able Register Transfer-Level (RTL) circuit

1.Introduction

With the advance in semiconductor technology, the complexity of Very-Large-Scale Integration (VLSI) designs is growing and the cost of testing is increasing. The external test with time consuming scan-in and scan-out of test data and expensive test equipment becomes more inefficient. The problems of the external automated test equipment (ATE) can be overcome by integration of built-in-self-test (BIST) technique in a circuit. In this technique [1], test pattern generators (TPGs) were added to primary inputs (PIs) and response analysers (RAs) were added to primary outputs (POs). Thus, the circuit is enabled to generate test patterns internally and compact the output responses. However, adding these TPGs and RAs are not sufficient to achieve high fault coverage (for single stuck-at faults), if the circuit contains feedbacks in its structure. Therefore, several techniques of design-for-testability (DFT) have been proposed [1].

Generally, BIST techniques are classified into test-per-scan scheme and test-per-clock scheme. In the test-per-scan scheme, the test registers generate test patterns and compact test responses in a few clock cycles based on scan-in and scan-out processes through scan path. The disadvantages of this scheme are long test application time and cannot operate at-speed.

In the test-per-clock scheme, the test registers generate test patterns and compact test responses in one clock cycle. The advantages of this scheme are that at-speed is possible and test application time is short. However, test-per-clock requires higher hardware overhead than the test-per-scan scheme does.

BIST technique in the test-per-clock scheme [2,3,4,5] can be classified into two categories. The first category requires the simultaneous testing of all combinational blocks whereas the second category requires each module to be tested independently from the other modules. These two methods add test pattern generator (TPG) at all primary inputs and register analysers (RA) at all primary outputs.

According to [2], they test all combinational blocks simultaneously. Their DFT method augments a given circuit with BILBO (built in logic block observer) or CBILBO (concurrent BILBO) so that the circuit becomes easily testable. Every feedback of the circuit must contain one CBILBO or two BILBOs. The advantage of this scheme is short test application time, since all combinational blocks are tested simultaneously. However, this scheme has high area overhead since several internal registers are modified to become BILBOs and CBILBOs.

In order to reduce hardware overhead in [2], Masuzawa et. al [3] proposed a BIST method based on hierarchical test where each combinational module is tested independently from other modules. The DFT method for this approach called single control testability. This scheme does not require to add extra register but only adds extra combinational logics to augment a given data path so that it is easily testable. In other words, the area overhead is lower than approach in [2]. The disadvantage of this scheme is long test application time because only a single combinational module is tested at a time.

Yamaguchi et. al [4] proposed a new testability called concurrent single control testability to remedy the disadvantage of test application time approach in [3]. They extend the concept of the testability in [3]

by introducing concurrent testing so that multiple combinational modules can be tested concurrently at a time. This scheme has shorter test application time than approach in [3] but suffer from area overhead.

In this paper, we propose a BIST technique of test-per-clock scheme and test all combinational blocks simultaneously. We classify the circuit into three categories based on BIST type and BIST insertion method to make the circuit BIST-able. They are called primitive BIST-able RTL circuit, concurrent BIST-able RTL circuit and reduced BIST-able RTL circuit. There is no element DFT augmented into primitive BIST-able RTL circuit. For concurrent BIST-able RTL circuit, the DFT augmented are following the Wunderlich [2] approach. In Wunderlich [2], they obtained short test application time and high area overhead. However, for reduced BIST-able RTL circuit we introduce DFT method. We improve approach in [2] by keeping the advantage of short test application time and resolving the disadvantages of area overhead. In order to reduce area overhead, we replace CBILBO in [2] with MISR for each feedback because MISR also can generate test pattern and compact test response simultaneously similar likes CBILBO. We also get similarly high fault coverage to that of the approach in [2].

This paper is organized as follows. In Section 2, we present the classification of BIST-able circuit where all combinational blocks are tested simultaneously. In Section 3, we define the DFT method for reduced BIST-able RTL circuit. In Section 4, we present the case study and Section 5 for conclusions.

2. Classification of BIST-able circuits

In this section, we classify the circuits into three categories based on BIST type and BIST insertion method to make the circuit BIST-able. They are called primitive BIST-able RTL circuit, concurrent BIST-able RTL circuit and reduced BIST-able RTL circuit. For primitive BIST-able RTL circuit, the original circuit is not augmented with DFT. For concurrent BIST-able circuit RTL circuit, the original circuit is augmented with DFT by inserting one CBILBO for each loop. For reduced BIST-able RTL circuit, the original circuit is augmented with our proposed DFT by inserting only a MISR for each loop. MISR can operate similarly to CBILBO to generate test pattern and compact test response simultaneously. To facilitate our discussion, we define the primitive BIST-able, concurrent BIST-able and reduced BIST-able as follows.

Definition 1. An RTL circuit is called *primitive BIST-able* if its primary inputs and primary outputs are modified into TPGs and RAs respectively.

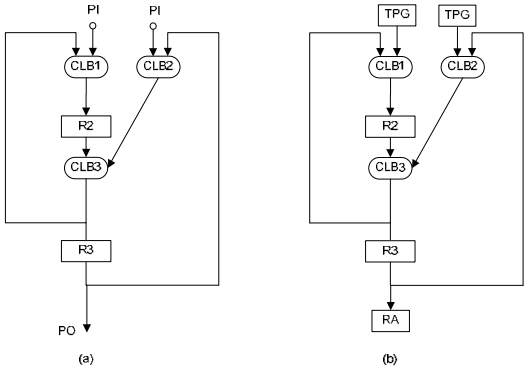


Fig. 1. (a) Original RTL circuit with two connected self loops S1. (b) Primitive BIST-able RTL circuit.

Example 1: Fig. 1(a) shows the original circuit which is not self-testable. Fig. 1(b) shows the primitive BIST-able RTL circuit. In order to make the circuit self-testable, we modify the PIs and POs into TPGs and RAs respectively as shown in Fig. 1(b). During testing, the TPGs apply pseudorandom patterns to CLB modules and the test responses of CLB modules are compressed by the register of PO which is modified into RA. Note that, the register R2 and R3 are not modified.

Referring to this example, there is no element DFT inserted into the primitive BIST-able RTL circuit. This approach obtains low fault coverage due to the feedback in the circuit. Note that for every loop, the register is not modified into test register.

According to Wunderlich et. al approach in [2], they extend the primitive BIST-able circuit by inserting two BILBOs or one CBILBO in each loop to guarantee high fault coverage. We call this as concurrent BIST-able RTL circuit. In order to define concurrent BIST-able, we define the transparent CBILBO first as follows.

Definition 2. A register, R is called transparent CBILBO or transparent MISR if R operates as signal in normal mode and R operates as CBILBO or MISR in test mode.

Definition 3. An RTL circuit is called *concurrent BIST-able* if the following conditions are satisfied:

i. Each loop has one CBILBO such that the CBILBO operates as TPG and RA simultaneously. If more than one register exist in one cycle, just one register is modified into CBILBO and the rest of original

registers are remained as unmodified register.

ii. Insert a transparent CBILBO instead of condition i, in case the cost of area overhead of condition ii is lower than that of condition i.

iii. Primary inputs and primary outputs are modified into TPGs and RAs respectively.

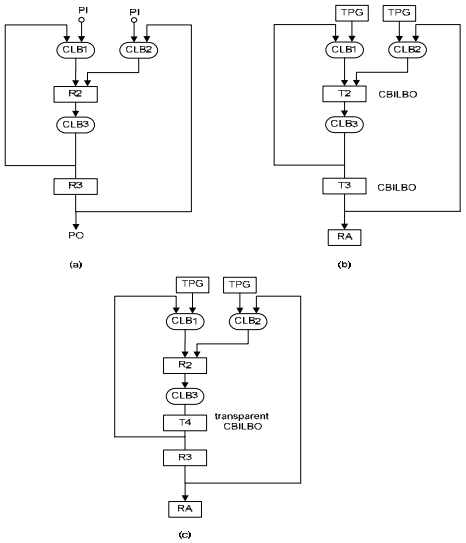


Fig. 2. (a) Original circuit (b) Concurrent BIST-able RTL circuit with CBILBO register T2 and CBILBO register T3 in each loop. (c) Concurrent BIST-able RTL circuit with one transparent CBILBO register T4.

Example 2: Fig. 2 shows two types of concurrent BIST-able RTL circuit. Fig. 2(b) shows concurrent BIST-able with CBILBO for each loop and Fig. 2(c) shows concurrent BIST-able with transparent CBILBO. Note that, in Fig. 2(a), the original register R2 is modified into CBILBO register T2 and original register R3 is modified into CBILBO T3. There is other way to make this original circuit self-testable. It also can be achieved by inserting just one transparent CBILBO T4 at the sharing path of both loops as shown in Fig. 2(c). Regarding the hardware cost, CBILBO is more expensive than BILBO and also inserting a transparent test register is more expensive than modifying an existing register with CBILBO.

By comparing the Example 1 and Example 2, it shows that the primitive BIST-able RTL circuit still has a loop because this approach does not break a loop with DFT element whereas the concurrent BIST-able RTL circuit does not have a loop anymore because the loop is broken by inserting CBILBO. Therefore, the fault coverage of concurrent BIST-able RTL circuit is higher than primitive BIST-able RTL circuit.

Completely similar to Wunderlich et.al approach in [2], our approach replaces CBILBO with MISR to break each loop. In our approach, the MISR not only can compact test responses but also can generate test patterns simultaneously. Our method can get high fault coverage and low area overhead. We call as reduced BIST-able RTL circuit.

Definition 4. An RTL circuit is called *reduced BIST-able* if the following conditions are satisfied:

i. Each loop has one MISR such that the MISR operates as TPG and RA simultaneously. If more than one register exist in one cycle, just one register is modified into MISR and the rest of original registers are remained as unmodified register.

ii. Insert a transparent MISR instead of condition i, in case the cost of area overhead of condition ii is lower than that of condition i.

iii. Primary inputs and primary outputs are modified into TPGs and RAs respectively.

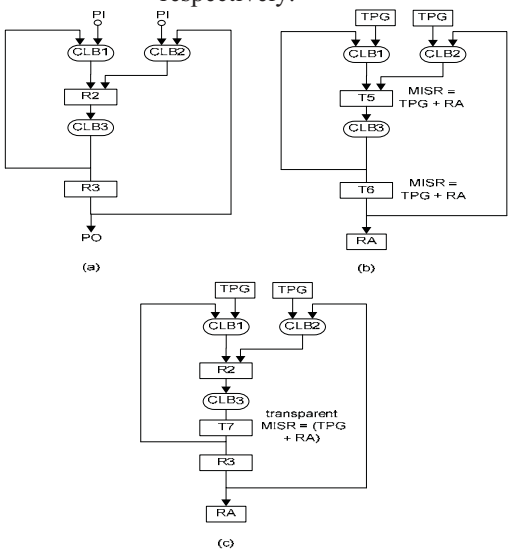


Fig. 3. (a) Original circuit (b) Reduced BIST-able RTL circuit with MISR register T5 and MISR register T6 in each loop. (c) Reduced BIST-able RTL circuit with one transparent MISR register T7.

Example 4. Fig. 3 shows two types of reduced BIST-able RTL circuit. Fig. 3(b) shows reduced BIST-able with a MISR for each loop and Fig. 3(c) shows reduced BIST-able with a transparent MISR. Note that, the original register R2 is modified into MISR

register T5 and original register R3 is modified into MISR T6 in Fig. 3(b). We also can insert only a transparent MISR T7 at the sharing path of both loops to make the circuit self-testable as shown in Fig. 3(c). Regarding the cost, we choose the lower one between these two types.

Concurrent BIST-able RTL and reduced BIST-able RTL have different ways to break each loop in the circuit. Concurrent BIST-able RTL circuit breaks a loop with CBILBO while reduced BIST-able RTL circuit breaks a loop with a MISR. The cost of CBILBO is more expensive than the cost of MISR. Since we can reduce the cost and area overhead by using a MISR, so we call our approach reduced BIST-able RTL circuit. We also can get shorter test application time because we test all combinational blocks simultaneously and get comparable high fault coverage likes approach in [2].

3. DFT for Reduced BIST-able RTL circuit

This section describes the DFT method that augments a given RTL into a reduced BIST-able RTL circuit. In the DFT, primary inputs and primary outputs are modified into TPGs and RAs respectively.

3.1 Problem Formulation

A reduced BIST-able RTL circuit has a MISR in each loop. The DFT for the reduced BIST-able RTL circuit is formalized as the following optimization problem.

- Input : a given RTL circuit
- Output : a reduced BIST-able RTL circuit
- Optimization : minimizing area overhead

3.2 DFT algorithm

In order to augment a given RTL circuit into reduced BIST-able RTL circuit, we introduce an extended R-graph and an extended minimum feedback vertex set. We first look at definition of R-graph of ADD.

Definition 5. An R-graph of an ADD is a directed graph $G=(V,A,w,)$ that has the following properties.

i. $v \in V$ is a read node or write node. If a read node and a write node correspond to the same variable, they are represented by the same vertex;

ii. $(v_i, v_j) \in A$ denotes an arc if there exists a path from the read node v_i to the write node v_j;

iii. $w: V \rightarrow Z^+$ (the set of positive integers) defines the size of read or write node corresponding to a vertex in V;

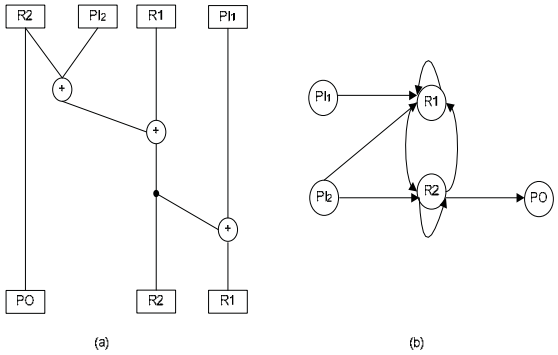


Fig. 4 (a) ADD for original RTL circuit S1 (b) R-graph for ADD S1

Example 5. Fig. 4(a) shows the R-graph of S1. Each read node and write node S1 are represented by input vertex and output vertex respectively in R-graph. Each path from read node to write node in ADD is represented by the arc from input vertex to output vertex in R-graph.

We define an extended R-graph as follows. An extended R-graph has an extra vertex called dummy vertex D. This dummy vertex represents an extra register as transparent register. These steps below are taken to construct an extended R-graph.

1. Transform an RTL circuit into an ADD representation.
2. Find the path such that dummy D is added into that path. Dummy D is added if there is a path between 2 operator nodes or a path between an operator node and a fan out node.
3. Construct an extended R-graph by replacing the read node, write node and dummy node in ADD into input vertex, output vertex and dummy vertex respectively in extended R-graph.

Definition 6. An extended R-graph of an ADD is a directed graph $G=(V,A,w,)$ that has the following properties:

i. $v \in V$ is a read node, write node or dummy node. If a read node and a write node correspond to the same variable, they are represented by the same vertex;

ii. $(v_i, v_j) \in A$ denotes an arc if there exists a path from the read node v_i to the write node v_j,

iii. $(v_i, v_n) \in A$ denotes an arc if there exists a path from the read node v_i to the dummy v_n

iv. $(v_i, v_m) \in A$ denotes an arc if there exists a path from the dummy v_i to the dummy v_k

v. $(v_i, v_s) \in A$ denotes an arc if there exists a path from the dummy v_i to the write node v_s

vi. $w: V \rightarrow Z^{+}$ (the set of positive integers) defines the size of read, write node or dummy node corresponding to a vertex in V;

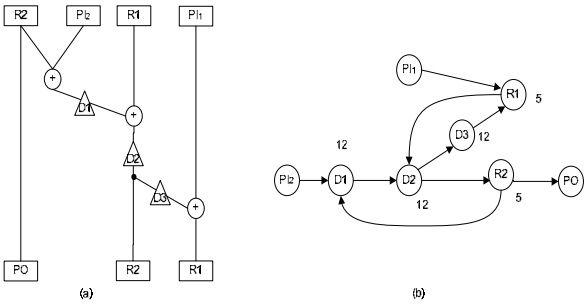


Fig. 5 (a) ADD for modified original circuit with dummy S2 (b) An extended R-graph for S2

Example 5. Fig. 5(b) shows an extended R-graph for S2. An extended R-graph has a cost for each vertex except the primary input and primary output vertex. In other words, cost means the area overhead of each vertex. The original register vertex represent as R1 and R2 and the dummy vertex represent as D1 until D3. In this case, since we modify the original register into MISR or transparent MISR to make this original circuit self-testable, so the each cost of R1 and R2 are 5 and the each cost of D1, D2 and D3 is 12. Note that, S2 modified with dummy D1, D2 and D3 in some paths. For example, D1 is added because there is a path between 2 operation nodes.

Definition 7. Minimum feedback vertex set is a set of vertices, V such that:

i. The number of vertices is minimized.
ii. Every loop is broken by a vertex in a set of vertices, $v \in V$.

Minimum feedback vertex set considers only the minimum number of vertex set that breaks every loop.

Example 6. Refer to Fig. 4(b), the minimum feedback vertex set is {R1, R2} because when R1 and R2 removed from the circuit, every loop is broken. Moreover, the circuit becomes acyclic when removing R1 and R2.

We define an extended minimum feedback vertex set so that the extended minimum feedback vertex considers the cost of feedback vertex set in our reduced BIST-able RTL circuit.

Definition 8. An extended minimum feedback vertex set is a set of vertices V such that:

i. Every loop is broken by a vertex in a set of vertices, $v \in V$.
ii. The summation cost of set of vertices that break a loop is minimized $\sum_{v \in V} \text{cost}(v)$.

By introducing the extended minimum feedback vertex set, we consider the minimized cost of summation vertices set. Cost of summation vertices set is a total area overhead of vertices in a set of vertices. The set of vertices are divided into two sets. There are set of dummy vertex and set of original register vertex. For each vertex in the extended R-graph, we define cost p(v) for dummy vertex and original register vertex as follows.

i. Each cost of original register vertex $p(v) = 5w_r$
ii. Each cost of dummy vertex $p(v) = 12w_d$

We total up the cost of vertex in the dummy vertex set and original register dummy set and find the minimum total cost for both vertex sets. Then, we choose the minimum one. If the minimum summation cost belongs to an original register vertex set, therefore, we replace each vertex of that sets with MISR. However if the minimum summation cost belongs to a dummy vertex set, we replace each vertex of that sets with transparent MISR.

Example 7. Refer to Fig. 5(b), we find the two sets of feedback vertex set. There are original register vertex set and dummy vertex set. The minimum feedback vertex sets for original register vertex is {R1,R2} and the minimum feedback vertex sets for dummy vertex is {D2}. Since the summation cost of {R1,R2} is 10 where the cost of each R1 and R2 is 5 and the summation cost of {D2} is 12, we choose the set of {R1,R2} as extended minimum feedback vertex set because the summation cost of the set of {R1,R2}is lower than and the summation cost of {D2}. Therefore, we modify the RI and R2 into MISR to make the circuit testable.

4. CASE STUDIES

4.1 Experiment Setup

In the case studies, we evaluated the effectiveness of the proposed method by performing fault simulation on the circuits augmented with the proposed BIST method. Experiments have been conducted on ITC'99 benchmark circuits where the behavioural descriptions are given. We compared reduced BIST-able RTL circuit with primitive BIST-able RTL circuit and concurrent BIST-able RTL circuit

(Wunderlich's method [2]). Table 1 presents the characteristic of the benchmark circuits. In Table 1, #FF represents the number of flip-flops while PI/PO represents the number of inputs/outputs of the circuit. The area column represents the area of the circuit where one unit of area is equal to the size of an inverter.

Table 1. Characteristics of the benchmark circuits

ITC'99	# Flip-flops	Area	IO pins	
			PI	PO
b01	5	90	4	2
b02	4	63	3	1
b03	30	422	6	4
b06	9	138	4	6
b09	28	391	3	1
b10	17	341	13	6

4.2 Experimental Results

We evaluated the effectiveness of the circuits in terms of fault coverage, area overhead and test application time. We compared our method with concurrent BIST-able RTL circuit and primitive BIST-able RTL circuit. Figs. 6-12 show the results of fault coverage versus clock cycles. Since the test patterns generated by MISR for a combinational block partly depend on the test responses from the preceding combinational block, some of the random test patterns cannot be generated. Because there are not many useful random test patterns missed by the MISR, the fault coverage of our method is just slightly lower than that of concurrent BIST-able RTL circuit. For benchmark circuit such as b02, our method obtains 88% while 90% using concurrent BIST-able RTL circuit after 100 clock cycles. We determine the clock cycles first then find out what fault coverage based on that clock cycles.

Table 2 represents the percentage of the area overhead compared to the area of primitive BIST-able RTL circuit. Our method shows that all the benchmark circuits have lower area overhead compared to concurrent BIST-able RTL circuit. Refer to circuit b02 in Table 2, only one register called stato that has 3 bits is modified into CBILBO and MISR. By modifying register stato into CBILBO and MISR, it needs 51% area overhead and 22% area overhead respectively. We modify the original register into MISR at behavioural descriptions to make RTL circuit BIST-able such that the area overhead becomes low.

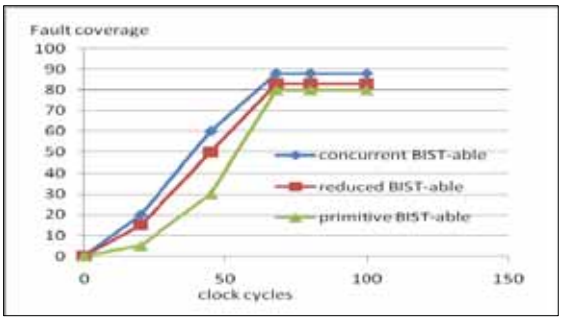


Fig. 6 b01 circuit

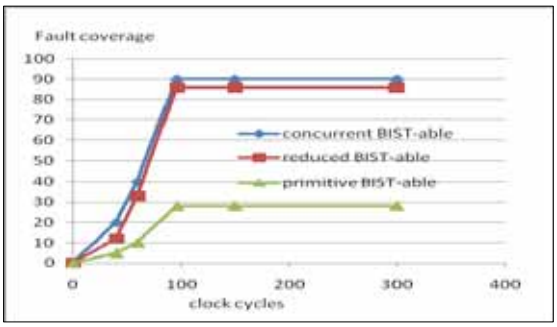


Fig. 7 b02 circuit

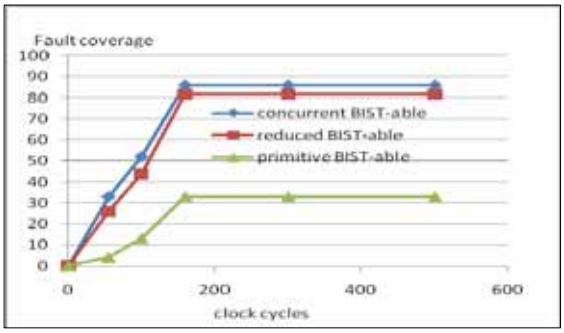


Fig. 8 b03 circuit

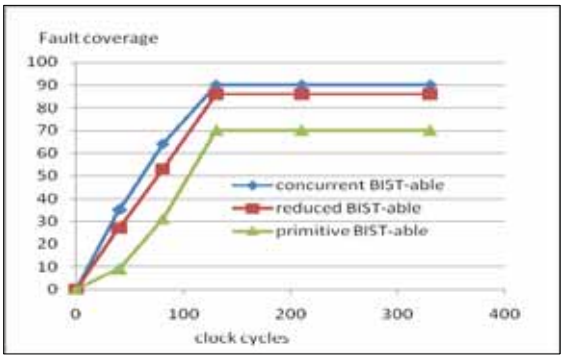


Fig.9 b06 circuit

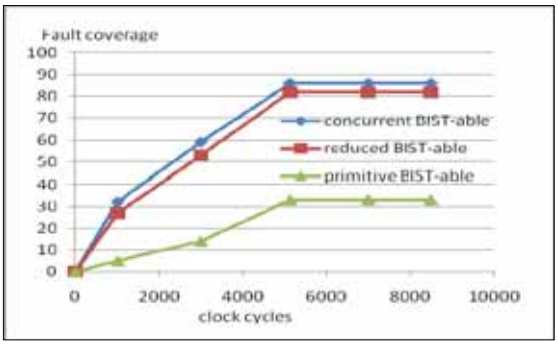


Fig.10 b09 circuit

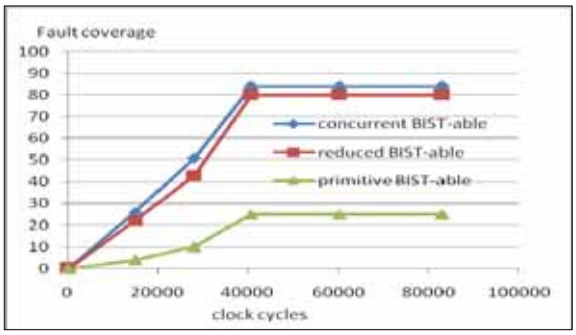


Fig.11 b10 circuit

5. Conclusions

A new BIST technique for test-per-clock scheme has been introduced in this paper. A given RTL circuit augmented by our DFT method is called reduced BIST-able RTL circuit. We modify the original register into MISR at behavioural descriptions to make RTL circuit BIST-able. Our method shows high fault coverage and lower area overhead for the ITC'99 benchmark circuits compared to concurrent BIST-able RTL circuit and primitive BIST-able RTL circuit.

Table 2. Area overhead

ITC'99	Primitive BIST-able	Concurrent BIST-able [2]	Reduced BIST-able [proposed method]
	Area (OH%)	Area (OH%)	Area (OH%)
b01	90(0%)	154 (71%)	107 (19%)
b02	63(0%)	95 (51%)	77 (22%)
b03	422(0%)	589 (40%)	449 (6%)
b06	138(0%)	208 (51%)	161 (17%)
b09	391(0%)	612 (75%)	437 (12%)
b10	341(0%)	383 (12%)	359 (5%)

REFERENCES

[1] M. Abramovici, M. A. Breuer, and A.D.Friedma: *Digital Systems Testing and Testable Design*, Computer Science Press (1990).

[2] A. P. Stroele and H. J. Wunderlich: "Hardware-optimal test register insertion," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 17(6):531–539 (1998).

[3] T.Masuzawa, M.Idutsu, H.Wada,and H.Fujiwara: "Single-control testability of RTL data paths for BIST," Proc. 9th *Asian Test Symposium*, pp.210-215(2000).

[4] K. Yamaguchi, H.Wada, T. Masuzawa, and H.Fujiwara:"A BIST Method Based on Concurrent Single-Control testability of RTL data paths for BIST," *Proc. 10th Asian Test Symposium*, pp.313-318(2001).

[5] Kenichi Yamaghuci, Michiko Inoue, Hideo Fujiwara: "Test-per-clock BIST with low overhead," 2007 Wiley Periodicals, Inc. Electronics and Communications in Japan Pt 2, volume 90, Issue 6, pp.47–58, (2007).

[6] B.Koenemann, J.Mucha,and G.Zwiehoff:"Built-in logic block observation techniques*," In IEEE Test Conf*., pp.37-41(1979).

[7] I. Ghosh, N. K. Jha, and S. Bhawmik: "A BIST scheme for RTL controller-data paths based on symbolic testability analysis," *In Proc. Design Automation Conf*., pp. 554–559 (1998).

[8] K. Kim, D.S. Ha and J.G. Tront, "On using signature registers as pseudorandom pattern generators in built in self testing," *IEEE Trans. Computer-Aided Design*,pp, 919-928, August 1988.

A Binding Method for Hierarchical Testing Using the Results of Test Environment Generation

Hiroaki FUJIWARA† Toshinori HOSOKAWA‡ Ryoichi INOUE† Hideo FUJIWARA††

†Graduate School of Industrial Technology, Nihon University, 1-2-1 Izumicho, Narashino, Chiba 275-8575, Japan

‡College of Industrial Technology, Nihon University, 1-2-1 Izumicho, Narashino, Chiba 275-8575, Japan

††Faculty of Informatics, Osaka Gakuin University, 2-36-1 Kishibe-Minami, Suita-shi, Osaka, Japan

E-mail: †cihr10016@g.nihon-u.ac.jp, ††tukinokaze@nifty.com,
‡hosokawa.toshinori@nihon-u.ac.jp, ‡‡fujiwara@ogu.ac.jp

Abstract The recent advances in semiconductor technology have resulted in an exponential increase in VLSI circuit density and complexity. Therefore, it is difficult to generate test sequences for sequential circuits using gate-level sequential ATPGs. Hierarchical test generation methods that generate test environments for each functional element at the functional register transfer level and test patterns for each combinational module at the gate level have been proposed to accelerate sequential test generation. In this paper, we propose a binding for testability method that preferentially assigns operations with test environments and operations without test environments to the same operational unit to reduce the number of hard-to-test operational units. The experimental results show the effectiveness of the proposed method.

Keywords hierarchical test generation, behavioral synthesis, test environment, binding for testability, functional register transfer level

1. Introduction

The recent advances in semiconductor technology have resulted in an exponential increase in very-large-scale integration (VLSI) circuit density and complexity. Effective and efficient test generation algorithms for combinational circuits have been proposed. These algorithms can achieve high fault efficiency even for large circuits [1-5]. However, it is very difficult to achieve high fault efficiency for sequential circuits in a reasonable test generation time by a conventional gate-level sequential automatic test pattern generator (ATPG).

Generally, the current large-scale integrated circuits (LSI) design flow starts from the register transfer level (RTL) design. Design productivity decreases as LSI design costs increase [6]. Therefore, a new design methodology that uses behavioral synthesis to describe the LSI behavior and synthesize RTL circuits is gathering attention.

Because the amount of LSI behavioral descriptions is less than that of RTL descriptions, designing at the behavioral level can improve design productivity. Because behavioral synthesis assigns resources such as registers and operational units, the synthesis performance greatly influences the performance of the synthesized circuits [6].

Behavioral synthesis primarily performs scheduling [6] and binding [6]. Scheduling assigns execution times to operations and binding assigns registers and operation units to variables and operations. Synthesized cycle-accurate circuits after scheduling are said to be functional RTL circuits. Synthesized circuits after scheduling and binding are said to be structural RTL circuits. Functional RTL circuits and structural RTL circuits are synthesized to gate-level circuits by logic synthesis.

Several design-for-testability (DFT) methods based on hierarchical test generation [7] for structural RTL circuits have been proposed [8-10]. In references [8-10], a structural RTL circuit consists of a data-path part and a controller part, both of which are isolated in the test modes. Separate DFT methods have been proposed for the data-path part and the controller part. Therefore, the area overhead for testing becomes larger. Hierarchical test generation methods for a stuck-at fault model for functional RTL circuits have been also proposed [11-13]. In these methods, a given functional RTL circuit is first described in an assignment decision diagram (ADD) [14]. Then, test environments are generated for each

operation. A test sequence is formed by substituting the corresponding test patterns of the gate-level circuit to the operation in the test environments. Compared with a gate-level ATPG, these methods can drastically reduce the test generation time without sacrificing fault coverage [11-13]. These methods perform binding by logic synthesis. Therefore, if the test environment coverage, which is the ratio of the number of nodes successfully generating test environments to the total number of nodes, is low, it is not guaranteed that the generated test sequences can achieve high fault coverage. In this paper, we propose a binding for testability method that preferentially assigns operations with test environments and operations without test environments to the same operational unit to reduce the number of hard-to-test operational module units.

The outline of this paper is as follows. Section 2 explains ADD. Section 3 describes behavioral synthesis. Section 4 describes hierarchical test generation. In Section 5, a binding for testability method is proposed. The experimental results are shown in Section 6. Finally, Section 7 concludes this paper.

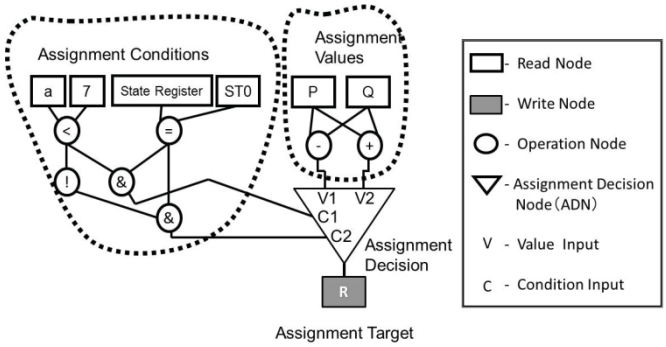


Fig. 1 Example of ADD

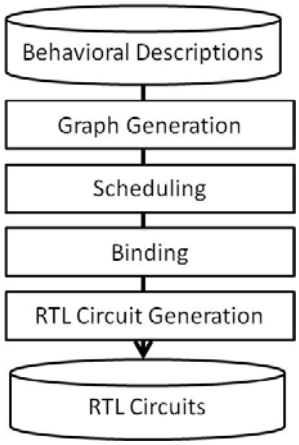


Fig. 2 Flow of behavioral synthesis

2. Assignment Decision Diagram

An ADD is an acyclic graph that consists of a set of nodes categorized into the following four types: read node, write node, operation node, and assignment decision node (ADN), as shown in Fig. 1. In addition, the ADD has a set of edges containing the connectivity information between two nodes. (Fig. 1). The read node represents a primary input port, a storage unit, or a constant. The write node represents a primary output port or a storage unit. The operation node expresses an arithmetic operation unit or a logic operation unit, and the ADN selects a value from a set of values provided to it based on the conditions computed by the logic operation units. If one of the condition inputs becomes true, the value of the corresponding data input is selected. Although ADD was essentially introduced as an internal representation in the behavioral synthesis process, it can be used to describe a functional RTL circuit, the controller part and the data-path part, which are homogeneously represented.

3. Behavioral Synthesis

Behavioral synthesis generates RTL circuits using hardware description languages such as VHDL and Verilog-HDL from behavioral descriptions using, for example, C language and System C. Behavioral synthesis consists of four steps: graph generation, scheduling, binding, and RTL circuit generation, as shown in Fig. 2. In the graph generation step, graphs such as the ADD [14] or the Control/Data-Flow Graph (CDFG) [6] are generated from the given behavioral descriptions. In the scheduling step, each operation is assigned to a time slot corresponding to a clock cycle or time interval. In the binding step, operational units (hardware components) and registers are assigned to operations and variables, respectively. In the RTL circuit generation step, the data-path part and the controller part are generated.

In behavioral synthesis using ADD, for example, scheduling and binding are implemented by transforming the graph structures. In this paper, an ADD after scheduling is said to be a functional ADD, and an ADD after binding is said to be a structural ADD.

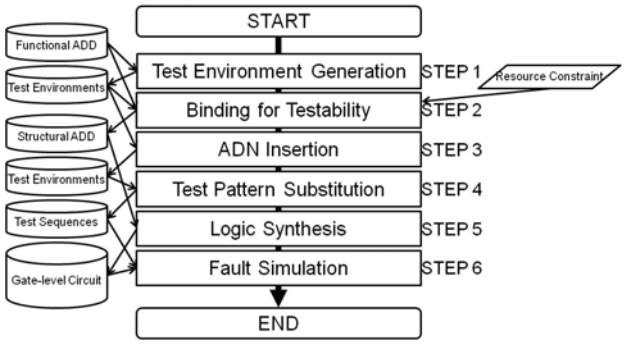


Fig. 3 Flow of Test Generation

4. Hierarchical Test Generation Based on a Functional ADD

4.1. Test Environment Generation

Hierarchical test generation for a functional ADD (i.e., after scheduling) is explained as follows. When a node *N* is being tested, the testability of the node is guaranteed if any value can be propagated from a read node corresponding to a primary input port to the input of *N*, and the value at the output of *N* can be propagated to a write node corresponding to a primary output port. The path allowing any value to be propagated from a read node corresponding to a primary input port to the input of *N* is called a justification path, and the path allowing the value at the output of *N* to be propagated to a write node corresponding to a primary output port is called the propagation path. In this paper, justification and propagation are done through symbolic processing that utilizes 9-valued algebra [11]. The series of symbols obtained from the symbolic processing that activates justification and propagation paths is known as the test environment for the node under test. For a given node under test, its test sequence is generated by first extracting a test pattern from the test set library and substituting the test pattern for the test environment. The test set library is obtained beforehand by first taking a gate-level circuit whose functionality is the same as that of the node under test, then generating the test patterns for all faults in the circuit using a combinational ATPG algorithm. In the case that the node is synthesized to a different circuit, fault simulation must be performed to check the fault efficiency of the test patterns.

4.2. Problem of Hierarchical Test Generation Based on a Functional ADD

If the test environment coverage [12-13], which is the ratio of the number of nodes successfully generating the test environments to the total number of nodes, is high, fault coverage at the gate level is also high. However, high fault coverage is not necessarily obtained compared to gate-level sequential ATPG methods, even if hierarchical test generation methods [11-13] are used. The reason is described as follows. In references [11-13], binding processing was not taken into account. The binding processing of logic synthesis may assign operation nodes with no test environments generated for the same operational unit. In this case, the operational unit does not have test environments. Therefore, we consider that because faults in the operational unit are not detected by the generated test sequences, fault coverage is not high. To resolve this problem, we propose a binding for testability method which preferentially assigns operations with test environments and operations without test environments to the same operational unit.

5. Binding for Testability

5.1. Test Design Flow

The procedure of the proposed test generation is explained. The flow of the test generation is shown in Fig. 3.

(STEP 1) Test environments are generated for a given functional ADD by the method shown in reference [11].

(STEP 2) The binding for testability is performed for a functional ADD under the constraint of the number of operational units. The binding for testability method is described in Section 5.3 in detail. A structural ADD is generated by the binding.

(STEP 3) ADNs are inserted into an ADD when the binding is performed for a functional ADD. Therefore, test environments for the inserted ADNs are generated.

(STEP 4) Test sequences are generated by first extracting test patterns from a test set library and by substituting these test patterns for the test environments.

(STEP 5) Logic synthesis is performed for the structural ADD generated in STEP 2 and a gate-level circuit is

generated.

(STEP 6) Fault simulation is performed for the gate-level circuit by the test sequences generated in STEP 4, and the fault coverage is calculated.

5.2. Problem Formulation

(Definition 1: Functional RTL test environment coverage)

$$RTE_F = \frac{\#N_{TE}}{\#N_{all}}$$

Functional RTL test environment coverage (RTE_F) is defined as the ratio of the number of nodes successfully generating test environments ($\#N_{TE}$) to the total number of nodes under test in a functional ADD ($\#N_{all}$). In this paper, the nodes under test are only the operation nodes.

(Definition 2: Structural RTL test environment coverage)

$$RTE_S = \frac{\#M_{TE}}{\#M_{all}}$$

Structural RTL test environment coverage (RTE_S) is defined as the ratio of the number of modules successfully generating test environments ($\#M_{TE}$) to the total number of modules under test in a structural ADD ($\#M_{all}$). In this paper, the modules under test are only the operational units.

(Problem formulation)

Inputs: Nodes under test in a functional ADD (e_1, …, e_n), where n is the number of nodes under test and each node has the information of whether a test environment is generated.

Outputs: The binding information for each module (B_1, …, B_m), where m is the number of modules and each binding information has assigned nodes.

Optimization: Maximize the structural test environment coverage.

Constraint: The number of operational units.

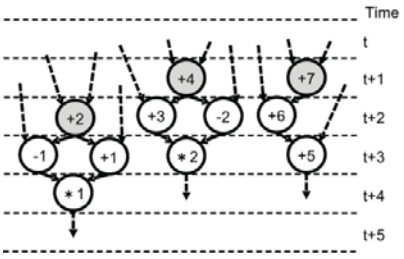


Fig. 4 An example of scheduling

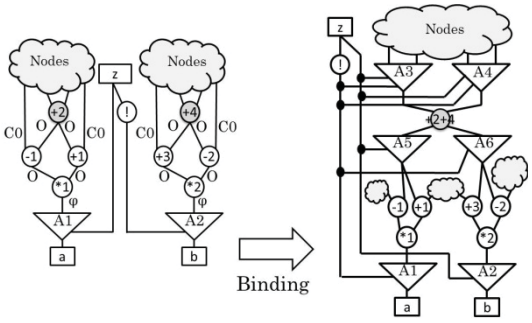


Fig. 5 The binding result (sharing +2 and +4)

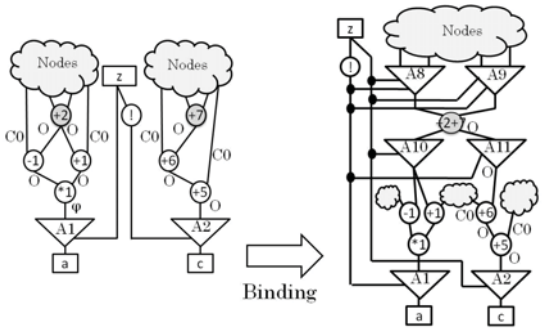


Fig. 6 The binding result (sharing +2 and +7)

```
1:  Binding_for_testability (an operation set OP) {
2:    for(each operation Ni with a test environment){
3:      for(each operation Nj without a test environment){
4:        if(Ni and Nj mergeable) {
5:          merge Ni and Nj to a new operation Nnew;
6:          add Nnew to OP;
7:          delete Ni and Nj from OP;
8:          break;
9:        }
10:     }
11:   }
12:   for(each operation Ni){
13:     for(each operation Nj except Ni){
14:       if(Ni and Nj mergeable){
15:         merge Ni and Nj to a new operation Nnew;
16:         delete Ni and Nj;
17:         add Nnew to OP;
18:         break;
19:       }
20:     }
21:   }
22:   assign each node in OP to each operational unit;
23: }
```

Fig. 7 Binding for testability algorithm

5.3. Binding for Testability Algorithm

The proposed binding algorithm for testability is explained using an example. Fig. 4 shows a scheduling example. Fig. 5 shows the binding results in which the add operation nodes +2 and +4 are assigned to the same adder unit for Fig. 4. Fig. 6 shows the binding results in which the add operation nodes +2 and +7 are assigned to the same adder unit for Fig. 4. In the test environment generations for +2 and +4 of Fig. 4, because the symbol for fault effects O [11] cannot be propagated to the outputs of the multiplier nodes *1 and *2, the test environments cannot be generated. The test environment for +7 can be generated. When +2 and +4 are assigned to the same adder unit, the test environment of the adder unit does not exist and the adder unit cannot be tested (Fig. 5). In contrast, when +2 and +7 are assigned to the same adder unit, the test environment of the adder unit exists and the adder unit can be tested. In this paper, we propose a binding for testability method which preferentially assigns operations with test environments and operations without test environments to the same operational unit. Our goal is to improve structural test environment coverage while satisfying the resource constraints.

Fig. 7 shows the algorithm of binding for testability. Explanations for each line of code are given in the following.

(line 1) An operation set *OP* is given. The elements of the *OP* are operation nodes. Each operation node has the information of whether a test environment has been generated.

(line 2) Lines 3 to 9 are iterated for each operation N_i with a test environment.

(line 3) Lines 4 to 8 are iterated for each operation N_j without a test environment.

(line 4) If N_i and N_j are able to be merged, lines 5 to 8 are performed.

(line 5) N_i and N_j are merged to a new operation N_{new}.

(line 6) N_{new} is added to the *OP*.

(line 7) N_i and N_j are deleted from the *OP*.

(line 8) The algorithm returns to line 2.

(line 12) Lines 13 to 19 are iterated for each operation N_i in the *OP*.

(line 13) Lines 14 to 18 are iterated for each operation N_j except N_i.

(line 14) If N_i and N_j are able to be merged, lines 15 to 18 are performed.

(line 15) N_i and N_j are merged to a new operation N_{new}.

(line 16) N_{new} is added to the *OP*.

(line 17) N_i and N_j are deleted from the *OP*.

(line 18) The algorithm returns to line 12.

(line 22) Each node in the *OP* is assigned to an operational unit.

6. Experimental Results

The test environments and the test sequences using the method shown in reference [11] were generated for Paulin, Diffeq, and DCT, which are functional RTL circuits. The binding for testability method proposed in this paper was applied to the circuits. For comparison, the left-edge algorithm (LEA) [15] was also applied to the circuits.

Table 1 shows the results of the test environment generation. In Table 1, “Circuits” denotes the names of the circuits, “With TE” denotes the operation nodes with test environments, and “Without TE” denotes the operation nodes without test environments. Table 2 shows the results of the two algorithms for binding the circuits. In Table 2, “Operations” denotes the types of operational units, “Proposed” denotes the names of the operation nodes assigned to the operational units using the proposed binding method, and “LEA” denotes the names of the operation nodes assigned to the operational units using the LEA. Table 3 shows the functional RTL test environment coverage (“RTE_F”) and the structural RTL test environment coverage (“RTE_S”). In Table 3, “Proposed” denotes the structural test environment coverage using the proposed binding method, and “LEA” denotes the structural test environment coverage using the LEA. The functional RTL test environment coverage was 70% to 76%. The structural RTL test environment coverage using the proposed binding method was 100%. In contrast, the structural test environment coverage using the LEA was only 75% to 80%.

After the test sequence generation and binding, the gate-level circuits were generated by logic synthesis. Fault simulations for single stuck-at faults in only the operational units were performed for the gate-level circuits by the generated test sequences. Table 4 shows the fault coverage, the execution time for the fault simulation, and the fault efficiency. In Table 4, “Bit width” denotes the bit widths of the data path in the circuits, and “Execution Time” denotes the execution time for the fault simulation. Untestable faults were

identified using the gate-level sequential test generation tool "TetraMAX™" from Synopsys. Thus, the fault efficiencies were calculated.

The fault coverage for the circuits generated by our proposed binding method increased by 4.22% on average compared with that generated by the LEA. The proposed binding for testability method effectively improved the structural test environment coverage and the fault coverage.

The fault coverage for DCT was low compared with that of the other circuits. Because DCT has many multipliers with constant input, many untestable faults in the gate-level circuit were identified by the TetraMAX ATPG.

Table 1 Experimental Results of Test Environment Generation

Circuits	With TE	Without TE
Paulin	+1, +2, -2, *1, *2, *4, *6	-1, *3, *5
Diffeq	+1, +2, -2, *1, *2, *4, *6, <	-1, *3, *5
DCT	+1, +2, +3, -1, -2, *1, *2, *3, *4, *5	+4, +5, +6

Table 2 Experimental Results of Binding

Circuits	Operations	Proposed	LEA
Paulin	MUL1	*3,*6	*3,*5
	MUL2	*1,*2,*4,*5	*1,*2,*4,*6
	ADD	+1,+2	+1,+2
	SUB	-1,-2	-1,-2
Diffeq	MUL1	*4,*5	*3,*5
	MUL2	*1,*2,*3,*5	*1,*2,*4,*6
	ADD	+1,+2	+1,+2
	SUB	-1,-2	-1,-2
	LES	<	<
DCT	MUL1	*1,*2,*3	*1,*2,*3
	MUL2	*4,*5	*4,*5
	ADD1	+1,+3,+4,+6	+1,+3,+3,+5
	ADD2	+2,+5	+4,+6
	SUB	-1,-2	-1,-2

Table 3 Experimental Results of Test Environments Coverage

Circuits	RTE_F (%)	RTE_S (%)	
		Proposed	LEA
Paulin	70.00 (7/10)	100.00 (4/4)	75.00 (3/4)
Diffeq	72.73 (8/11)	100.00 (5/5)	80.00 (4/5)
DCT	76.92 (10/13)	100.00 (5/5)	80.00 (4/5)

Table 4 Experimental Results of Fault Coverage

Bit width	Circuits	Fault Coverage (%)		Fault Efficiency (%)		Execution Time (s)	
		Proposed	LEA	Proposed	LEA	Proposed	LEA
4	Paulin	100.00	89.65	100.00	89.65	0.03	0.02
	Diffeq	96.58	96.58	96.85	96.58	0.04	0.03
	DCT	65.17	62.85	95.68	95.51	0.03	0.04
8	Paulin	100.00	80.05	100.00	80.05	0.26	0.26
	Diffeq	98.16	98.16	98.16	98.16	0.28	0.28
	DCT	60.19	59.06	99.57	98.62	0.11	0.12
16	Paulin	99.17	94.97	99.17	94.97	3.91	3.80
	Diffeq	98.96	98.96	98.96	98.96	5.27	5.12
	DCT	51.26	51.26	99.73	99.56	0.64	0.75

7. Conclusion

In this paper, we proposed a binding for testability method which preferentially assigns operations with test environments and operations without test environments to the same operational unit to improve the structural test environment coverage. The experimental results showed that both the structural test environment coverage and the fault coverage of the proposed binding method increased compared with those of LEA without considering the test environment. In future work, register binding for hierarchical testability will be examined.

References

[1] M. Schulz, E. Trischler and T. Sarfert, "SOCRATES: A Highly Efficient Automatic Test Pattern Generation System," IEEE Trans. on Computer-Aided Design, Vol. 7, No. 1, pp. 126-137, Jan. 1988.

[2] W. Kunz and D. Pradhan, "Recursive Learning: An Attractive Alternative to the Decision Tree for Test Generation in Digital Circuits," in Proc. IEEE International Test Conference on Discover the New World of Test and Design, pp. 816-825, 1992.

[3] M. Henftling, H. C. Wittmann and K. J. Antreich, "A Single-Path-Oriented Fault-Effect Propagation in Digital Circuits Considering Multiple-Path Sensitization," in Proc. 1995 IEEE/ACM International Conference on Computer-Aided Design, pp. 304-309, 1995.

[4] C. Wang, S. Reddy, I. Pomeranz, X. Lin and J. Rajski, "Conflict Driven Techniques for Improving Deterministic Test Pattern Generation," in Proc. IEEE International Conference on Computer-Aided Design, pp. 87-93, 2002.

[5] E. Gizdarski and H. Fujiwara, "SPIRIT: A Highly Robust Combinational Test Generation Algorithm," IEEE Trans. on Computer Aided Design for Integrated Circuits and Systems, Vol. 21, No. 12, pp. 1446-1458, Dec. 2002.

[6] D. D. Gajski, N. D. Dutt, A. C-H Wu and S. Y-L Lin, HIGH-LEVEL SYNTHESIS: Introduction to Chip and System Design, Kluwer Academic Publisher, 1992.

[7] B. T. Murray and J. P. Hayes, "Hierarchical Test Generation Using Pre Computed Tests for Modules," IEEE Trans. Computer-Aided Design, Integrated Circuits & Syst., Vol. 9, No. 6, pp. 594-603, Jun. 1990.

[8] I. Ghosh, A. Raghunathan and N. K. Jha, "Design for Hierarchical Testability of RTL Circuits Obtained by

Behavioral Synthesis," IEEE Trans. Computer-Aided Design for Integrated Circuits and Systems, Vol. 16, pp. 1001-1014, Sep. 1997.

[9] 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 (JETTA), Vol. 16, No. 5, pp. 553-566, Oct. 2000.

[10] H. Wada, T. Masuzawa, K. K. Saluja and H. Fujiwara, "Design for Strong Testability of RTL Data Paths to Provide Complete Fault Efficiency," in Proc. of 13th International Conf. on VLSI Design, pp. 300-305, Jan. 2000.

[11] I. Ghosh and M. Fujita, "Automatic Test Pattern Generation for Functional RTL Circuits Using Assignment Decision Diagrams," in Proc. of ACM/IEEE Design Automation Conference, pp. 43-48, Jun. 2000.

[12] L. Zhang, I. Gosh and M. Hsiao, "Efficient Sequential ATPG for Functional RTL Circuits," in Proc. International Test Conference, pp. 290-298, Sep. 2003.

[13] H. Fujiwara, C. Y. Ooi and Y. Shimizu, "Enhancement of Test Environment Generation for Assignment Decision Diagrams," 9th IEEE Workshop on RTL and High Level Testing (WRTLT'08), pp. 45-50, Nov. 2008.

[14] V. Chaiyakul, D. D. Gajski and L. Ramachandran, "High-Level Transformations for Minimizing Syntactic Variances," in Proc. Design Automation Conference, pp. 413-428, Jun. 1993.

[15] F. J. Kurdahi and A. C. Parker, “REAL: A Program for Register Allocation,” in Proc. Design Automation Conference, pp. 210-215, Jun. 1987.

Testability Challenges of SoC with Integrated Power Management

Unified Top Level Design, Verification and Test Methodology

Lakshmanan Balasubramanian
MCU,
Texas Instruments (India) Pvt. Ltd.,
Bengaluru, India.
lakshmanan@ti.com

Rajesh Kumar Mittal
WCS,
Texas Instruments (India) Pvt. Ltd.,
Bengaluru, India.
rajmittal@ti.com

Abstract – Power management (PM) integration to a mixed-signal SoC causes significant challenges in initialization, test implications of power-up sequencing, specification testing and calibration of integrated PM modules like a low drop out regulator (LDO) and bandgap reference (BG). The scan patterns (ATPG) are also impacted due to PM and analog integration. Requirement of isolation of signals from different voltage and power domains in analog modules is important for IDDQ testing. Further the current non-standard and ad-hoc verification of the PM module aggravates the problem. All the verification is not feasible in mixed mode simulation as it is time consuming process. This paper discusses DFT techniques to test these modules while providing cost effective solution and provide right verification methodology with analog behavioural models (BMOD). The paper also discusses real issue observed in scan pattern which was caused by the complexity of PM integration.

Keywords – Power management; DFT; LDO; BG; Trim; ATPG.

I. INTRODUCTION – POWER MANAGEMENT INTEGRATION AND ASSOCIATED TEST CHALLENGES

It is becoming a necessity to integrate the complete power management (PM) system including its core analog functionalities into SoC to stay competitive especially in terms of the simplicity of the platform solution (avoiding several board level components), cheaper solution cost, higher flexibility in driving finer system power optimization, and above all the power and ground isolation needs of analog and RF integration [1][2].

In spite of the aforementioned necessities and benefits of PM integration, it brings with it additional complexities which have specific impact on the system performance, test time and hence test cost, verification complexities, new DFT requirements and even unexpected complexities to the traditional scan based DFT approaches [3][4].

The block diagram of a typical power management (PM) system integrated into an embedded system is shown in Figure 1. Key components of the PM system include bandgap (BG) reference circuit for all the voltage and current reference requirements of the analog circuits in the entire SoC, independent power supplies for analog and digital sections of the SoC and a power-up sequencer which generates a power-on reset signal (POR) for the entire SoC.

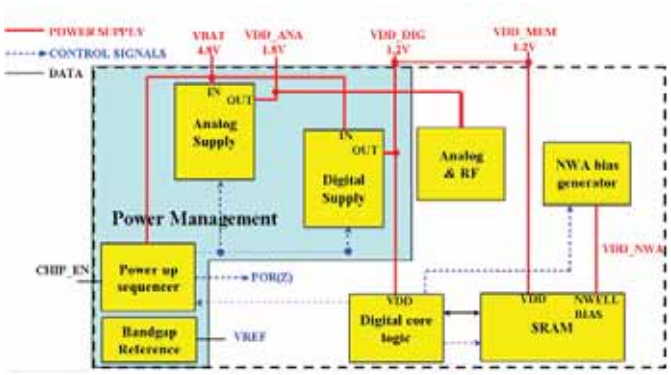


Figure 1 Typical power management system block diagram in the context of an embedded system

This paper is organized into 4 sections. Section II describes the additional testability requirements of on-chip power supplies and various methodology components to deal with the complexities. Section III describes the necessity and complexity of trimming critical analog parameters and various methods to deal with it. Section IV describes the implications of PM integration on traditional scan based DFT techniques and methods to circumvent the problems. Section V describes the design and verification methodologies employed to enable efficient PM integration and testing. Section VI presents some results along with potential benefits for schemes under implementation. Finally Section VII concludes the paper with a detailed summary of problems, proposed solution, their impact and directions towards further improvements.

II. TESTING ON-CHIP POWER SUPPLIES

The on-chip power supplies, designed as either linear regulators (LDO) or switching voltage regulators (either capacitive or inductor based), provide a voltage source with a wide but limited range of current i.e., power supply capacity. The magnitude of the voltage level of the power supply is designed to be a scaled factor of an accurate and constant voltage reference. As is well known the dynamic power consumption of the digital circuits is directly proportional to the square of the supply voltage level. The static power consumption also depends on the supply voltage level. Further

the speed performance of the digital circuits is inversely proportional to the supply voltage level. For analog circuits, usually designed to operate at a constant current consumption, the power consumption is directly proportional to the supply voltage level. But the critical performances of analog circuits like power supply rejection (PSR), noise, impedances, gain and bandwidth get critically and adversely affected at lower supply voltages due to lower voltage head rooms. Thus there is an optimal voltage level for consistent analog performance [5]. The digital circuits usually can be supported by a continuous range of supply voltage level for a known speed performance versus power consumption trade-off. The analog performance is also limited by the power supply noise while the digital circuits inject noise into the supply during every switching activity. The above discussion clearly shows that there is a justification in separating and having independent the analog and digital supplies as shown in Figure 1. It also justifies the use of a less noisy power supplies with higher PSR as analog supply and a possible use of relatively more noise supply for majority of digital loads. Usually several critical analog sections have their own dedicated linear regulator as a power supply to avoid any noise coupling issues. Thus a switching regulator can be used as a master system and digital power supplies, while LDOs are used for analog supplies.

A. Test implications of integrated of power supplies

The above discussion shows that the accuracy of the voltage level, PSR, noise, and output impedance are the critical parameters of the on-chip power supplies. In addition the settling time and current consumption at power-up and during switching between various power/performance modes of operation (active and low power modes) and transient load regulation are also critical. PSR, noise, output impedance, settling time and load regulation measurements are often difficult, time consuming and involve costly test and measurement equipments and hence are not part of final production test suite of an SoC. They are only part of device characterization and are performed on limited samples representing the possible extreme corner conditions in a lab bench based setup. The DC PSR, line regulation, DC load regulation, voltage accuracy and current consumption are very critical measurements that are part of production test suite and have impact on the product yield.

B. Pre-silicon verification of PM

Pre-silicon verification of PM is usually done in following steps:

- Requirements are derived and specified at SoC and specific IP levels
- Detailed verification including transistor level (TL) simulations of the analog module
- Simulation and formal verification based connectivity checks are done between the module and rest of SoC
- Simulations at SoC level involving digital RTL/GL and analog BMOD/TL
- Further there can be simulations at SoC level involving digital design, at various abstraction levels like RTL, gate level (GL) with and with out timing annotation, and the analog modules in their various abstraction levels like the BMOD and TL using the digital event driven simulators, SPICE based simulators and the co-simulation platforms

For the SoC level simulations involving RTL and BMOD or TL, the analog and power supply connections have to be comprehended correctly. Conventionally the BMOD are written completely at a digital functional equivalent by just modelling the correct and incorrect behaviour of each terminal or net by an equivalent logic ‘1’ and ‘0’ respectively with out comprehending the analog parameter ranges, settling times and other critical parameters. Such modelling styles provide for bare minimum functional checks and are mainly focusing the analog-digital interface connectivity correctness. But they still lack comprehending currents and current directions, port impedances and any power supply / voltage level sensitivity which cause significant coverage loss at SoC level.

III. REFERENCE TRIM

A. Criticality of trim

On-chip voltage and current reference generation circuits cater to the needs of various analog and mixed signal modules in an SoC. As discussed earlier the performance of on-chip power supplies, analog functional performance and the entire system performance versus power consumption trade-off strongly and critically depend on the accuracy of the reference circuit. Usually a bandgap based reference generator is used to get supply voltage level insensitive references. But still it will have a temperature dependence, usually of the order of few hundred parts per million across the operational temperature range (-40ºC to 125ºC for commercial application or even higher temperatures for military and critical applications, higher temperature also being impacted by the peak power consumption and package types among other things). This temperature sensitivity can be minimized by a process called trimming [6]. The process of trim is required to centre or optimize the reference voltage for process variations and minimize the temperature sensitivity. Further many of the temperature or frequency references built on-chip for various system requirements in term need a constant voltage or current references with minimal temperature sensitivity.

B. Different methods of trim

Usually the absolute accuracy of the bandgap reference voltage for typical embedded applications is required to be in the range of a few milli Volts at room temperature. Due to the range of process variation to be covered, the trim range will be few hundred milli Volts. This results in the trim process to be a long and iterative one costing a significant portion of the overall production test time. There are various trim methods available namely the a) LASER based trimming at production; b) One time programmable trim using anti-fuse or fuse elements; and c) Trimming using the non-volatile memory elements like FLASH, EEPROM and proprietary techniques like E-Fuse [7].

Due to the flexibility and cost impacts majority of the SoCs employ trimming during the wafer testing and programming them in one of non-volatile memory elements discussed above. Further there are necessities for post package trim to overcome the stress [8] or humidity effects caused by the package causing significant variation in the reference value.

C. *Pre-silicon verification of trim procedure*

The reference or any parametric trim process in general is iterative and long. The length of trim process is variable for each every die and cannot be fixed as most parameters under correction are dependent on process corner and other environmental conditions, some of them being ambient/die temperature, supply voltage level, humidity and mechanical stress due to package. Usually the trim process is done under a fixed, nominal, standardised condition (typically at nominal supply voltage, room temperature - 27ºC) and is meant to achieve minimising process and temperature dependence. Verification of the complete trim process pre-silicon is important like all other functional requirements of an SoC. Currently it is done in the several steps listed in section II.B in additional to the following:

- The real test vectors which will be used at the test time for actual trim operation will be generated and verified, their usual format being test description language (TDL).

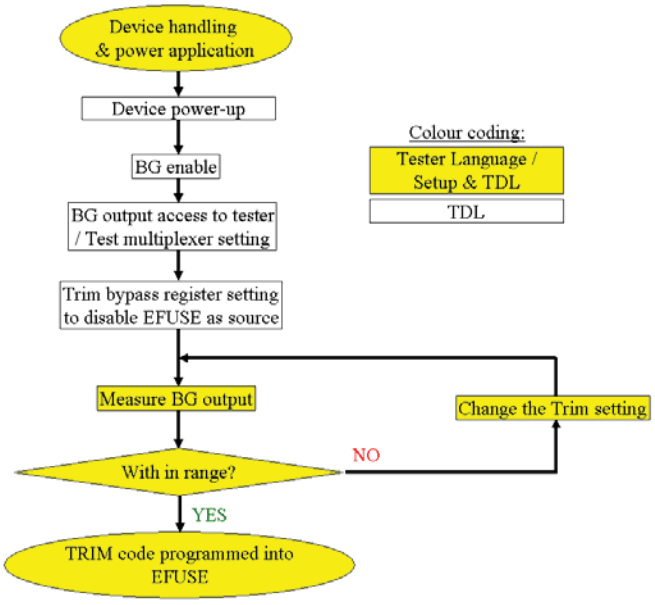


Figure 2 Conventional trim process

TDLs are usually used for test stimuli application for the device and are part of the standard mechanism for most test processes. The actual device testing involves, other than just running the TDL, additional setup for the off-chip test resources which are done manually as well as in a semi-automated fashion using certain tester / test environment specific languages. Especially achieving the right trim code itself is a dynamic process, device dependent and cannot be pre-judged like most other parametric, functional and electrical device characterization whose pass/fail criteria can be judged during design stage. Hence the complete trim procedure cannot be coded during the pre-silicon stages using a static mechanism like a TDL, and hence involves some manual or semi-automated steps involving the tester specific. This procedure is illustrated in Figure 2. Because of this dynamic and manual/semi-automated nature of the trim process, conventionally it is not fully verifiable pre-silicon. Hence only the coarse trim functionality, trim step or resolution, and trim range are verified pre-silicon.

IV. DIGITAL DFT IN A PM INTEGRATED DESIGN

PM integration in a SoC makes the design complex from the digital scan test perspective. Typically the PM module generates the I/O configuration (direction, pull-up, & pull-down) control signals (IO_EN) and the system power on reset (POR) control. To meet various complex system requirements and to save system cost many device pins are shared between different functionality. In such a scenario a set of pins are used for various special system power-up configurations and after power-up are used for other functional requirements. For such pins the state at power-up of the chip before the POR de-assertion has to be latched to use that off-chip configuration information for proper system configuration. The POR signal is used to keep the entire system at a reset condition or an inactive condition until the PM system completely powers up for a reliable state definition. Similarly the IO_EN signal is used to keep all the I/O pins at a pre-determined and safe state per platform requirement until the complete system powers up and takes control of the I/O functionality.

Typical issues encountered in SoCs with integrated PM include scan chain failure in the scan test mode and high I_{DDQ} current [3]. The scan chain failure can happen due to several causes including but limited to

- Certain control signals to the analog IPs are made part of scan chain and cause random activity due to scan shift operation
- Scan flip-flop driving an analog signal either causing contention among them resulting in a scan test failure.

PM module integration also has an effect on I_{DDQ} current. The leakage current due to different PM / analog module can be much higher than digital leakage current. Therefore analog / PM module need to be powered down or to be kept in a low power mode while measuring I_{DDQ} current.

These are issues chiefly caused by incomplete or incorrect isolation mechanism. The isolation requirement is not very straight forward as some signals need to be defined at the same state as power-up reset state while the others need to be defined at a specific state as per the intended system state during the scan test operation. Some signals also require different power-up behaviour depending on the intended system configuration. Current design methods include identifying and defining the various control signals and their state of isolation during the scan test operation. Typically they involve communication between multiple design and development teams in case of complex SoC design.

The scan and I_{DDQ} high current issues caused due to incorrect test mode handling in the presence of PM and analog integration can be overcome by a device debug process involving analysis and identification of problem causing test vectors and avoiding them in subsequent tests or replacing them with newer test patterns, in addition to specific software overrides where possible. In many instances such workarounds are not possible and where possible may result in significant loss of test coverage. In cases where the test coverage loss is unacceptable for cost and quality reasons, silicon iteration is necessary causing further cost escalation and delayed time to market.

V. Unified Top Level Design, Verification & Test Methods Enabling PM Integration

A. BIST methods for testing integrated power supplies

All the production test parameters discussed above require the chip to be powered up by an external power supply and then initiating the power-up sequence before performing the required measurements. Most on-chip power supplies also require significant input and output capacitances, usually ranging from a few hundred pico Farads to a few micro Farads. This in addition to the limited current capacities of the test power supplies and the on-chip power supplies under test and the effects of the package, board and tester probe parasitic elements cause long settling times before reaching the stable state under which the DC measurements can be performed. This results in longer test time and ultimately high test cost impact. Since many of these parameters are usually variable, a significant waiting time margin is added in the tester programmes before the measurements are taken. To over come this on-chip techniques will help as some of these effects are either minimal or don't exist when on-chip measurements are done. There have been BIST techniques presented earlier [4] targeting PM modules which can help in performing simpler and faster measurements causing significant test time and cost saving which are quantified in literature [4]. Transient load regulation effects at the on-chip loads are difficult to measure through off-chip measurements due to additional filtering effects of the parasitic elements discussed earlier. To overcome these usually internal nodes of the power supplies and ground lines are brought through sense lines to a dedicated test port where they are measured. The earlier presented BIST techniques are very useful in measuring on-chip transients. While the current implementations support DC or steady state measurements of such parameters, fast transient measurements require advanced circuit techniques to be implemented to extend the BIST functionality discussed earlier.

Using either the trimmed references available on-chip or with just additional constant, accurate external voltage reference from the tester, faster and accurate pass/fail tests for upper and lower bounds i.e., accuracy of the power supply voltage levels can be performed.

Further optimization can be achieved when multiple on-chip power supplies can be tested for accuracy simultaneously using on-chip BIST implementation.

B. Trim optimisation

Conventional trim process involves a step-by-step approach that involves a) bringing the reference voltage to be trimmed to a tester resource through a dedicated interface, b) measuring it in a nominal condition, c) repeating this measurement by programming the trim control register on-chip through some communication means between the tester and the SoC until the expected reference value is measured. An optimal voltage search algorithm is employed and is implemented either in the tester hardware, or on an on-chip hardware. On-chip hardware based approach requires the measured value by the tester to be communicated to an on-chip controller through digital means. A tester based approach requires the complete trim programmability look-up table to be implemented in the tester hardware and just communicating the next trim code to the SoC. Further optimizations involve employing the on-chip BIST approach, discussed above for an on-chip power supply testing. The BIST approach is resulting in a significant test cost saving [6].

The trim process is further optimised as discussed below. In an SoC with different functional IPs having multiple integrated PM (LDO & BG) modules, not all the IPs and their respective PM modules get enabled by default. To enable them after power-up as and when required will cause significant test time impact as the major component is the power-up and settling times. To avoid this all the IPs and their PM modules are enabled simultaneously (as long as they don't affect each other performance in any way). Usually all effective trim codes at the end of trim process are stored in EFUSE, which is implemented as rows of fixed bit widths. EFUSE writing and reading are performed row wise. The amount of EFUSE bits needed, and resultant test time and power-up time are significantly saved by using complete trim codes (~8 to 10 bits) for one BG only and have a smaller number of bits (~ 2 or 3 bits) to apply necessary correction to the other BGs in the SoC with respect to the BG that is trimmed first. Instead of trimming each and every BG starting from their default setting, subsequent trimming process can be started from the code corresponding to the latest trimmed BG to save test time. All these are effective as on-die variation of BG output levels will not be significant.

C. Analog behaviorual models and verification methodology

For a consistent, fast and early functional simulation of behavioural model (BMOD) of the analog and PM modules are developed. An efficient SoC level simulation now is possible with both functional and test mode scopes and use of the digital in RTL and all the analog modules in BMODs. The BMOD should include all input – output functionalities, both functional and test modes. While all functional behaviour are mandatory to be modelled in BMOD, the performance metrics like PSR, noise, bandwidth, current consumption, impedances are not required. The accuracy and scope of the BMOD can be improved by using VHDL real number or Verilog wreal construct, without resorting to analog macro models or TL simulations. This improves the overall SoC simulation time since both the digital sections in RTL and the analog BMODs are simulated using only digital event driven simulators, and a

need for analog simulator or analog-mixed signal (AMS) co-simulation is obviated.

Further the worst case settling time behaviours, voltage and/or current accuracy are also modelled in a real number based platform. Custom record type real ports in VHDL along with appropriate resolution functions can be used to model power supply nodes and bidirectional analog behaviour. The BMODs have to be validated against the specification at the early design stages and with the TL implementation at late and mature stages of design. Conventionally this BMOD validation involves (BMV), manually comparing the resultant waveforms for each test case of SPICE simulation of TL design and an independent BMOD simulation under same conditions. A proprietary automation script is used for HDL based assertions [9] of TL design. It is further extended to real number assertions using Verilog-A for automated run time checkers for TL simulations and corresponding assertions in the BMOD, to make the respective abstraction levels self-checking. This results in an automated and more efficient BMV with just running the simulations under right test conditions and minimizing or avoiding manual waveform analysis [10][11].

BMODs for bandgap references and on-chip power supplies include their functional behaviour, settling time behaviour, and all test mode behaviours. All the trim and voltage or current level programmability functions are modelled for their nominal behaviour at root temperature. In a VHDL based modelling approach, GENERIC definitions are used for specifying temperature and process information. The process and temperature sensitivity of appropriate parametric behaviours are also modelled. Especially the worst case sensitivity for voltage references and any on-chip temperature sensor outputs are important to be modelled.

Most analog and power management modules in an SoC have a large number of input control signals from the digital section of the SoC, and have a very few digital output signals and analog signals interacting between them. All these digital signals required to be in a known state at power-up to ensure nominal functionality of the SoC. The power-up conditions of the critical control signals can result in inconsistent power-up behaviour and in specific cases cause even catastrophic functional failures. A purely simulation based verification approach will result in a functional failure of the tests under simulation in case of erroneous state of these digital control signals and require a manual debug process to root-cause the failure which can be time consuming and complex. To make this process more efficient the power-up default values of all input control signals, voltage and current reference or bias inputs of analog modules are coded as input condition assertions in respective BMODs. Thus any violation of the valid behaviours in the states of these input signals are reported in run-time revealing the problem scenario as soon as they occur and helping close the debug process in an efficient manner. Such input assertions coded include static power-up default values of the digital control signals, specific order in which a set of control signals are asserted or de-asserted to enable a certain mode of operation of an analog module and critical requirements of specific minimum timing requirements among such signals.

Real number ports are not comprehended for connectivity by most existing, industry standard, commercial EDA tools. Some latest versions have support for connectivity but with sever limitations on the real type ports. They don't support vector or array type of real ports. They neither support multi-field record types which are necessary if one has to model terminal currents and impedances in addition to the basic voltage level functionality. To overcome this difficulty the global signals are used for analog connectivity between IPs at SoC level as shown in Figure 3.

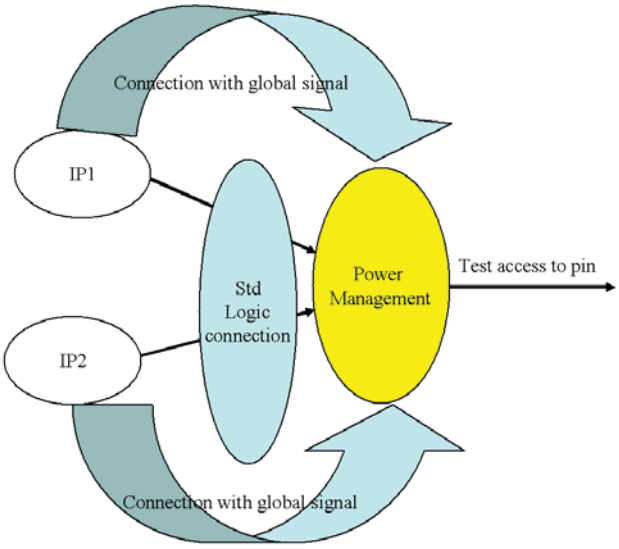


Figure 3 Global signal based analog connectivity for verification

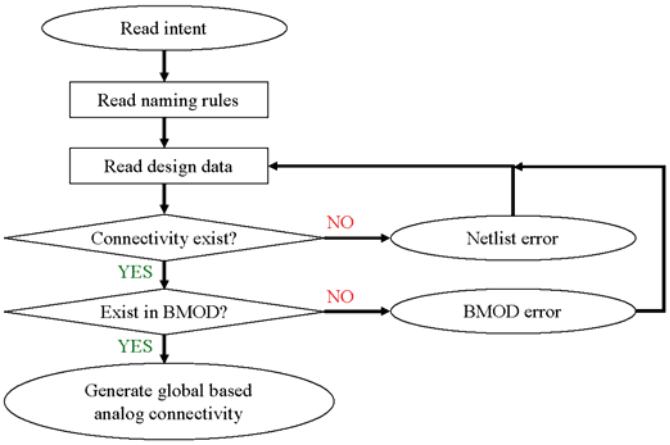


Figure 4 Automated global signal based analog connectivity generation

In general the use of global signals or variables bring with them additional unintended effects. Especially since the global signal based connection between IPs is a superfluous construct and is not governed by the actual physical connectivity there can be false passes and the completeness of all analog connections have to be checked manually or by additional simulations with BMODs with purely logical connections between terminals. To avoid the additional efforts and overcome the error-prone nature of the use of global signals,

the whole process is automated in the following way. The analog connectivity intent is documented in a machine readable format (say the conventional spread sheet format). It is very common to use the spreadsheet based SoC level analog connectivity intent. A custom script based approach, as shown in Figure 4 can be applied to first check for the existence of logical / physical connection for all SoC level analog connectivity and then generate automatically the global signal assignments (the names can be derived based on certain predefined rules which are also followed while developing the BMODs). Such automatically generated connectivity information is used for SoC level simulations.

D. Unified design, verification and test methods for trim

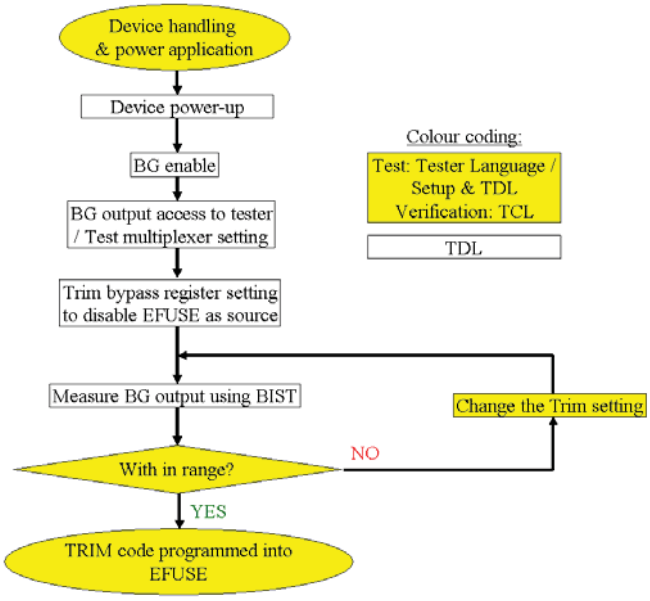


Figure 5 Partial BIST based trim procedure flow diagram

As discussed earlier the conventional trim procedure is dynamic and cannot be verified completely pre-silicon. To overcome this limitation, all the manual steps needed in the tester during the trim process is documented early during the design phase, then an intent document is created (preferably in a machine readable format e.g. spreadsheet). From the intent a verification wrapper is created, say in tool command language (TCL) and the test procedure is created in a compatible tester language. This automation helps having all necessary equivalent verification and testing procedures are generated from one common source and thereby avoiding any manual error. This can further be extended for the BIST based approach as shown in Figure 5. The difference between the conventional and BIST approach being the measurement mechanism.

The verification/test procedures can be automated two different ways. In both methods the complete relation between trim code and the corresponding expected change in output, simply called the trim table, should be comprehended in the verification/test procedure. This can be coded as the look up table (LUT) in the procedures. Based on this the procedure can be coded to change the trim code appropriately for the next search step. This is done in iteration until the trim procedure is completed. For this purpose one single TDL is used to load the trim code into the device with an initial or nominal value. Then the trim code can be changed dynamically by the test / verification procedure as per the decision of the search algorithm. In an alternate approach one can pre-generate number of TDLs for each trim code and choose the TDL to be loaded into the device as per the decision of the search algorithm in every step. Such a unified verification and test method is needed also for a BIST based trim scheme, especially if the search algorithm is implemented in the tester. For a comprehensive and complete BIST scheme on chip the whole procedure becomes much simpler. The tester level iterations are taken care of on-chip and a pass or fail signal or the trim code at the end of the conversion will be transferred back to the tester for writing into the EFUSE later as shown in Figure 6. The full BIST scheme will also have significant test time savings by avoiding the communication between the device and tester at every step of the search algorithm. The various procedures discussed here for the trim scheme result in a tightly couple of verification and test strategy. This will result in a high level of confidence due to completely verified test procedures for silicon test and characterization and minimising iterations in test procedure development post device fabrication.

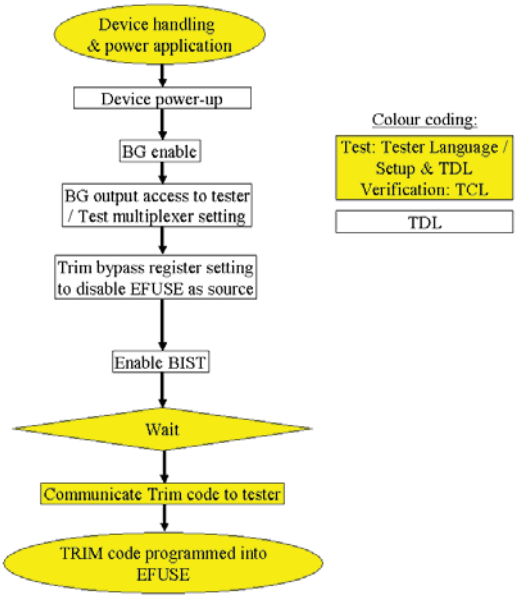


Figure 6 BIST based Trim procedure flow diagram

Most designs require multiple parameters to be trimmed, which are interdependent. For example, a BG voltage reference (VREF) requires temperature co-efficient trim and magnitude accuracy trim. Further a current reference (IREF) derived from the VREF requires a trimmed VREF for its nominal functionality and hence these parameters have to be trimmed in the following order, after taking into account the trim code of previous parameter is effective before the subsequent parameter is trimmed: VREF temperature coefficient (TEMPCO), VREF magnitude accuracy and IREF

trim as illustrated in Figure 7. Further the exact interdependencies have to be monitored in the pre-silicon verification by performing the simulation exactly in the same order and condition stated above.

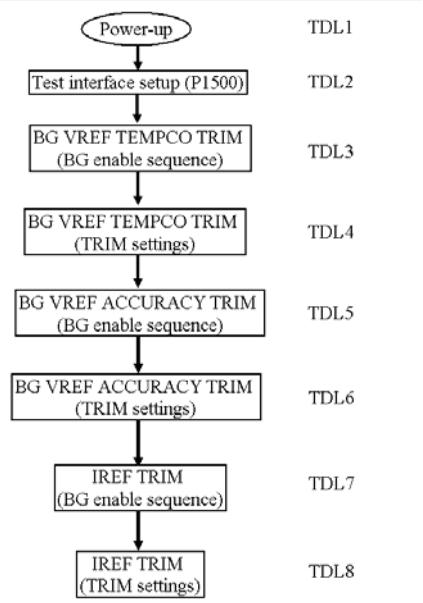


Figure 7 Modular, multi-TDL trim procedure

E. Design methods for scan and I_{DDQ}

The issues caused by incorrect handling of scan, scan based I_{DDQ} tests in case of on-chip analog or PM content are over come by formalizing the process.

1) Gating of critical control signals: SoC level, module level and hybrid approaches

The critical control signals which should be defined at a known state during the scan and I_{DDQ} test modes are identified. These include the module enable signals of critical PM functions like bandgap reference, on-chip supplies etc. whose state should not be disturbed for a robust functionality. Even for those modules which remain disabled or powered-down by virtue of gating the critical signals listed above, there are other signals like the control to the analog test multiplexers connected to such modules can cause unintended effects if allowed to freely assume any state. Such signals or any master controls to such functions, if any, are also to be kept at disable condition through out the scan test operation. Once identified, such signals are documented and their isolation or gating to the defined states can be taken care of at different levels. All such signals from various IPs can be identified and gated at the SoC top level where the mode identification status is available. But this requires a tight and effective communication between various design teams as will be discussed in the next sub-section. The gating can also be implemented at module level by providing the mode identification signal from the SoC test controller to every analog module. Then the analog modules can independently take care of gating all critical signals. Further it makes the module level sign-off much easier and effective.

2) Pre-silicon verification methods: Use of assertions in BMODs & Module level SPICE simulations

The analog modules are to be verified thoroughly in stand alone TL simulations under all system conditions including various functional and test modes. The BMODs need to be modelled thoroughly and comprehensively to comprehend the various test modes, effects of every input control signal on their output functionality. Such BMODs should also be validated against the specification at early design phases and also against the TL implementation at the mature stages of design cycle. While the current consumption is a key parameter that can pin point to any unintended behaviour, it is not always possible to model them easily and hence assertions indicating the current consumption in any given mode are useful to identify exercise of incorrect modes. Further all known illegal conditions including the incorrect isolation gating per given operational mode have to be identified and indicated by assertions. It is not possible to model the incorrect functional behaviour of all illegal conditions which can be observed as a violation at SoC level simulations, and hence having them identified in assertions will be very useful and help quicker debug effort. Verilog-A assertions can be used to have these checkers also functional during TL simulations, while Verilog, VHDL or PSL assertions can be used for BMODs developed compatible with digital event driven simulators. The effectiveness of isolation gating can be thoroughly and easily verified at module level TL simulations, for the module level isolation gating scheme discussed above. It is possible to get the worst case module level current consumption under the I_{DDQ} mode. Again this is relatively easy in case of module level isolation gating scheme. For SoC top level gating scheme, this requires module TL simulations to be re-run with stimuli generated (typically in value change dump, VCD format) from SoC level simulations.

3) SoC level AMS co-simulation & connectivity checks

For the gating implemented at the SoC level, to verify the effectiveness and correctness, SoC level simulations need to be performed under this mode. While a SoC level AMS co-simulation keeping the analog modules in TL can comprehensively verify the implementation, it is not practically possible due to the slow TL simulations and the relatively very long simulation time needed to cover the various scan patterns to be simulated. For the SoC top level gating implementation, this can be effectively achieved by having right BMODs as discussed above and performing SoC level simulations with analog BMODs. For module level gating scheme, the current consumption information from module level TL simulations can be used to easily derive the overall SoC level I_{DDQ} contribution by all analog modules. The digital power estimation can be independently done by conventional power estimation simulation. It is mandated to perform SoC level AMS co-simulation targeting typical, power-up scenarios of all possible modes of operation. Most other system scenarios including the scan test modes are only covered at SoC level simulations using analog BMODs.

In mixed signal design, the SoC level simulations have to be performed for dynamic scan shift operation and I_{DDQ} current measurement test. The dynamic simulation related to scan shift

can be optimized with number of initialization cycles and actual shift mechanism so as to enable faster turnaround time. I/O toggling should be carefully monitored along with internal toggling during scan mode simulations.

VI. RESULTS

The design, verification and test methods discussed in section V can be applied to mixed-signal SoC design with analog and PM integration and their benefits are summarized in TABLE I.

TABLE I SUMMARY OF DESIGN, VERIFICATION & TEST METHODS

#	Stage[1]	Technique	Benefits	Comments
1	D	BIST (20 LDO & 6 BG)	96% test time	2.4 sec conventional test time
2	V	VHDL-RN BMOD	Bugs identified 2 months earlier	Early SoC level bug identification
3	V	Assertions	2 – 4 days / bug	Debug cycle time
4	V	Analog connectivity by global & automation	Critical bugs identified 1 month earlier	Enables quality BMOD & early connectivity bug identification
5	V, T	Trim: Unified verification & test	1 month test development time	Trim procedure matures pre-silicon
6	D, V	Scan & I_{DDQ} gating methodology & verification	2 months test debug time & silicon iteration	To attain correct I_{DDQ} measurement for known good device

[1] Stage: D – Design; V – Verification; T – Test.

The use of BIST scheme for PM characterization and BG trim resulted in about 2.3 seconds (96%) of test time saving per die with 6 on-chip BG and 20 LDOs [4]. The use of VHDL-RN based analog BMOD for PM resulted in better quality of SoC level verification and early identification of integration and specification bugs in the system. This is further enhanced by the extensive assertions. Especially the input condition assertions helped in identifying incorrect integration instantly without spending a long time debugging the erroneous functional behaviour and root cause the source of error. The power-up default conditions, scan and I_{DDQ} mode assertions helped catch SoC level bugs earlier and quicker. Application of global signal based analog connection enabled more accurate and comprehensive real numbered BMOD to be used at SoC level verification for analog modules which is otherwise not possible. Due to the automation of the process, it was achieved with insignificant overhead. The proposed unified verification and test method for trim process will significantly reduce the time it takes to mature the trim procedure after device fabrication. Potentially this scheme will result in a mature and correct by construct trim procedure to be readily available before the device fabrication. This is expected to save at least about a month's time it takes after device fabrication for trim procedure development and closure and hence enabling a faster time to market. The design, verification and test methodology, which is correct by construct, for scan and I_{DDQ} test modes resulted in robust scan and I_{DDQ} test results and saved at least 2 months of effort to debug issues related to I_{DDQ} test and to achieve the correct base-line I_{DDQ} current measurements on a known good device.

VII. CONCLUSIONS

The necessity of integration of power management in a mixed-signal SoC results in several complexities in the SoC design and integration, additional DFT concerns and testing. This paper presents a unified top level methodology for design, verification and test of such SoC with power management integration, specifically targeting the PM test, bandgap trim, and scan based digital structural and I_{DDQ} tests. The methodology includes several aspects on the analog behavioural models, SoC level verification methodology and test procedure improvements.

ACKNOWLEDGMENT

The authors would like to acknowledge the following members of Texas Instruments: Rubin Parekhji for his motivation and support; Keith Kunz, Ranjit Kumar Dash, Harikrishna Parthasarathy, Ravi C Vijayaraghavan, and Sudhir Polarouthu on PM architecture, design and verification support.

REFERENCES

[1] A. Moloney, "Power Supply Management – Principles, Problems, and Parts", Analog Dialogue, Vol. 40, Analog Devices Inc., 2006.

[2] M. Manninger, "Power Management for Portable Devices", European Solid State Circuits Conf., pp. 167-173, 2007.

[3] L. Balasubramanian, P. Sabbarwal, R.K. Mittal, P. Narayanan, R.K.Dash, A.D. Kudari, S. Manian, S. Polarouthu, H. Parthasarathy, R.C. Vijayaraghavan, S. Turkewadikar, "Circuit and DFT Techniques for Robust and Low Cost Qualification of a Mixed-signal SoC with Integrated Power Management System", Design, Automation and Test in Europe (DATE), 2011.

[4] R. Mittal, L. Balasubramanian, A. Sontakke, H. Parthasarthy, P. Narayanan, P. Sabbarwal and R. A. Parekhji, "DFT for extremely low cost test of mixed signal SOCs with integrated RF and power management," IEEE Intl. Test Conference (ITC), 2011, in press.

[5] B. Razavi, Design of Analog CMOS Integrated Circuits, McGraw-Hill, 2001.

[6] D. Spady, V. Ivanov, "A CMOS bandgap voltage reference with absolute value and temperature drift trims", Intl. Symp. on Circuits and Systems, pp. 3853-3856, 2005.

[7] W.R. Tonti, "eF The methodouse Design and Reliability," Intl, Integrated Reliability Workshop, pp. 1 – 32, 2008.

[8] Buddhika Abesingha, Gabriel A. Rincón-Mora, and David Briggs, "Voltage shift in Plastic-Packaged Bandgap References", IEEE T. CAS-II, Vol. 49, No. 10, pp. 681-685, Oct. 2002.

[9] V. Sharma, G. Lakshmanan, S. Tare, S. Dhamankar, "Predicting the Correlation between Analog Behavioural Models and SPICE Circuits for robust SoC Verification," IEEE BMAS Work-shop, pp. 130-135, Sep 2008.

[10] L. Balasubramanian, P. Sundar, T. W. Fischer, "Assertion Based Self-checking of Analog Circuits for Circuit Verification and Model Validation in SPICE and Co-simulation Environments," Accelera Design Verification Conference (DV Con), Mar. 2011.

[11] L. Balasubramanian, Mithun Kumar G, Chandrashekar BG., " Assertion based efficient analog behavioural model validation for early functional simulation of mixed signal SoC," Cadence CDN Live India, Oct. 2011.

Additional Path Delay Fault Detection with Adaptive Test Data

K. Miyase [1], H. Tanaka [1], K. Enokimoto [1], X. Wen [1], S. Kajihara [1]

[1] Kyushu Institute of Technology, Iizuka, Japan

Abstract– Adaptive tests utilize data collected after the testing of manufactured LSIs in order to make LSI testing efficient and to ensure a high yield. In this work, we improve the ability of transition delay tests to detect path delay faults by using adaptive test data. By improving path delay fault detectability, timing related defects can be found early on, resulting in increased test application time. The proposed method detects the most path delay faults possible via transition delay tests without test data inflation. Experimental results for large ITC'99 benchmark circuits demonstrate that the proposed method can detect many path delay faults in addition to transition delay faults in transition delay tests.

Keywords *– test generation, path delay fault, transition delay fault, adaptive test, X-bit.*

I. Introduction

Process-variation must be considered in logic synthesis, placement, and routing in order to ensure a given performance for LSI. Static timing analysis (STA) [1] extracts a critical path, which has the worst slack under worst case conditions. Usually, such critical paths are too pessimistic; therefore, statistical static timing analysis (SSTA) [2-4] is used to avoid this pessimism.

Using the aforementioned critical path in path delay testing is less effective, because the paths are not always sensitized in functional operation. Furthermore, process-variation has to be considered in order to avoid overkill, since such overkill is directly related to yield loss. If non-sensitized paths or paths causing overkill are used, testing for the path becomes meaningless.

It is important for candidate paths in path delay testing to be carefully selected, because the total number of paths in a logic circuit is extremely large. Some adaptive test techniques [5-8] utilize statistical data to select candidate paths, which are sensitized in functional operation and do not cause yield loss.

An example flow of an adaptive test is shown in Figure 1. In the figure, the data used for the adaptive test is obtained through failure analysis. Note that the data for the adaptive test is also obtained by sensor circuits [9, 10] embedded on a chip or a wafer. When a decent amount of adaptive data exists, paths subject to path delay testing become more effective.

Adaptive test techniques contribute to the avoidance of pessimistic testing as well as the identification of defective chips as early as possible. Once a chip has been identified as defective, the remaining test vectors do not need to be applied to the chip. Consequently, the earlier a chip is identified as defective, the more test application time it can save.

In general, transition delay tests are applied earlier than path delay tests, since the majority of timing related defects can be found by transition delay tests. Therefore, if transition delay tests can find timing related defects also detectable by path delay tests, the path delay tests do not need to be applied, resulting in increased test application time for path delay tests.

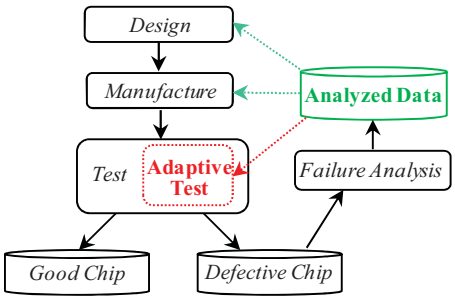


Figure 1. Conventional LSI Test and Adaptive Test Flows.

Obviously, not all path delay faults can be detected by transition delay tests, since the total number of paths is extremely large. However, realistic and effective paths for path delay testing can be extracted by utilizing adaptive test data. Paths in the list may be collected by failure analysis or an embedded sensor used in adaptive testing.

In this paper, we propose a method for improving the ability of transition delay tests to detect path delay faults using conventional transition and path delay ATPG. Given a path list, the proposed method assigns priorities to each

path according to its slack value in order to preferentially detect the path delay faults with the worst slack value. Experimental results for large ITC'99 benchmark circuits show the proposed method can increase path fault detectability that will help to save overall test application time for manufactured LSI chips. Note that path selection is not an objective of this work.

The rest of the paper is organized as follows: Section II describes the background; the proposed method is described in Sections III; the experimental results are shown in Section IV; and our conclusions are summarized in Section V.

II. Background

Previously, we proposed a method for detecting bridging faults with stuck-at tests [11]. The method identifies *X*-bits in generated stuck-at tests to obtain test cubes [12]. After that, it utilizes the dynamic compaction technique [13] to further detect bridging faults by applying stuck-at tests. Fig. 2 shows method [11] and the method proposed in this paper.

Both methods utilize the analyzed data obtained through testing of manufactured LSIs. Like the number of path delay faults, the number of bridging faults is extremely large; method [11] therefore narrows down bridging faults that are likely occur by utilizing analyzed data (adaptive test data).

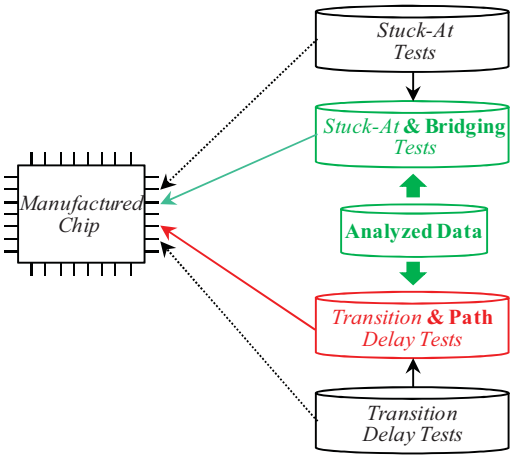


Fig. 2. Previous Method and Proposed Method.

The benefit of the previous method is that the size of final tests that detect both stuck-at faults and bridging faults is relatively small, since *X*-bit identification [12] does not cause test data inflation without stuck-at fault coverage loss. The drawback of method [11] is that it has to incorporate dynamic compaction, which often causes aborts in test generation or causes a longer computation time.

In this paper, we use a simple merging technique that merges compatible test cubes for transition faults and path delay faults, allowing us to achieve a relatively short computing time in which to obtain transition delay tests that detect path delay faults.

III. Improving Selected Path Detectability

In order to save overall test application time for manufactured LSIs, the proposed method detects the greatest number of path delay faults possible in transition delay tests. We first describe how to detect path delay faults with transition delay tests, and then we present a technique that prioritizes path delay tests in order to achieve greater test application time saving.

A. Merging Test Cubes for Detecting Path Delay Faults with Transition Delay Tests

In order to detect additional path delay faults with transition delay tests, the proposed method merges test cubes for transition delay faults and path delay faults. The proposed method prepares test cubes for both transition delay faults and path delay faults using ATPG without random-fill. This technique can easily be implemented with a general commercial tool.

Fig. 3 shows test cube *ct* for transition delay faults $tf_1$and tf_2, and test cube *cp* for path delay fault *pf*. Dynamic compaction may be able to detect *pf* by *ct*, but it may also result in a longer computation time. The proposed method prepares *pf* and merges *ct* with *cp*, resulting in a shorter computation time.

However, even though we used adaptive test data in order to narrow down the number of path delay faults, not every path delay fault can be detected. Therefore, we prioritize each path so that path delay faults that likely occur can be preferentially detected.

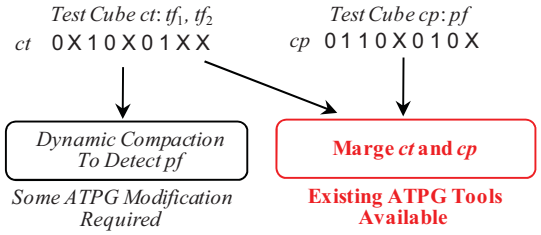


Fig. 3. Previous Method and Proposed Method.

B. Ordering Path Delay Tests for Early Detection

Utilizing adaptive test data, we can narrow down the number of path delay faults that likely occur in manufactured LSI chips. However, even when detected by transition delay tests, the number of path delay faults is still

large. Therefore, we prioritize each path according to its slack value.

If a path has only a small amount of slack, it causes timing failure with a slight increase in delay. On the other hand, if a path has a large amount of slack, its path delay faults can be detected only if there is a significant increase in delay. Such path delay faults can be covered (detected) with transition delay tests. Therefore, path delay faults with a short amount of slack should be detected prior to path delay faults with a large amount of slack.

Fig. 4 summarizes the concept of the proposed method. We first sorted the transition delay test cubes in descending order of *X*-bits. A test cube with a large number of *X*-bits can be easily merged with another test cube. Therefore, the probability of merging a transition delay test cube with a path delay test cube increases significantly. The sorted test cubes for transition delay faults is shown to the left in Fig. 4.

Next, we generated test cubes for given paths and prioritized each path in the list based on its slack value. Although prioritization can be defined by users, we sorted the paths in increasing order of path slack value. Clearly, since not every path delay fault can be detected, we tried to detect path delay faults with higher priority. In Fig. 4, the sorted test cubes for path delay faults are shown to the right.

Finally, we merged test cubes for path delay fault with test cubes for transition delay fault. The merging operation is performed based on the order of path slack value. With the example shown in Fig. 4, we tried merging cp_1 with ct_1. If cp_1 and ct_1 were successfully merged, we proceeded to cp_2. If cp_2 could not be merged with ct_2, we tried merging cp_2 with ct_3, and so on. Although this is not an optimum algorithm, many test cubes can be merged, as demonstrated by the experimental results.

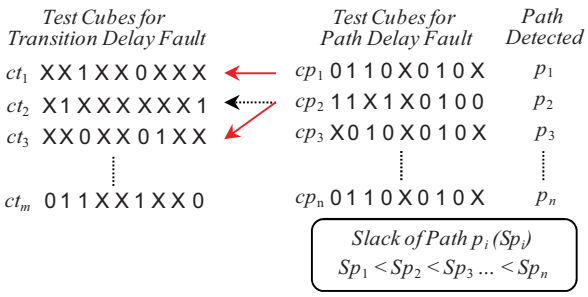


Fig. 4. Priority Detection of Short Slack Paths.

C. Procedure of the Proposed Method

Fig. 5 shows the procedure for the proposed method. The inputs are test cube set *CT* for transition delay faults and a path set, *P*, extracted through the use of adaptive test data. The output is a transition delay test set *V* that can detect the greatest number of high priority path delay faults possible.

In the proposed method, given a test cube set *CT* and path set *P*, we: (1) sort transition delay test cubes in descending order of *X*-bits; (2) generate test cubes for given paths; (3) sort the path delay test cubes based on the slack value of each path; and (4) merge as many path delay test cubes as possible with the transition delay test cubes. Finally, test vector set *V*, which can detect transition delay faults and path delay faults, is obtained.

```
Improving Selected Path Detactability Procedure( )
    Input: Test cube set CT and Path set P;
    Output: Test vector set V;
{
   /* ordering ct in CT in decreasing order of X-bits */
   ordering_ct(CT);

   /* generate test cube set CP for P */
   CP = path_atpg(P);

   /* ordering cp in CP in increasing order of slack Sp */
   ordering_cp(CP);

   /* Merging cp with ct */
   for each test cube cp_i in CP (i=1 to N_cp){
      for each test cube ct_j in CT (j=1 to N_ct){
         if ( cp_i is compatible with ct_j ){
            merge cp_i with ct_j ;
            break;
         }
         else if ( cp_i is not compatible with ct_j ){
            continue;
         }
      }
   }
   /* filling random values to X-bits remained */
   random_fill(CT);

   return V;
}
```

Fig. 5. Procedure of Proposed Method.

IV. Experimental Results

We implemented the proposed method using a programming language and EDA tools, and then applied it to larger four ITC'99 benchmark circuits. We generated test cubes for transition delay fault and path delay fault using TetraMAX®. To prepare paths subject to path delay ATPG as a given path set *P* presented in Fig. 5, we used a STA tool (PrimeTime®). We implemented the proposed merging procedure shown in Fig. 5 using C programming language. The tools and C program were performed on workstations Dual-Core AMD OpteronTM: 2.8GHz / Memory: 16GB and Intel Xeon X5680: 3.33GHz / Memory 64 GB, respectively.

Table I shows the circuit size, number of gates, and number of flip-flops in the first three columns. "# STA paths" is the number of paths obtained by STA, while "# Det. Path" is

the number of detectable paths obtained by the path delay fault ATPG. We used these detectable paths as the given paths, which are represented as the analyzed data shown in Fig. 1. In practice, paths are collected from adaptive test data by users with appropriate criteria.

TABLE I. RESULTS OF PATH ATPG

Circuits	*# Gates*	*# FFs*	*# STA Paths*	*# Det. Path*
b22	38675	645	642	100
b17	26559	1317	511	189
b18	86754	3064	2452	508
b19	172265	6130	4865	874

Table II shows the results of the proposed method. The second column shows the number of test cubes for transition delay fault. The columns "before" and "after" indicate the results before or after applying the proposed method under the columns "# Det. Paths" and "Trans. F. Cov.", respectively. "# Det. Paths" is the number of path delay faults detected by transition delay tests, and "Trans. F. Cov." is the fault coverage of the transition delay tests. The last two columns list CPU times for transition delay fault ATPG and the merging operation shown in Fig. 4.

TABLE II. RESULTS OF PROPOSED METHOD

Cir.	*# Trans. Vec.*	*# Det. Paths*		*Trans. F. Cov.*		*CPU time (sec)*	
		before	*after*	*before*	*after*	*ATPG*	*Marge*
b22	2135	**3**	**6**	94.0%	94.0%	321.5	0.2
b17	1235	**14**	**155**	72.5%	72.6%	598.1	0.1
b18	2598	**5**	**502**	64.8%	64.9%	2194.2	1.0
b19	4564	**19**	**862**	66.0%	66.0%	5191.9	2.7

From Table II, it is clear that the proposed method is capable of improving the ability of transition delay tests to detect path delay faults. While only 3 additional path delay faults could be detected for b22, 843 additional path delay faults could be detected for b19. Furthermore, there is no transition delay fault coverage loss even after applying the proposed method. CPU time for merging operation is quite short compared for transition test cube test generation.

V. CONCLUSIONS

In this work, we improved the ability of transition delay tests to detect path delay faults through the use of adaptive test data. We assumed a path list collected by STA and ATPG tools as the adaptive test data and proposed an algorithm that merges path delay test cubes with transition delay test cubes.

Experimental results for large ITC'99 benchmark circuits demonstrated that the proposed method could detect many path delay faults in addition to transition delay faults in transition delay tests. Future work includes analysis of the proposed method's time-saving impact on test application and experiments with larger industrial circuits.

ACKNOWLEDGMENT

This work was partly supported by JSPS KAKENHI Grant-in-Aid for Young Scientists (B) 23700061. This work was also partly supported by JSPS KAKENHI Grant-in-Aid for Scientific Research (B) 22300017.

REFERENCES

[1] M. Orshansky, et al., "A General Probabilistic Frame-Work for Worst Case Timing Analysis," *Proc. Design Automation Conf.*, pp. 556-561, 2002.

[2] M. Orshansly, et al., *Design for Manufacturability and Statistical Design*, Springer, 2007.

[3] C. Visweswariah, et al., "First-order Incremental Block-based Statistical Timing Analysis," *Proc. Design Automation Conf.*, pp. 331-336, 2004.

[4] H. Chang, et al., "Statistical Timing Analysis under Spatial Correlations," *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 24, No. 9, pp. 1467-1482, 2005.

[5] S. Benner, et al., "Optimal Production Test Times Through Adaptive Test Programming," *Proc. Int'l Test Conf.*, pp. 908-915, 2001.

[6] R. Madge, et al., "The Value of Statistical Testing for Quality, Yield and Test Cost Improvement," *Proc. Int'l Test Conf.*, pp. 322-332, 2005.

[7] M. Shintani, et al., "An Adaptive Test for Parametric Faults Based on Statistical Timing Information," *Proc. Asian Test Symp.*, pp. 151-156, 2009.

[8] T. Uezono, et al., "Path Clustering for Adaptive Test," *Proc. VLSI Test Symp*., pp. 15-20, 2010.

[9] T. Takahashi, et al., "On-Die Parameter Extraction from Path-Delay Measurements," *Proc. Asian Solid-State Circuits Conf.*, pp. 101-104, 2009.

[10] X. Wang, et al., "A Novel Architecture for On-Chip Path Delay Measurement," *Proc. Int'l Test Conf.*, Paper 12.1, 2009.

[11] K. Miyase, et al., "On Improving Defect Coverage of Stuck-at Fault Tests," *Proc. Asian Test Symp*., pp. 216-221, 2005.

[12] K. Miyase, et al., "XID: Don't Care Identification of Test Patterns for Combinational Circuits," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol. 23, No. 2, pp. 321-326, 2004.

[13] P. Goel, et al., "Test Generation and Dynamic Compaction of Tests," *Digest of Papers 1979 Test Conf.*, pp. 189-192, Oct. 1979.

On the Optimality of *K* Longest Path Generation

Jie Jiang[1], Matthias Sauer[2], Alexander Czutro[2], Bernd Becker[2], Ilia Polian[1]
[1]Department of Informatics and Mathematics, University of Passau, Innstr. 43, D-94032 Passau, Germany
[2]Department of Computer Science, Albert-Ludwigs-University, Georges-Koehler-Allee 51, D-79110 Freiburg, Germany
{jie.jiang|ilia.polian}@uni-passau.de {sauerm|aczutro|becker}@informatik.uni-freiburg.de

Abstract—Adequate coverage of small-delay defects in circuits affected by statistical process variations requires identification and sensitization of multiple paths through potential defect sites. Existing *K* longest path generation (KLPG) algorithms use a data structure called *path store* to prune the search space by restricting the number of sub-paths considered at the same time. While this restriction speeds up the KLPG process, the algorithms lose their optimality and do not guarantee that the *K* longest sensitizable paths are indeed found. We investigate, for the first time, the effects of missing some of the longest paths on the defect coverage. We systematically quantify how setting different limits on the path-store size affects the numbers and relative lengths of identified paths, as well as the run-times of the algorithm. We also introduce a new optimal KLPG algorithm that works iteratively and pinpointedly addresses defect locations for which the path-store size limit has been exceeded in previous iterations. We compare this algorithm with a naïve KLPG approach that achieves optimality by setting the path-store size limit to a very large value. Extensive experiments are reported for 45nm-technology data.

***Index Terms*—Parameter variations, small-delay testing, *K* longest path generation**

I. INTRODUCTION

Delays in nanoscale circuits are heavily affected by statistical process variations [1][2][8][13][19][20][21]. This has a fundamental impact on the testability of small-delay defects (SDDs). A given SDD may lead to violating timing conditions (and therefore be a critical defect) in one manufactured instance of a circuit, but may not impact the functionality of a different instance of the same circuit. To reliably detect a critical SDD, different excitation and propagation conditions must be fulfilled for each circuit instance. Hence, standard SDD test methods [14][16] do not always provide a good defect coverage under process variations.

The following two basic classes of approaches have been suggested to deal with this problem. Approaches from the first class are based on constructing a model of process variations and integrating it into the test generation framework [10][11][12][22]. For example, a space of process parameters is defined in [10]; for a given SDD and a test pattern, the process-parameter combinations for which the SDD is detected are identified. Test generation is continued until a sufficient portion of the parameter space has been covered. The drawback of such approaches is the difficulty to obtain a technology-aware model of process variations that is accurate as well as sufficiently compact for integration into a test-generation flow. Specific challenges are the unclear validity of assumptions on process variations in a given technology; the high effort to perform a statistically significant number of either electrical simulations or actual measurements; the large number of process parameters which affect many locations of the circuit; and the complex interaction of process variations and actual defect mechanisms (such as resistive opens [6]).

The second class of approaches covers an SDD by propagating the defect through a number of different paths [15][17][18]. For each SDD-location from a given set (for instance, outputs of all the logic gates in a circuit), a number *K* of longest sensitizable paths are identified (where the length of the path is determined using nominal gate delays). For each found path, a test pair detecting the defect by sensitizing this path is generated. This *K* Longest Path Generation (KLPG) is done in order to account for the fact that the lengths of the paths vary in different manufactured instances of a circuit. Detection of a critical SDD in a manufactured circuit instance is guaranteed as long as one of the *K* paths through its location selected by KLPG is indeed the longest sensitizable path in the regarded circuit instance. KLPG does not require explicit modeling of process variations and can be assumed to be more computationally feasible than the variation-aware test generation mentioned before.[1] A related problem is the search for *K* globally longest sensitizable paths in a circuit [5][9][13].

KLPG algorithms maintain a number of partial paths which are possible sub-paths of one of the *K* longest sensitizable paths. These sub-paths are continuously extended until they either become a complete sensitized input-to-output path, or until they are proven to be unsensitizable. A number of advanced speed-up techniques for KLPG have been shown to scale this basic algorithm to large circuits [5][9][15]. Most of these techniques have no impact on the *optimality* of the algorithm, i.e., the algorithm yields *K* longest sensitizable paths if that many different paths exist. A KLPG algorithm is not optimal if it may miss one or several of the *K* longest sensitizable paths and either finds a shorter path or no path at all. The optimality of KLPG implementations in [5][15] is affected by imposing an upper limit on the number of partial paths maintained at any point of time. The corresponding data structure is called

[S] This work has been supported by the German National Science Foundation (DFG) under grant numbers BE 1176-15/2, PO 1220-2/2 and GRK 1103.

[1] This is similar to n-detection, i.e., testing the same stuck-at fault by multiple different patterns, in order to cover not modeled defects such as shorts or opens [6]

a *path store* in [15] (and corresponds to Ψ in [5]). It contains partial paths for which it is not yet definitely known whether they can be extended to one of the *K* longest sensitizable paths. Imposing a limit on the path store size leads to a *path-store overflow*, which implies that some of the partial paths cannot be maintained. If some of the actual *K* longest paths are extensions of such an excluded partial path, they will not be found, thus leading to a sub-optimal solution.

In [15], a path-store size limit of 3,000 is used for the experiments. Different limits varying between 10 and 100 are used in [5] (these low values might be an explanation for the very low run-times reported in that paper). This prompts a question whether the impact of path-store overflows on the quality of the obtained solution is significant. This is the subject of investigation in this paper.

Using an implementation of a KLPG algorithm along the lines of [15], we systematically evaluate the impact of path-store overflows on the quality of the found paths. We identify SDDs for which not all partial paths fitted into the path store and compare the paths determined by the KLPG algorithm with the optimal solution, i.e., with actual *K* longest sensitizable paths. For different values of the path-store size limit π_{max}, we report the number of SDDs for which less sensitizable paths were found due to the overflows. For SDDs with an equal number of found paths, we compare the lengths of the paths. Using a path of a lower length for testing implies a sub-optimal coverage of certain defect sizes. In particular, we calculate the number of SDDs for which the overflows lead to substantially shorter paths.

Since the aim of imposing a path-store size limit is run-time efficiency, we also study the scalability of the algorithm as a function of π_{max}. Moreover, we investigate the cost of generating a provably optimal solution. One trivial way to obtain such a solution is to use a very high value of π_{max}, or a dynamic data structure with no imposed limit. We propose an alternative iterative algorithm *Opt-KLPG*. We investigate the run-time behavior of *Opt-KLPG* and compare it to the original algorithm. We are not aware of a previous study that systematically investigates the optimality of KLPG.

II. *K* LONGEST PATH GENERATION ALGORITHM

A. Overview of the method

The KLPG algorithm was implemented along the lines of [15] (a flowchart is found in Figure 1). As in [15], partial paths are grown from the circuit's inputs to the outputs. They are required to pass through the SDD location. To grow a path, the side-inputs of the corresponding gates are assigned appropriate logic values and their implications are calculated. Whenever an assignment results in a local conflict, the partial path is dropped. For each partial path under consideration, a value called *max esperance*, corresponding to the maximal length of an extension of the partial path, is calculated and continuously updated. When *K* complete sensitizable paths are available, all partial paths with a max esperance less than the minimal length of these *K* paths are dropped. This is done because these paths will definitely be shorter than the *K* already known paths when fully extended.

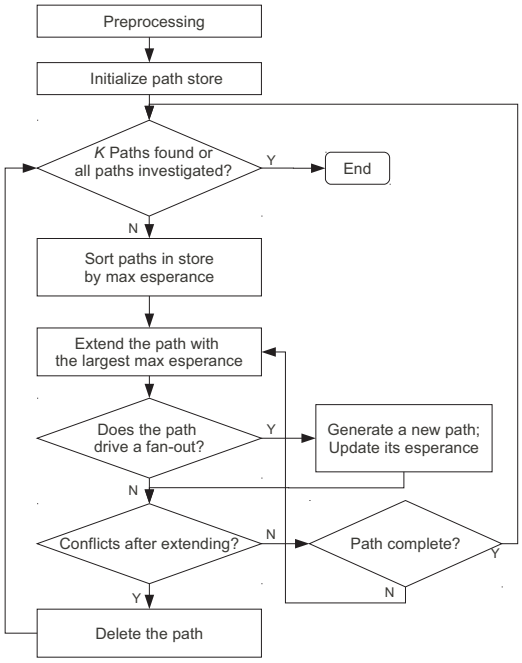


Figure 1. Flowchart of the KLPG algorithm from Section II.

As soon as a complete path of sufficient length and without local conflicts has been calculated, it is sensitized. In [15], the FAN algorithm is used for this purpose. In our implementation, a Boolean-Satisfiability (SAT) instance is constructed and the sensitization is performed using the TIGUAN engine [7]. Unlike [5], no incremental SAT solving is used.

B. Pre-processing

Before path generation starts, all gates in the circuit are topologically sorted in decreasing order of their levels. The max esperance is calculated for all gates in the sorted order (where max esperance of a gate is an upper bound of maximal length of a partial path from the gate to an output). The recursive max esperance calculation works as follows: if a gate drives the primary output and no other gates, the max esperance of the gate equals its maximal (port-to-port) gate delay. Otherwise the gate's max esperance is the sum of its maximal gate delay and the maximum of max esperances of gates driven by its output (which have been calculated already).

C. Path generation

For each SDD in the fault list, the path store is progressively filled with the partial paths that start at a primary input in the SDD's input cone, since only these paths can be potentially extended to the SDD location. The max esperance of a partial path is calculated for each considered path, by adding the actual partial-path length and the max esperance of the last gate on the partial path. The partial path with the largest max esperance is selected and extended by one gate. In case of a fanout to multiple gates, a copy of the partial path is stored in the path store unless its size limit π_{max} is exceeded. Then, the partial path is extended by the gate with the largest max esperance. For example, if partial path *a–b–c* is considered and *c* has three successors *d*, *e* and *f*, a copy of the partial path *a–b–c* is stored

in the path store and the partial path is extended such as to have the highest max esperance, e.g., *a–b–c–d*. The copy of partial path *a–b–c* kept in the path store contains only two successors (*e* and *f*), as successor *d* is being evaluated already. Moreover, the max esperance of the copy of *a–b–c* in the path store is computed taking into consideration successors *e* and *f* only.

After path extension, the *sensitizability* of the extended partial path, i.e., its ability to propagate a signal transition from the primary input to its current last gate, is checked. We implemented sensitizability conditions for robust, non-robust and functional sensitization [4]. To guarantee the sensitization of the path, the side inputs of the newly added gate are assigned logic values according to the selected sensitizability conditions. The value assignments can imply values on other signals in the circuit; if these signals are already set to a different value, a *conflict* is detected and the extending fails. We apply the "direct implication" technique [15] to identify conflicts. Not all possible conflicts are guaranteed to be detected by this technique.

If the extension of a partial path fails, that partial path is dropped as it cannot be extended to a sensitizable path. Then, the partial path with the largest max esperance is taken out of the path store. The algorithm attempts to extend this path guided by the max esperance of the successor gates. Partial paths that have been considered previously are excluded from consideration. For example, if *a–b–c–d* could not be sensitized, partial path *a–b–c* may be selected and extended by the remaining successor (e.g., *e*) with the highest max esperance.

D. Path sensitization

A completely extended path (going from a primary input through the SDD location to a primary output) has no conflicts identified by the direct implication technique, but such a path is still not guaranteed to be sensitizable. In order to check the sensitizability of a complete path, we derive a *conditional multiple stuck-at* (CMS@) fault [7] from the generated path. A CMS@ fault consists of a list of fault activation conditions and a list of stuck-at-like fault effects. In our case, fault activation conditions are given by the values assigned to the side inputs of the gates on the path, and the only fault effect is the erroneous value on the SDD location. We use the SAT-based ATPG tool TIGUAN [7] to find a test that detects the constructed CMS@ fault. If a test pattern is found, it sensitizes the path. If TIGUAN reports that the CMS@ fault is untestable, the path is not sensitizable.

The KLPG algorithm terminates when either *K* sensitizable paths have been found or when the path store becomes empty.

III. The Optimal KLPG Algorithm

The algorithm *Opt-KLPG* is outlined in Figure 2. It starts by running KLPG with a user-defined path-store size limit π_{max}. The path store is empty at the beginning of the first iteration of *Opt-KLPG*. In contrast to the algorithm in the previous section, partial paths which are not included in the path store due to overflows (*overflow paths*) are kept for subsequent iterations. Only the paths in the path store are considered by the KLPG algorithm. The first iteration of *Opt-KLPG* yields a (potentially sub-optimal) set of sensitizable paths for each SDD.

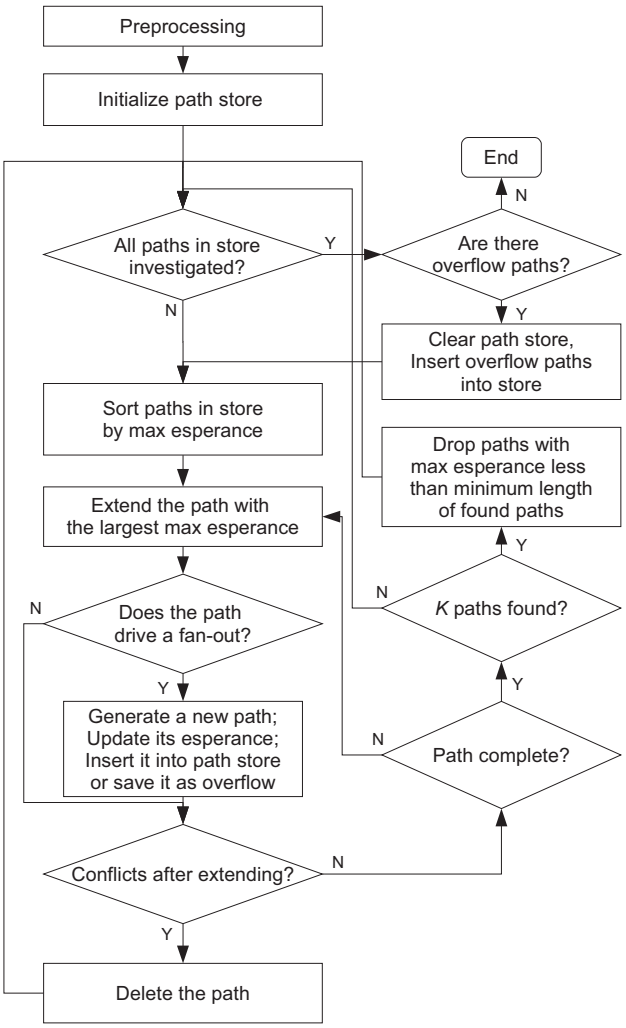


Figure 2. Flowchart of the *Opt-KLPG* algorithm from Section III.

If an SDD is affected by an overflow, subsequent iterations of *Opt-KLPG* invoke further KLPG runs (using the same π_{max}-value). In these iterations, the path store is initialized with overflow paths generated in previous iterations. If the number of such partial paths exceeds π_{max}, the ones with the largest max esperance are selected. This means that some of the paths that have not been selected may be re-generated during KLPG.

As soon as *K* sensitizable paths have been found, all partial paths with max esperance below the length of the shortest of the *K* paths are dropped immediately. This procedure is iterated until all overflow partial paths have been inserted into the path store and investigated. Note that the solution quality yielded by *Opt-KLPG* is not affected by the choice of π_{max} as the algorithm is guaranteed to always find the optimal solution.

IV. Experimental Results

We applied our implementation of KLPG and *Opt-KLPG* to ISCAS-85 circuits (which are small but are known to have a large number of reconvergencies and are therefore challenging for path-selection algorithms), combinational cores of ITC-99 circuits, and industrial circuits provided by NXP. We synthe-

TABLE I. EXPERIMENTAL RESULTS FOR FUNCTIONAL SENSITIZATION

Circuit (gate cnt)	π_{max}	**KLPG**				***Opt-KLPG***					**Comparison**		
		#Over-flows	Path length	CPU time [s] Total	CPU time [s] Excl SAT	#Over-flows	Average # re-iterations	Path length	CPU time [s] Total	CPU time [s] Excl SAT	Δ path length [%]	Gates with less paths	Gates with short paths
c1908	10	143619	75569	36	4	332249	42.25	86185	346	26	14.05	15	671
(795)	50	215713	82819	76	8	280743	7.56	86185	337	22	4.06	1	352
	100	125803	85717	68	8	163067	2.46	86185	309	21	0.55	0	118
	500	13356	86155	82	13	29950	0.08	86185	306	42	0.03	0	4
	1000	14932	86181	122	30	23696	0.03	86185	343	77	0.00	0	2
	1500	16278	86181	177	56	20164	0.02	86185	386	122	0.00	0	2
	3000	12902	86181	375	186	12902	0.01	86185	526	252	0.00	0	2
	30000	0	86185	815	510	0	0.00	86185	774	490	0.00	0	0
c5315	10	117690	201540	60	7	170043	8.02	215813	147	9	7.08	8	1306
(2228)	50	68354	215615	94	8	74667	0.97	215813	126	8	0.09	0	96
	100	33548	215770	111	9	35998	0.26	215813	131	8	0.02	0	21
	500	1402	215812	132	11	1402	0.01	215813	134	11	0.00	0	1
	1000	0	215813	134	11	0	0.00	215813	134	11	0.00	0	0
	1500	0	215813	134	12	0	0.00	215813	134	11	0.00	0	0
	3000	0	215813	134	11	0	0.00	215813	133	11	0.00	0	0
	30000	0	215813	134	12	0	0.00	215813	135	12	0.00	0	0
c7552	10	210254	249957	141	17	258557	9.19	260443	324	22	4.20	25	1858
(2952)	50	102766	259465	211	17	110967	1.06	260443	277	20	0.38	0	382
	100	49843	260108	237	20	52150	0.30	260443	268	20	0.13	0	125
	500	509	260443	265	23	509	0.00	260443	264	22	0.00	0	0
	1000	0	260443	264	22	0	0.00	260443	265	22	0.00	0	0
	1500	0	260443	264	23	0	0.00	260443	265	22	0.00	0	0
	3000	0	260443	269	23	0	0.00	260443	264	23	0.00	0	0
	30000	0	260443	264	23	0	0.00	260443	270	23	0.00	0	0
b14	10	214621	97653	127	15	1417923	21.03	126305	5688	336	29.34	19	931
(6763)	50	475719	113633	367	19	1360685	4.09	126305	5792	147	11.15	2	662
	100	573137	121862	490	23	1201388	1.84	126305	5255	114	3.65	0	550
	500	485778	125788	1745	48	761236	0.26	126305	5214	131	0.41	0	90
	1000	378266	126103	2614	108	560882	0.11	126305	5559	246	0.16	0	38
	1500	309834	126169	3084	175	414736	0.05	126305	5378	320	0.11	0	32
	3000	221434	126252	3825	360	267347	0.02	126305	4984	511	0.04	0	14
	30000	0	126305	6959	2717	0	0	126305	7928	3566	0	0	0
b15	10	185349	50203	138	58	879949	9.9	69250	5268	944	37.94	61	700
(8931)	50	231739	61128	270	78	823864	1.88	69250	5249	903	13.29	15	530
	100	250348	62495	398	97	799232	0.93	69250	5306	941	10.81	15	484
	500	252861	69093	1357	341	373079	0.09	69250	4406	850	0.23	0	51
	1000	179548	69188	2239	578	250773	0.03	69250	4727	1002	0.09	0	20
	1500	150366	69219	2948	803	198749	0.02	69250	5003	1215	0.04	0	12
	3000	78511	69223	4279	1404	97662	0.01	69250	5872	1910	0.04	0	10
	30000	0	69250	7151	3242	0	0	69250	8545	4244	0	0	0
p35	10	293796	7395	542	40	360685	1.83	8223	1218	55	11.19	0	852
(19964)	50	270197	7813	589	40	342143	0.36	8223	1329	51	5.25	0	219
	100	263782	7869	701	39	359731	0.19	8223	2178	63	4.49	0	160
	500	242401	7988	1095	48	368222	0.04	8223	2959	95	2.94	0	116
	1000	268786	8091	1749	69	285228	0.02	8223	2644	92	1.62	0	87
	1500	216978	8150	1451	74	226846	0.01	8223	1710	92	0.90	0	51
	3000	159280	8196	1594	116	160445	0.00	8223	1887	150	0.33	0	40
	30000	0	8223	506	57	0	0.00	8223	489	55	0.00	0	0
p45	10	80796	3824	142	58	609588	2.68	4569	8048	647	19.47	17	699
(22877)	50	122172	4376	261	68	480945	0.43	4569	7715	265	4.41	0	342
	100	115619	4494	470	82	425049	0.19	4569	6717	240	1.66	0	169
	500	111723	4555	1105	123	326295	0.03	4569	9029	549	0.30	0	50
	1000	100109	4562	1580	216	267548	0.01	4569	7105	947	0.15	0	32
	1500	102996	4563	1703	312	239735	0.01	4569	6102	1153	0.12	0	24
	3000	98656	4568	2898	849	195398	0.00	4569	7082	2276	0.03	0	12
	30000	7513	4569	20185	15750	7513	0.00	4569	20211	15756	0.00	0	1
p78	10	166206	6342	240	101	689810	1.20	7149	7139	418	12.72	8	901
(57608)	50	177257	7039	391	112	501469	0.18	7149	5451	198	1.56	0	722
	100	99069	7147	335	108	373759	0.07	7149	5483	174	0.03	0	96
	500	135763	7148	703	125	321940	0.01	7149	5240	245	0.01	0	5
	1000	150340	7148	1168	180	299284	0.01	7149	7062	413	0.01	0	6
	1500	151226	7149	1344	208	283626	0.00	7149	5099	498	0.00	0	2
	3000	148676	7149	2257	461	243633	0.00	7149	5545	927	0.00	0	2
	30000	0	7149	10590	5859	0	0.00	7149	10560	5840	0.00	0	0

sized the original circuits using a publicly available 45nm cell library [3] in order to use realistic pin-to-pin gate delays. We omit results for circuit c6288, a multiplier with a large number of long non-sensitizable paths. In [15], a specific technique called *Smart-PERT* was developed in order to handle c6288; our implementation does not include this technique.

Table I summarizes the results for $K = 5$ (i.e., five longest sensitizable paths are generated for each gate output in the circuit) and functional sensitization conditions. For each circuit, KLPG and *Opt-KLPG* were run for a number of different path-store size limits π_{max} between 10 and 30,000. We used fault lists consisting of all gates for ISCAS circuits and 1,000 randomly selected gates for ITC-99 and NXP circuits. Column 1 shows the circuit name with its gate count in parentheses. Column 2 and 3 contain π_{max} and the number of overflows encountered by KLPG. In column 4, the sum of lengths of the (up to) five paths identified by KLPG is reported. If the number of overflows is 0, then KLPG's results are optimal and the length in column 4 is maximal; otherwise it can be sub-maximal. Note that the lengths are real numbers rounded to the nearest integer. For instance, KLPG fails to find the optimal solution for p45 with $\pi_{max} = 30{,}000$, but the reported length of 4659 corresponds to the optimal length due to rounding artifacts. The run-times of KLPG can be found in columns 5 and 6. We report the total run-times as well as the times excluding the calls of the SAT-ATPG engine (columns "Excl SAT"), which describe the effort spent in actual search for paths.

From the KLPG results, it can be seen that very small values of π_{max} typically result in a large number of overflows (we will discuss the impact of the overflows on test quality related metrics such as the lengths of the found paths later). The minimal value of π_{max} which results in no overflows varies among the circuits (and does not always grow with circuit size). The number of overflows does not always decrease for larger path store size limits. The reason is that a larger path store may contain a partial path which in turn results in generation of more partial paths, while a smaller path store may simply not contain this path. The run-time difference of our implementation and [15] can be partially attributed to the lack of a number of advanced speed-up techniques such as forward-trimming or Smart-PERT in our implementation. Our aim is to study the impact of π_{max} on the results and not to outperform earlier implementations. A further reason is the amount of time spent for the path sensitization. In contrast to the FAN algorithm which can be seamlessly integrated into the KLPG framework in [15], each path sensitization requires a lengthy process in our case: A path is mapped onto a CMS@ fault for which an ATPG-instance is constructed, which in turn is mapped onto a SAT-instance and solved. Even though the employed SAT-ATPG engine can perform these tasks in a fraction of a second, but the large number of paths to be evaluated results in a large number of SAT-ATPG instances and therefore high run-time requirements. In general, run times rise with growing π_{max}.

Columns 7 through 11 provide information on the optimal algorithm *Opt-KLPG*. Column 7 contains the number of overflows encountered during the complete run of all the iterations. This number is always higher than the number of overflows in column 3, as KLPG constitutes the first iteration of *Opt-KLPG*. Column 8 contains the number of required "re-iterations", that is, iterations after the initial KLPG run. The value reported is the average over all the SDD locations (gate outputs). This number is always 0 if there were no overflows in the initial KLPG run. In column 9, the sum of the lengths of the optimal paths can be found (this value is independent from π_{max}). Columns 10 and 11 contain the run-times. As for KLPG, the run times tend to be larger for a larger π_{max}, yet the trend is less pronounced. This is because a larger π_{max} implies less re-iteration of *Opt-KLPG*. While *Opt-KLPG* requires more time than KLPG for the same π_{max}, it always yields the optimal solution. For instance, circuit c1908 requires $\pi_{max} = 30{,}000$ in order to avoid overflows altogether; this necessitates a total run time of 815 CPU-seconds. However, *Opt-KLPG* achieves the same result for $\pi_{max} = 500$ using 306 CPU-seconds. Furthermore, even running *Opt-KLPG* with $\pi_{max} = 10$ results in a large number of re-iterations per gate, but the total run time of 346 CPU-seconds is still less than for KLPG with $\pi_{max} = 30{,}000$.

The last three columns of Table I evaluate the coverage problems due to sub-optimal path selection by the KLPG algorithm. Column 12 shows the difference between the total length of the paths found by *Opt-KLPG* (column 9) and KLPG (column 4). This difference is rather large for very small values of π_{max}; it tends to be within 1% for $\pi_{max} \geq 100$. This is because many SDDs are not affected by overflows or because the identified paths are not much shorter than the optimal ones. The second-to-last column contains the number of SDD-locations (gates) through which KLPG found less paths than are in existence. This is a severe situation with a significant impact on testability under process variations. It rarely occurs for $\pi_{max} > 10$. The final column shows the number of gates for which the total length of paths identified by KLPG was shorter than the optimum. This corresponds to a coverage impact for the SDD associated with this particular gate. In some of the manufactured instances of the circuit, such SDDs will not be adequately tested. A significant number of gates turn out to be vulnerable to this problem even for large values of π_{max}. Figure 3 shows, in graph form, the number of such gates in circuit p35, accompanied by run times for KLPG and *Opt-KLPG* (which is not affected by this problem).

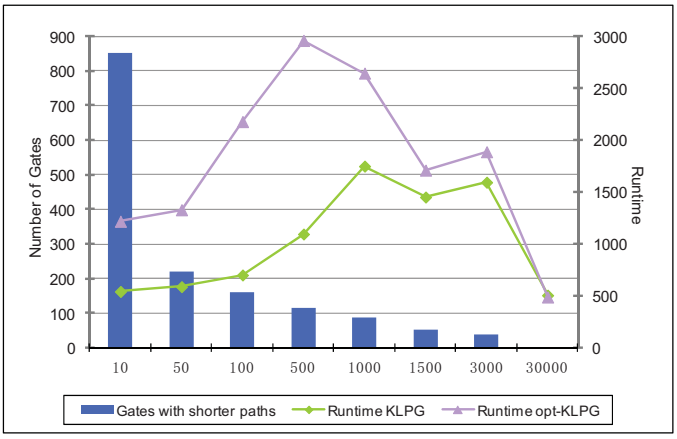


Figure 3. Experimental results for circuit p35. Run times are shown by lines; number of gates are shown by bars.

TABLE II. COMPARISON OF KLPG AND *OPT-KLPG* FOR DIFFERENT MODEL ASSUMPTIONS, CIRCUIT C7552

π_{max}	Non-robust sensitization			Unit delay, functional sens.			Unit delay, non-robust sens.		
	Δ path length [%]	Gates w. less paths	Gates w.short paths	Δ path length [%]	Gates w. less paths	Gates w.short paths	Δ path length [%]	Gates w. less paths	Gates w.short paths
10	3.95	19	1603	4.22	32	1638	4.60	33	1755
50	0.24	0	197	0.40	0	407	0.37	0	354
100	0.10	0	81	0.12	0	118	0.12	0	106
300	0.00	0	2	0.00	0	7	0.00	0	2
500	0.00	0	0	0.00	0	0	0.00	0	0
1000	0.00	0	0	0.00	0	0	0.00	0	0
1500	0.00	0	0	0.00	0	0	0.00	0	0
3000	0.00	0	0	0.00	0	0	0.00	0	0
30000	0.00	0	0	0.00	0	0	0.00	0	0

TABLE III. COMPARISON OF KLPG AND *OPT-KLPG* FOR DIFFERENT MODEL ASSUMPTIONS, INDUSTRIAL CIRCUIT P45

π_{max}	Non-robust sensitization			Unit delay, functional sens.			Unit delay, non-robust sens.		
	Δ path length [%]	Gates w. less paths	Gates w.short paths	Δ path length [%]	Gates w. less paths	Gates w.short paths	Δ path length [%]	Gates w. less paths	Gates w.short paths
10	22.17	25	698	17.12	19	673	21.02	27	676
50	5.59	1	358	4.36	0	310	6.11	2	338
100	2.01	1	180	2.08	0	156	2.79	1	178
300	0.51	1	62	0.67	0	57	0.76	0	58
500	0.33	1	44	0.47	0	47	0.50	0	44
1000	0.13	0	23	0.27	0	32	0.25	0	28
1500	0.08	0	14	0.21	0	27	0.12	0	19
3000	0.01	0	5	0.08	0	13	0.02	0	5
30000	0.00	0	0	0.00	0	0	0.00	0	0

Tables II and III contain the comparison between KLPG and Opt-KLPG (as in the final three columns of Table I) under non-robust sensitization conditions (as opposed to functional sensitization in Table I) for two alternative delay models: gate delays extracted from the technology library and unit-delay model which assumes that each gate has a delay of 1. The influence of the model assumptions tends to be limited. We made similar observation for further benchmarks which are not reported here. It can be seen that path store size affects the optimality of the solution for different assumptions.

V. CONCLUSIONS

K longest path generation is an attractive technique to test small-delay defects under process variations without having to explicitly model the variation mechanism. We studied, for the first time, the optimality of the KLPG algorithm in a systematic way, and discovered that the path-store size limit π_{max} has a significant impact on the results. In practice, conservative setting of this limit to values beyond 1,000 does yield optimal results for most (but not all) of the considered circuits. In contrast, setting π_{max} to very small values, as suggested by some recent publications, appears to have grave impact on the test quality. We suggested a provably optimal algorithm *Opt-KLPG*, which works iteratively and outperforms conventional KLPG with a large path store. Overall, it seems that KLPG optimality should be paid attention to when high test quality is required for circuits affected by process variations.

REFERENCES

[1] ITRS, "International Technology Roadmap for Semiconductors," 2009 Edition, http://www.itrs.net & http://www.itrs.net/Links/2009ITRS/Home2009.htm.

[2] DATE Workshop on Process Variability, "New Techniques for the Design and Test of Nanoscale Electronics", Proc. Design, Automation and Test in Europe, DATE 2009, Nice; France, April 20-24, 2009.

[3] Nangate 45nm Open Cell Library, http://www.nangate.com.

[4] M.L. Bushnell and V.D. Agrawal. *Essentials of Electronic Testing.* Kluwer Academic Publishers, 2000.

[5] J. Chung, J. Xiong, V. Zolotov and J.A. Abraham, "Testability Driven Statistical Path Selection," in IEEE Design Automation Conf., pp. 417–422, June 2011.

[6] A. Czutro, N. Houarche, P. Engelke, I. Polian, M. Comte, M. Renovell, and B. Becker. A Simulator of Small-delay Faults Caused by Resistive-open Defects. Proc. IEEE European Test Symp., pages 113–118, Verbania, I, 2008.

[7] A. Czutro, I. Polian, M. Lewis, P. Engelke, S.M. Reddy and B. Becker, "Thread-Parallel Integrated Test Pattern Generator Utilizing Satisfiability Analysis," Int'l Jour. of Parallel Programming, vol. 38, pp. 185–202, June 2010.

[8] D. Gizopoulos (Ed.), "Advances in Electronic Testing: Challenges and Methodologies," Series: Frontiers in Electronic Testing, Vol. 27, Springer, 2006.

[9] Z. He, T. Lv, H. Li, and X. Li, "An Efficient Algorithm for Finding a Universal Set of Testable Long Paths," IEEE Asian Test Symp., 2010, pp.319–324.

[10] F. Hopsch et al, "Variation-Aware Fault Modeling," in Asian Test Symp., 2011, pp. 87–93.

[11] U. Ingelsson, B.M. Al-Hashimi, S. Khursheed, S.M. Reddy, and P. Harrod. "Process variation-aware test for resistive bridges," IEEE Trans. on CAD, Vol. 28, No. 8, Aug. 2009, pp. 1269–1274.

[12] J.-J. Liou, A. Krstic, Y.-M. Jiang, and K.-T. Cheng, "Modeling, testing, and analysis for delay defects and noise effects in deep submicron devices," IEEE Trans. on CAD, Vol. 22, No. 6, June 2003, pp. 756–769.

[13] J.-J. Liou, A. Krstic, L.-C. Wang, and K.-T. Cheng, "False-path-aware Statistical Timing Analysis and Efficient Path Selection for Delay Testing and Timing Validation," Design Automation Conf., 2002, pp. 566–569.

[14] A. K. Pramanick and S. M. Reddy, "On the detection of delay faults," Proc. IEEE Int'l Test Conf., 1988, pp. 845–856.

[15] W. Qiu and D.M.H. Walker, "An Efficient Algorithm for Finding the K Longest Testable Paths Through Each Gate in a Combinational Circuit," in Int'l Test Conf., pp. 592–601, 2003.

[16] Y. Sato, S. Hamada, T. Maeda, A. Takatori, Y. Nozuyama, and S. Kajihara. "Invisible Delay Quality – SDQM Model Lights up What Could not be Seen," Proc. Int'l Test Conf. 2005, Paper 47.1.

[17] M. Sauer, J. Jiang, A. Czutro, I. Polian and B. Becker, "Efficient SAT-based Search for Longest Sensitisable Paths," in IEEE Asian Test Symp., 2011.

[18] Y. Shao, S.M. Reddy, I. Pomeranz, and S. Kajihara, "On Selecting Testable Paths in Scan Designs," Jour. Electronic Testing, vol. 19, no. 4, pp. 447–456, 2003.

[19] A. Srivastava, D. Sylvester, and D. Blaauw, "Statistical Analysis and Optimization for VLSI: Timing and Power," Springer, New York, NY, USA, 2005

[20] D. Sylvester, K. Agarwal, and S. Shaha, "Variability in Nanometer CMOS: Impact, Analysis, and Minimization," Integration, the VLSI Journal, Vol. 41, No. 3, May 2008, pp. 319–339.

[21] C. Visweswariah, "Fear, uncertainty and statistics," Proc. Int'l Symp. on Physical Design, 2007, p. 169.

[22] M. Yilmaz, K. Chakrabarty, and M. Tehranipoor. "Interconnect-aware and Layout-oriented Test Pattern Selection for Small-delay Defects," Proc. Int'l Test Conf. 2008, Paper 28.3.

Path Selection for High-Quality Small Delay Defect Testing

Dong Xiang

School of Software,
Tsinghua University,
Beijing 100084, China
dxiang@tsinghua.edu.cn

Abstract —Testing for small-delay defects (SDDs) requires fault-effect propagation along the longest testable paths. However, the identification of the longest testable paths requires high CPU time. We present a new technique for identifying the longest testable paths through each gate for SDD testing. The new path selection technique identifies b-f segments for the two-frame circuit model based on fanout-free segments, according to which the longest testable paths are selected without any backtrack. The resulting test patterns sensitize the longest testable paths that pass through each SDD site. Simulation results for a set of IWLS2005 benchmark circuits demonstrate the effectiveness of this method.

Keywords —Broadside delay testing, critical testable path, path selection, small delay defect.

I. Introduction

Two types of fault models are commonly advocated for delay testing in logic circuits, namely the transition fault model and the path-delay fault model. The path-delay fault model is based on the cumulative delay through a target path. Although this fault model is recognized to be very effective for delay testing, the exponential number of path-delay faults makes it impractical for large circuits if all paths are targeted. It is therefore necessary to target selected critical or the longest testable paths for realistic circuits. The transition fault model is more tractable, since it assumes a gross defect and a fault effect that is large enough to cause a timing failure irrespective of the path used for error propagation.

The traditional transition-fault tests are generated assuming a fixed cycle time for each clock domain. This implies that a delay defect will be detected only when it causes a transition to reach an observation point with a delay that exceeds the slack of the affected path. The slack of a path is a measure of how close is its delay relative to the clock cycle time. A path is deemed to be *long* if its slack is less than a predetermined threshold, such as 90% of the clock cycle time. With continued process scaling, aggressive clocking, and the increased use of power management strategies on chip, more paths in a design tend to become long paths.

A small-delay defect (SDD) may escape a test when the fault effect is propagated along a short path. SDDs have emerged as a serious concern for nanoscale CMOS technologies [1, 2, 4, 6, 19]. Short paths have large slacks, hence they are useful only for the detection of gross-delay defects. For a given defect size, the path used to activate and propagate the fault is therefore very important. It is obvious that if the delay-defect size is greater than the slack of the shortest functional path S_1, the circuit will malfunction regardless of the location of the fault. The path-oriented transition fault testing looks attractive [10, 18], which are good to detect SDDs unlike the traditional transition fault testing schemes. This fault can be easily detected using the transition fault model. If the delay defect size is smaller than the slack of the longest functional path S_2, the circuit will operate correctly despite the fault, and the fault is redundant. When the defect size is between S_1 and S_2, the fault will affect the normal operation of the circuit if the summation of the longest functional path passing through the fault site and the defect size is greater than the system clock period. These faults should be activated and propagated along the longest testable paths passing through them.

The problem of finding the longest testable paths in the circuit is a computation bottleneck. In this paper, we present a new technique for quickly identifying the longest testable paths to accelerate test generation for SDDs. The resulting test patterns sensitize the longest testable paths that pass through each SDD site. However, the sensitization of all such paths leads to large pattern counts. For example, it has been reported recently that the use of a commercial timing-aware ATPG tool for microprocessor blocks led to a 15x increase in pattern count. An efficient test compaction scheme for SDD testing is essential. Details for test compaction scheme will not be presented in this paper for space limit. We consider broadside testing (launch-on-capture) in this work because of its use in industry.

The rest of the paper is organized as follows. Section II describes related prior work. Section III presents the technical preparation for the path-selection method. The path selection procedure is presented in Section IV. Experimental results are presented in Section IV. Finally, Section V concludes the paper.

II. Related Prior Work

While the testing for SDDs has received much attention recently [1, 2, 19], this problem was first identified and addressed as early as in 1988 [8]. In recent work, Ahmed *et al.* [1] preprocessed scan flip-flops based on the path slacks and the detectable delay-defect size. A new delay defect-size metric based on the affected path length and required increase in test-clock frequency was developed. In [2], a new transition fault model ALAPTF (as late as possible transition fault) was proposed to satisfy the launch condition needed to capture errors due to small-delay defects accumulated along the transition-launch segment. However, it did not consider the propagation path from the fault site to an observation point. Lin *et al.* [6] proposed a timing-aware ATPG method based on timing information derived from the standard delay format (SDF) files. A local ATPG technique was used in Gurumurthy *et al.* [3] to generate instruction sequences for path-delay faults, which are then mapped to an entire pro-

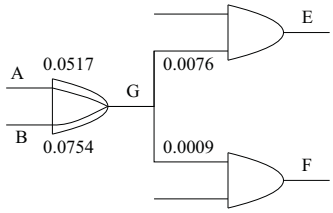


Figure 1: The delay model based on the standard delay format (SDF) files.

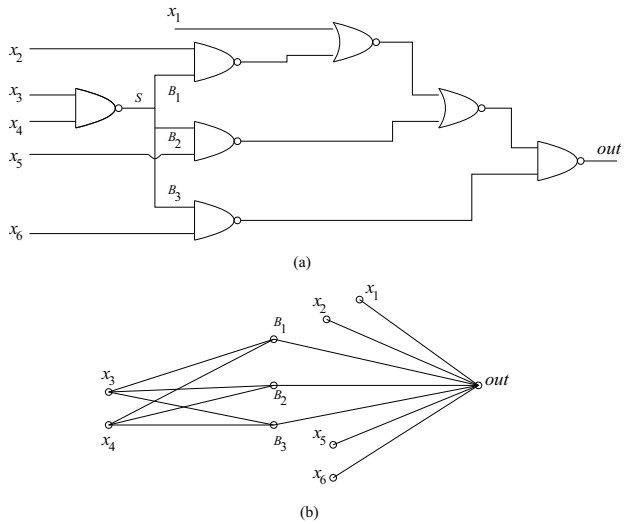


Figure 2: The segment partitioning scheme for path selection: (a) the circuit, and (b) the segment graph.

cessor in a global manner. An ATPG-based path-selection scheme was combined with the ATPG method.

In other related work, Sharma and Patel in [14] proposed a graph-theoretic and ATPG-driven method to select a small set of longest testable paths that cover all gates. Qiu and Walker [11] proposed a method to find the k-longest testable paths that pass through each gate. The technique described by Tayade and Abraham [15] extracts logic and timing constraints, according to which a SAT-based constraint-satisfaction problem is formulated to estimate the maximum path delay in the presence of coupling noise. A drawback of all the above methods [14, 11, 15] is that they attempt to identify testable paths using ATPG schemes, which can be time-consuming due to the need for backtracking. Shao *et al.* [13] proposed a path selection scheme based on path expansion. This technique relies on backtracking during stepwise expansion, therefore it also suffers from high run time.

In more recent work, Yilmaz *et al.* [19] proposed a test-selection scheme, where the most effective patterns are selected from an n-detect test set generated using timing-unaware test generator. This method suffers from the drawback that it relies on the availability of a high-quality repository test set.

Xiang, *et al.* [17] proposed a dynamic test compaction scheme for SDDs based on the selected longest testable paths. However, the method in [17] selected the longest testable paths from all testable paths, therefore, it is unable to handle large enough circuits.

identify-bf-segment()
{

1. Partition the circuit into fanout-free segments S_1, S_2, ..., S_n.
2. For each segment with the source a PI or PPI, traverse the segment graph from the input to all reachable outputs. Record all successive segments for each segment.
3. For each segment S_i, do 4) and 5),
4. for each successive segment S_j, check whether justification of all off-path line sensitization values for two segments produces a conflict in the two-frame circuit.
5. If justification of all off-path line sensitization values for two segments produces conflicts, both segments establish a b-f segment.

}

Figure 3: Identification of b-f segment pairs.

III. Preparation for the Longest Testable Path Selection

The methods in [11] and [13] presented techniques for path selection in broadside path delay testing, which may still have difficulty in selecting paths for larger circuits. The simplest way to estimate path length (or path delay) is to add the delays of the gates and interconnects constituting the path. However, it is difficult in nanoscale technologies to approximate the delay of a gate using a single number. The gate delay is especially difficult to ascertain before chip fabrication because of process variations. In this work, we compute path delays using standard delay format (SDF) files generated in industry using a recent process technology [7].

It is not necessary for a path with the most delay to have the most number of gates. As shown in Fig. 1, the delays between the two inputs A and B to its output G are 0.0517 and 0.0754 nanosecond, respectively. The delays of the interconnects between the fanout stem G to inputs of its two successors E and F are 0.0076 and 0.0009 nanosecond, respectively. Therefore, the gate counting scheme can be very inaccurate. The SDF files provide three separate delay data (a,b,c) for a gate or a connection, where a, b and c represent the minimum, normal, and maximum delay. Our method uses the maximum delay to estimate the delay of a path (in most cases, the three values are the same).

The longest testable path identification is carried out using the two-frame circuit model. Let us consider the rising transition fault at a node l. Our method sets a value constraint at the source node of the path in the second frame, which must set l' to 1, while the source node in the first frame is set to the complement value in order to generate a rising transition at l.

Our method still identifies the b-f pairs like many other methods [13]. The method in [13] identified the b-f segments in the enhanced scan circuits, where a b-f segment includes two gates. One gate is a successor of the other. They also extended the front line to the fanout-free region. All sensitization conditions, that correspond to the two fanout-free segments, are in conflict. Therefore, any path, that contains the b-f pair, is untestable. Identification of b-f segments can greatly reduce the CPU time to find the longest testable paths.

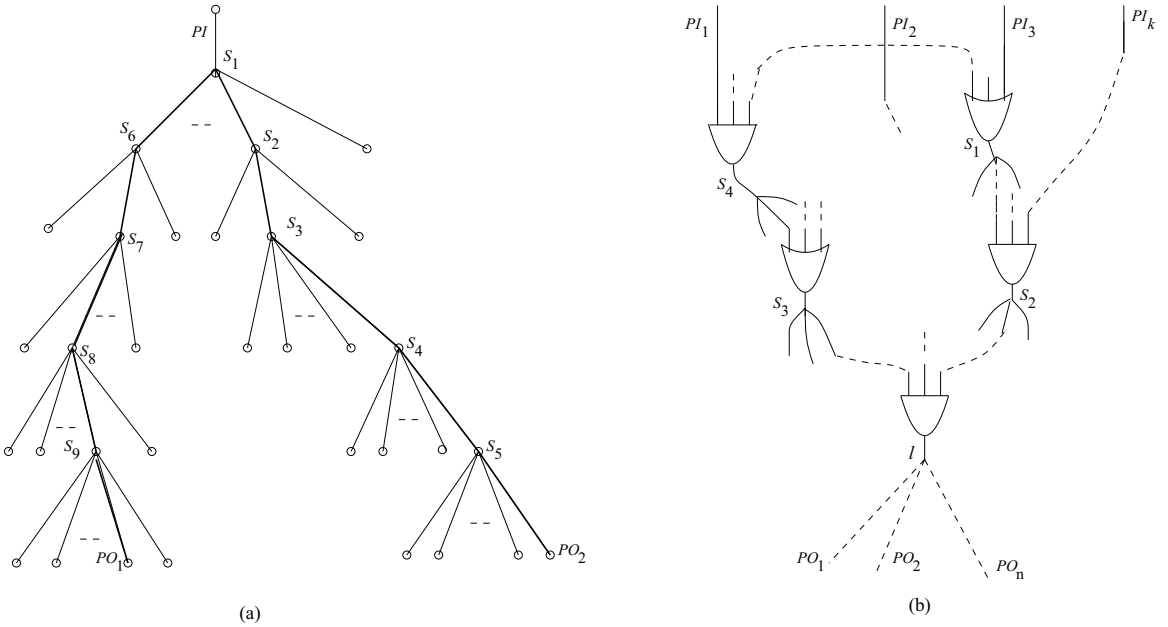


Figure 4: The longest testable path selection scheme in broadside delay testing for small-delay defects.

Our method extends the b-f pair definition to the two-frame circuit model. Any pair of segments S_1 and S_2 in the second circuit frame is a b-f pair if all sensitization values of two fanout-free segments in the second frame cannot be justified simultaneously to PIs in both frames and PPIs in the first frame, where nodes related to S_1 are successors of the gates of S_1.

The segment graph can be defined as follows. All nodes of a segment graph include all circuit inputs and circuit outputs, and the fanout branches. Each node of the segment graph is represented by the source node of the segment. A node S_1 is connected another node S_1 if the sink of the segment S_1 is connected to the source of the segment S_2. The output is connected with all segments, where the output is their sink. The segment graph of the circuit as presented in Fig. 2(a) is given in Fig. 2(b).

As shown in Fig. 3, the procedure to identify b-f segments is presented. Any unsatisfiable segment is itself a b-f segment. The proposed procedure just checks any segment S_i and its successive segment S_j. If justification of the off-path lines for both segments introduces some conflict, both segments S_i and S_j establish a b-f segment. It is found that the CPU time to identify all b-f segments is much less than that for the longest testable path identification. The b-f segment identification process can save much CPU time for path selection. The longest testable path identification is also completed based on fanout-free segments. As shown in Fig. 2(a), x_3-S and B_1-*out* establish a b-f pair. However, the path x_3-B_2-*out* is testable. Generally, the b-f pair identification is an extension of the technique in [13].

IV. The Procedure to Identify the Longest Testable Paths

We would like to select the longest testable paths of the most delay without identifying all testable paths. The proposed method does not need any backtracking. Our method selects the longest testable path for each node in the circuit.

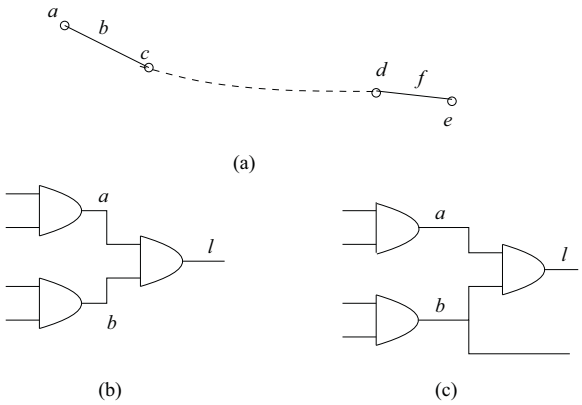


Figure 5: Techniques to accelerate path selection.

It is easy to estimate the length or delay of the longest sub-path from each node to a primary output (PO) or pseudo-primary output (PPO). This can be completed by a forward traversal of the circuit. As shown in Fig. 4, let us consider identifying the longest testable path starting at a primary input PI. The proposed method checks whether the path segment for PI to the first fanout S_1 is sensitizable. Assume that the number of fanout branches at S_i is f_i. It is easy to find the longest sub-path from a PI to a node, or for the node to a PO/PPO. If not, no testable path starting at PI is available. Suppose the segment PI-S_1 is testable. Our method selects the longest paths from the f_1 paths. Let the longest sub-path from S_1 be S_1-PO_2. There exist other fanouts S_1, S_3, S_4 and S_5 in the sub-path. Our method first checks whether the path contains any b-f segment. If so, it is not necessary to check whether it is testable. Assume the longest path PI-S_1-PO_2 is found to be untestable. Our method identifies the longest testable path among the $\sum_{i=1}^{i=9} f_i - 1$ candidate paths.

Let the path PI-S_1-PO_1 be the longest path among the remaining $\sum_{i=1}^{i=9} f_i - 1$ paths. Our method first check whether any b-f segment is contained in the path. Assume that the

path is identified untestable. Assume that the path PI-S_1-PO_1 contains the fanouts S_6, S_7, S_8 and S_9. We now attempt to select the longest path among the remaining $\sum_{i=1}^{i=9} f_i - 2$ paths. The above process continues until the longest testable path has been identified. The above scheme can also be used to select k-longest testable path for each node.

An important problem is how to select the longest testable path that contains any internal node l. It is easy to get the length of the longest sub-path from a PI or pseudo-primary input (PPI) to each node l. A backward traversal scheme is used from l to PIs or PPIs. We mark all nodes during the backward traversal from l to the inputs that reach l. The marked region is indicated in Fig. 4(b). We obtain the length of the longest path from each PI or PPI among PI_1, PI_2, ..., PI_k to a PO or PPO, which contains the node l. As shown in Fig. 4(b), let PI_2 be the input that is on the longest path to a PO or PPO, which contains l. Our method checks the longest testable path from PI_2 to a PO that contains l using the technique proposed earlier in this section. However, only the marked region from one of PI_1, PI_2, ..., PI_k to l is considered. We check the longest path from one of PI_1, PI_2, ..., PI_k to a PO among the remaining paths, where the sub-path from one of PI_1, PI_2, ..., PI_k to l only traverses the marked area.

The proposed method to select k-longest testable paths is fundamentally different from [11]. The method in [11] finds the region that reach an internal node l, which helps in the identification of the PI or PPI that is the start point of a path with the most delay. This method also checks whether the path is testable. However, it is quite possible for the path to move out of the marked region because it may not contain l. The method in [11] checks the path with the most delay among all paths whose source is one of PI_1, PI_2, ..., PI_k. In many cases, this method may still check a path that does not contain l. This may necessitate considerable wasteful computation, and lead to high CPU time. Our method always checks the paths that contain l. Another important difference is that we add a fanout-free segment each time; however, the method in [11] adds a new gate to the path each pass. Therefore, the proposed technique can also reduce the CPU time needed to select paths.

The proposed method also avoids redundant computation. The b-f path segments are identified first [13] in the two-frame circuit model. A b-f path segment contains two sub-path ac and cd as shown in Fig. 5(a), where both sub-paths cannot be sensitized simultaneously. Any path that contains a b-f pair is untestable. Our method identifies all b-f pairs in the two-frame circuit model, which includes two fanout-free segments. The amount of b-f path segments in the two-frame circuit model can be much more than that in the enhanced scan circuit, as stated in [13]. This technique can significantly reduce the path-selection time.

The selected testable path can be used for immediate successors. As shown in Fig. 5(b), let the longest testable path that contains a and b have been identified. We would like to find the longest testable paths for node l. We can select the longest testable paths from the 2 selected paths because both paths, which contain l, must pass through a or b. For the case as shown in Fig. 5(c), there exists a fanout branch at b that does not pass through l. Assume that the longest testable paths that contain a and b have been found. We now identify the longest testable paths that contain l. Our method orders two longest testable paths according to their delay, and the path with the most delay, which contains l, is selected directly. The above techniques can reduce the CPU time to select paths significantly. We are considering more efficient and powerful techniques to reduce CPU time for path selection in further work of this paper.

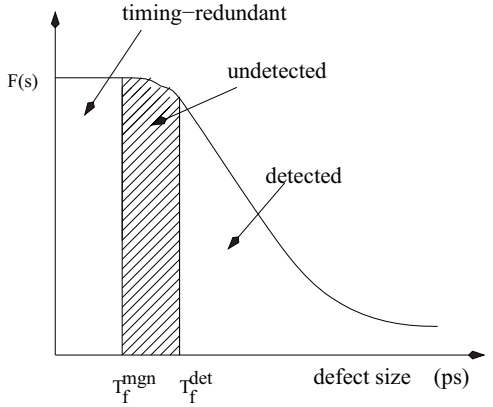


Figure 6: The scheme to estimate the SDQL [12].

V. SDQL Measure to Evaluate the Proposed Timing-Aware ATPG Tool

We present the SDQL (statistical delay quality level) measures [12] for the proposed timing-aware ATPG tool. It can also be used to evaluate test quality of other ATPG tool including the commercial timing-aware ATPG tools. The SDQL measure is estimated as follows,

$$SDQL = \sum_{f \in F} \int_{T_f^{mgn}}^{T_f^{det}} F(s) \cdot ds, \tag{1}$$

$$F(s) = \lambda \cdot e^{-\lambda \cdot s}, \tag{2}$$

$$T_f^{mgn} = T_s - PD_f^s, \tag{3}$$

$$T_f^{det} = T_t - PD_f^a. \tag{4}$$

where $F(s)$ is the delay defect distribution function (the probability of SDDs of defect size s), T_f^{mgn} and T_f^{det} are estimated by Equations (3) and (4) for $f \in F$, and F is the detectable SDD set. Here, PD_f^a is the delay of the longest sensitized path corresponding to the test set, and PD_f^s is the delay of the longest testable path that passes f. Parameters T_s and T_t are the system clock period, and the test clock period, which are set to the same value for at-speed testing. The defect size is the incremental delay caused by a delay. Assume that the delay defect distribution function is derived from fabrication process. The area of the shadowed region is the SDQL measure corresponding to SDD f as shown in Fig. 6. More details for SDQL estimation can be found from [6, 12]. In any case, the lower the SDQL value is, the higher the delay-test quality is.

Each detectable SDD is sensitized along the selected longest testable path that contains it based on the proposed method, therefore, T_f^{mgn} and T_f^{det} are the same for the detectable SDD based on the selected longest testable path set.

This makes the area of the shadowed region as shown in Fig. 6 zero. That is, the SDQL measure for any detectable SDD based on the proposed method is zero and the area of the shadowed part as shown in Fig. 6 is also zero. Anyway, the SDQL measure is estimated based on the SDF file, which cannot be accurate. Also, the proposed method to identify the longest testable path cannot be exact because of process variations. However, the proposed path selection procedure can still identify the longest testable paths for all nodes when more accurate standard, such as, SDF, is given. Anyway, the SDF files provide accurate enough information, which is very popular in industry now. In any case, the SDF file can provide much more accurate delay information than that provided by gate count.

VI. Experimental Results

We have implemented the proposed ATPG technique on a DELL Precision 490 workstation. An upper limit of 100 is set on the number of testable paths explored for a baseline method that enumerates paths (the path enumeration ATPG scheme). Experimental results are presented in Table 1. In Table 1, the SDQL refers to the statistical delay quality level, *i.e.*, the probability of detecting small-delay defects [12]. When calculating SDQL, we use the method in [12] and use nominal delays for gates and interconnects based on SDF files to compute the path delay. The clock period is determined from the longest testable path of the circuit.

Table 1 shows the number of longest testable paths selected for compact ATPG of small-delay defects and the CPU time to select the paths for the IWLS2005 circuits. The SDF files are available for the eleven IWLS2005 circuits.

Table 1 presents comparison between the proposed compact ATPG method and the conventional small-delay ATPG method with path enumeration, and a commercial timing-aware ATPG tool. Columns 4, 5, and 6 present fault coverage, CPU time, and the SDQL value for the baseline that uses path enumeration and searching. The path enumeration ATPG method is implemented with up to 100 paths being evaluated for each SDD. Columns 7-12 present fault coverage of the proposed method, the CPU time (seconds) for compact ATPG, the SDQL measure for the generated test set, the number of selected paths, CPU time to select the paths.

The last three columns present fault coverage, the number of test pairs, and SDQL measure of the timing-aware commercial ATPG tool for SDDs. It looks that the proposed method gets the best SDQL measure for all circuits and the commercial ATPG tool gets much better SDQL measure than the path enumeration scheme. The SDQL measure of the proposed method is 0 in all cases, which highlights the effectiveness of the proposed SDD testing method. The reason for this is that any small-delay defect is propagated along the longest testable path that contains it. The most important reason should be that the delay of the sensitized path is equal to the delay of the actual longest testable path that contains each SDD because the system clock period and the test clock period are the same. One may think that the SDQL measure estimation is inaccurate, however, if provided more accurate standard instead of SDF, our method can still get the longest testable paths that contain each SDD. The SDQL measures for the path enumeration scheme, and the commercial tool are not zero for all circuits.

Table 1 highlights the effectiveness of the proposed test-compaction scheme. The transition ATPG tool, the path-enumeration scheme and the proposed method for SDDs obtain the same fault coverage for all circuits. This is reasonable because all testable transition faults are still testable based on the path-enumeration scheme and the proposed test compaction scheme. It is also reasonable that the met get different fault coverages from our method because many factors, such as fault collapsing, influence fault coverage. Columns 2 and 3 present the number of flip-flops and the number of faults. Columns 4, 5 and 6 show the fault coverage, the CPU time (seconds) and the SDQL measure for the path enumeration ATPG scheme. Columns 7-12 provide the fault coverage, the number of test vector pairs, CPU time for the proposed compact test generation method, the SDQL measure, the number of selected paths, and the CPU time (seconds) to select the paths of the proposed method. Columns 13 and 14 give the fault coverage and the number of test pairs for the compact transition fault ATPG method developed in our group. Finally, the last three columns present the fault coverage, the test set size and the SDQL measure of the timing-aware transition commercial ATPG tool.

It is found from Table 1 that the number of test pairs decreases significantly with the proposed test compaction scheme compared to the path enumeration ATPG. The path enumeration scheme does not consider any test compaction. Moreover, the proposed method is combined with an efficient fault simulation scheme, which also contributes to the smaller number of test patterns. Compared to the timing-aware commercial ATPG tool, the proposed method generates less test set for all circuits except DMA. The proposed method provides much smaller test sets than the commercial tool for circuits ac97, tv80, systemcaes, usb_funct, aes_core, wb_conmax, mem_ctrl, and pci_bridge32. As shown in Table 1, the ATPG results for vga are still not available at this point.

Compared to the tATPG tool (a transition ATPG tool), it is found that the proposed SDD compact ATPG tool based on the selected longest testable paths still generates much bigger test set for almost all circuits. The transition ATPG tool was developed by our group, which produces bigger test sets for usb_funct, wb, pci, and ethernet. The most important reason is that the transition ATPG tool did not include any static test compaction scheme at this point. However, the proposed SDD compact ATPG tool includes some effective static test compaction techniques. The CPU time spent on static test compaction is much less than that for dynamic test compaction.

VII. Conclusions

Fault effects of SDDs should be propagated along the longest testable paths for high-quality SDD testing. An effective path selection scheme was proposed based on SDF files, which is popular in industry. The new path selection scheme introduced a new b-f identification technique based on fanout-free segments, according to which most untestable paths can be filtered very quickly. The new path selection scheme does not need any backtracking. Experimental results for benchmark circuits, including some of the largest circuits available publicly, show that the proposed method achieves high test quality in reasonable CPU time, and with very low pattern count. Further work is to propose more effective path selection scheme by avoiding many aimless trials, which can greatly reduce CPU time.

Table 1: Test Quality Comparison with the Previous Methods

Circuit	FFs	flts	path enumeration			proposed						tATPG		com.		
			FC(%)	CPU	SDQL	FC(%)	vec	CPU_1	SDQL	paths	CPU	FC	vec	FC(%)	vec	SDQL
tv80	359	29116	80.69	989	0.502	80.69	947	311	0	8021	3761	80.69	193	79.98	1348	0.115
systemcaes	190	39665	86.45	373	0.301	86.45	400	2515	0	11872	192	86.45	319	90.37	1434	0.059
mem_ctrl	1083	42734	95.4	1010	0.724	95.4	639	558	0	12683	156	95.4	380	53.19	989	0.247
usb_funct	1746	66918	97.98	1989	0.356	97.98	418	1102	0	19909	2332	97.98	440	91.78	1774	0.025
ac97_ctrl	2199	82462	91.53	802	0.584	91.53	277	1093	0	22155	326	91.53	204	85.83	1079	0.101
aes_core	530	70210	96.0	9120	0.592	96.0	835	63069	0	23823	5623	96.0	475	98.09	2676	0.008
dma	2192	106440	92.06	5100	0.462	92.06	1875	24454	0	32699	1246	92.06	1197	91.42	1661	0.083
wb_conmax	3359	143296	94.02	10995	0.889	94.02	1075	63374	0	49255	2084	94.02	1253	93.12	3263	0.104
pci_bridge32	2324	139968	96.1	10795	1.194	96.1	864	7076	0	37072	1361	96.1	965	90.94	2862	0.051
ethernet	10554	480764	98.23	133399	1.956	98.23	3753	476828	0	119876	21389	98.23	5834	92.97	8489	0.1
vga_lcd	17079	785428	97.03	221810	1.281	-	-	-	0	200022	46457	97.03	8502	94.14	10826	0.027

References

[1] N. Ahmed, M. Tehranipoor, and V. Jayaram, "Timing-based delay test for screening small-delay defects," in *Proc. of ACM/IEEE Design Automation Conference*, paper 19.2, 2006.

[2] P. Gupta and M. S. Hsiao, "ALAPTF: A new transition fault model and the ATPG algorithm," in *Proc. of IEEE Int. Test Conference*, pp. 1053-1060, 2004.

[3] S. Gurumurthy, R. Vemu, J. A. Abraham, and D. G. Saab, "Automatic generation of instructions to robustly test delay defects in processors," in *Proc. of IEEE European Test Symposium*, 2007.

[4] S. Hamada, T. Maeda, A. Takatori, Y. Noduyama, and Y. Sato, "Recognition of sensitized longest paths in transition delay faults," in *Proc. of IEEE Int. Test Conference*, paper 11.1, 2006.

[5] S. Kajihara, S. Morishima, A. Takuma, X. Wen, T. Maeda, S. Hamada, and Y. Sato, "A framework of high-quality transition fault ATPG for scan circuits," in *Proc. of IEEE Int. Test Conference*, paper 2.1, 2006.

[6] X. Lin, M. Kassab and J. Rajski, "Test generation for timing-critical transition faults," in *Proc. of 16th IEEE Asian Test Symposium*, pp. 487-492, 2007.

[7] M. Orshansky and A. Bandyopadhyay, "Fast statistical timing analysis handling arbitrary delay correlations," in *Proc. IEEE Design Automation Conf.*, pp. 337-342, 2004.

[8] E. S. Park, M. R. Mercer, and T. W. Williams, "Statistical delay fault coverage and defect level for delay faults," in *Proc. IEEE Int. Test Conference*, 1988, pp. 492-499.

[9] S. Paul, H. Mahmoodi, and S. Bhunia, "Low-overhead F_{max} calibration at multiple operating points using delay sensitivity-based path selection," *ACM Trans. Design Automation of Electronic Systems*, vol. 15, no. 2, article 19, Feb. 2010.

[10] I. Pomeranz and S. M. Reddy, "Transition path delay faults: A new path delay fault model for small and large delay defects," *IEEE Trans. on Very Large Scale Integration Systems*, vol. 16, no. 1, pp. 98-107, Jan. 2008.

[11] W. Qiu, *et al.*, "K-longest paths per gate test generation for scan-based sequential circuits," in *Proc. of IEEE Int. Test Conference*, pp. 223-231, 2004.

[12] Y. Sato, *et al.*, "Invisible delay quality–SDQM model lights up what could not be seen," in *Proc. of IEEE Int. Test Conference*, paper 47.1, 2005.

[13] Y. Shao, *et al.*, "On selecting testable path in scan designs," *Journal of Electronic Testing: Theory and Application* vol. 19, pp. 447-456, 2003.

[14] M. Sharma and J. H. Patel, "Finding a small set of the longest transition paths that cover every gate," in *Proc. of IEEE Int. Test Conference*, paper 34.1, 2002.

[15] R. Tayade and J. A. Abraham, "Critical path selection for delay test considering coupling noise," in *Proc. of 13th European Test Symposium*, pp. 113-124, 2008.

[16] H. Turakhia, W. Daasch, M. Ward, and J. Van Slyke, "Silicon evaluation of longest path avoidance testing for small-delay defects," in *Proc. of IEEE Int. Test Conference*, 2007.

[17] D. Xiang, B. Yin, and K. Chakrabarty, "Compact test generation for small delay defects using testable path information," in *Proc. of Asian Test Symposium*, Nov. 2009.

[18] K. Yang, K. T. Cheng, and L. C. Wang, "TranGen: A SAT-based ATPG for path-oriented transition faults," in *Proc. of ASP-DAC*, pp. 92-97, 2004.

[19] M. Yilmaz, K. Chakrabarty, and M. Tehranipoor, "Test-pattern selection for screening small-delay defects in very-deep submicron integrated circuits," *IEEE Trans. on Computer-Aided Design*, vol. 29, pp. 760-773, May 2010.

Robust PDFs Testing of Combinational Circuits based on Covering BDDs

A. Matrosova, E. Nikolaeva, S. Ostanin
Tomsk State University,
Tomsk, Russia
e-mail: mau11@yandex.ru

V. Singh
Indian Institute of Science
Bangalore, India
e-mail: viren@serc.iisc.ernet.in

***Abstract*—A method of deriving test pair v_1, v_2 for robust path delay fault (PDF) of special combinational circuit is suggested. Circuit is obtained by covering binary decision diagram (BDD) with look up table (LUT) based configurable logic blocks (CLBs). It is found out that for each path of the circuit there exists a test pair v_1, v_2 on which delay fault manifests itself as robust. Triplets v_1, v_2, v_1 or v_2, v_1, v_2 detect delays of both rising and falling transitions of the same path. Integrating triplets of all paths we derive test T that detects any path delay fault of the circuit, single and multiple.**

Keywords: path delay fault (PDF), robust PDF, binary decision diagram (BDD), design for testability, FPGA.

I. INTRODUCTION

Delay testing has become very important problem with development of nanometer technologies. The objective of delay testing is to detect timing defects degrading the performance of a circuit. Path delay fault model is considered more preferable for detection of timing defects.

To observe delay defects, it is necessary to generate and propagate transitions in the circuit input. This requires application of a pair of vectors v_1, v_2. The first vector v_1 stabilizes all signals in the circuit. The second vector v_2 causes the desired transition in the input of a circuit. Take into account that delays of falling transition and rising transition along the same path from a primary input to a primary output in a circuit may be different. In the general case it is necessary a pair of vectors v_1, v_2 for each kind of transitions of a path. We will call a pair of vectors on which PDF manifests itself as PDF test pair (for the corresponding transition along the path). Single and multiple PDFs are distinguished.

In accordance with the conditions of fault manifestation single PDFs are divided into robust and non robust [1, 2]. PDF is robust if there is a test pair on which the fault manifestation does not depend on delays of other circuit paths.

PDF is non robust if a manifestation of the fault on a test pair is possible only when all other paths of a circuit are fault free.

It is very important to provide testability during circuit design. Circuits derived from BDDs as a rule are implemented with multiplexors. Their testability is investigated under different fault models [3-6] but the approaches suggested did not provide100% testability. In the paper [7] simple transformation of a circuit is suggested that guarantees 100% testability for both single stuck-at fault (SAF) and PDF models. The circuits are derived from BDDs with using multiplexors. The size of a circuit is directly proportional to the given BDD size. Optimization connected with variable ordering directly transfers to the circuit size. Disadvantage of this approach consists of using additional input.

In the paper [8] circuits are derived by covering BDDs with CLBs. In this paper is revealed that each single stuck-at fault at the CLB pole is equivalent either single stuck-at fault of the proper internal node of BDD or 10 (01) faults of edges coming from internal node. It is also determined that each single stuck-at fault at the CLB pole is detectable. It means that the circuit guarantees 100% testability under SAF. Moreover test for all multiple stuck-at faults may be derived directly from test for single stack-at faults. Test for multiple stuck-at faults is 2.5 times longer [8] than test for single stuck-at faults (at the average).

In this paper we show that circuit obtained from BDD by covering CLBs guarantees 100% testability for PDFs without an additional input. A size of the circuit is directly proportional to the given BDD size. Optimization connected with variable ordering directly transfers to the circuit size. Moreover the lengths of tests for all PDFs of circuits considered in this paper and the sizes of these circuits as a rule less than ones in circuits implemented with multiplexors [7].

In Section 2 a problem of deriving special combinational circuits is discussed. In Section 3 test pair is found on which PDF manifests itself as robust for rising and falling transitions. In Section 4 experimental results are given.

II. A COMBINATIONAL CIRCUIT DESIGN

It is well known that BDD is a directed acyclic graph based on using Shannon decomposition in each non terminal node v:

$$f_v = \overline{x_i} f_v^{x_i=0} + x_i f_v^{x_i=1}$$
$$f_v^{x_i=0} = f_v(x_1,...,x_i = 0,...,x_n), \quad (1)$$
$$f_v^{x_i=1} = f_v(x_1,...,x_i = 1,...,x_n))$$

Here f_v is the function corresponding to the node **v**, dashed edge points to $f_v^{x_i=0}$ and solid edge points to $f_v^{x_i=1}$. A BDD is called ordered if variables are encountered in the same order on all paths connecting the BDD root with a terminal node. A BDD is reduced if it does not contain either isomorphic subgraphs nor nodes so that $f_v^{x_i=0} = f_v^{x_i=1}$. Reduced and ordered BDD is a canonical representation of Boolean function for the chosen order of variables [9].

Any path that connects the BDD root with the 1 terminal node originates the product of the Disjoint Sum of Products (DSoP) of a function *f* that is represented with this BDD. DSoP is a sum of products in which any two product cubes don't intersect.

Let $F = \{f_1,...,f_m\}$, be the system of Boolean functions describing a combinational circuit behavior. Derive BDD using the same order of variables for each Boolean function from *F*. Join isomorphic subgraphs in the different BDDs. Combine BDDs 1 terminal nodes into one 1 terminal node and their 0 terminal nodes into one 0 terminal node. Due to we obtain the graph with *m* roots and two terminal nodes. This graph jointly represents a system of *m* Boolean functions. It is called Shared BDD [10]. Without loss of generality we will consider further system with one function.

In Fig. 1 a BDD for one output Boolean function is shown. For each path connecting the BDD root with the 1 terminal node define the product of the DSoP. The DSoP of the function *f* is as follows.

$$f = x_1\bar{x}_2\bar{x}_3 \vee x_1\bar{x}_2x_3\bar{x}_4x_5 \vee x_1\bar{x}_2x_3x_4\bar{x}_5 \vee x_1x_2\bar{x}_4x_5 \vee$$
$$x_1x_2x_4\bar{x}_5 \vee \bar{x}_1\bar{x}_2\bar{x}_4x_5 \vee \bar{x}_1\bar{x}_2x_4\bar{x}_5 \vee \bar{x}_1x_2\bar{x}_3\bar{x}_4x_5 \vee \quad .$$
$$\bar{x}_1x_2\bar{x}_3x_4\bar{x}_5 \vee \bar{x}_1x_2x_3\bar{x}_5.$$

Eliminate from BDD all edges connected with 0 terminal node and obtain the BDD representing a combinational circuit behavior. Call this BDD as Circuit BDD. Cover Circuit BDD with CLBs to get a combinational circuit executing that we use the following rules [8].

1. CLB output corresponds to either non terminal node or the root of the Circuit BDD.

2. CLB input corresponds to either output of another CLB or variable of the Boolean function.

3. If two or more edges drop in a non terminal node of the Circuit BDD then this node may be split and covered with different CLBs.

4. The Boolean function implementing by a CLB is represented with the part (subgraph) of a Circuit BDD that is covered by the CLB. This function is derived from subgraph as DSoP depending on internal and input variables of the combinational circuit.

As a result we have got the combinational circuit *C*.

Covering Circuit BDD in accordance with rules 1-4 we have to provide coincidence of DSoPs system represented by the Circuit BDD with the DSoPs system derived from the combinational circuit *C* with substituting the proper DSoP of CLB instead of each internal variable of the combinational circuit *C*.

Consider Circuit BDD obtained from the BDD in Fig. 1 with using 3 inputs CLBs. Subgraphs of covering CLBs are represented in Fig. 2. The circuit obtained is shown in Fig. 3

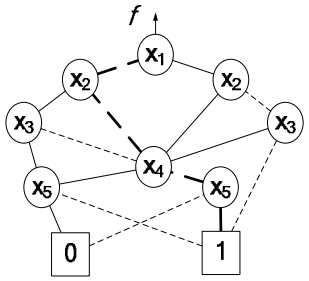


Figure 1. BDD for *f*

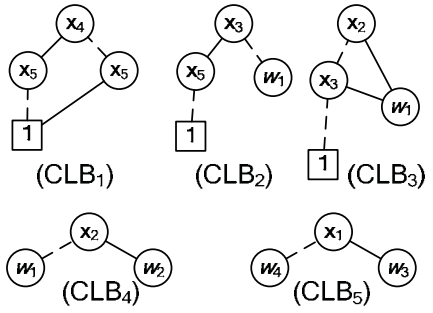


Figure 2. Subgraphs of covering CLBs

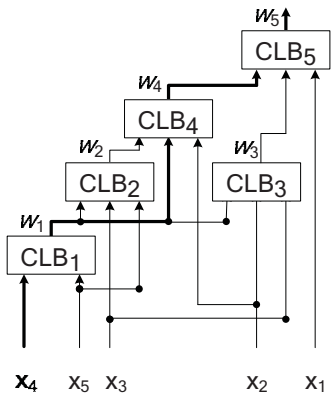


Figure 3. Circuit *C*

Thus for each CLB we have subgraph of the Circuit BDD and the corresponding DSoP. CLB output may be either output of the circuit or internal node of the circuit that is input of another CLB.

For example DSoP $x_2 w_1 \vee \bar{x}_2 x_3 w_1 \vee \bar{x}_2 \bar{x}_3$ corresponds to CLB_3 derived from relevant subgraph of Fig. 2. Here x_2, x_3 – variables corresponding to the circuit inputs, and w_1 is internal circuit variable relating to the output of CLB_1.

Non terminal node of Circuit BDD corresponds to CLB output if the node is a root of the CLB subgraph.

III. DERIVING TEST PAIR FOR PDF

A. Premises of test pair

We will examine one output circuit and corresponding BDD. Consider path α in the circuit of Fig. 3 (thick line). It begins from input variable x_4 and traverses the CLBs 1, 4, 5. CLB_5 output is output of the circuit.

We suppose that delays of the different paths from input to output of the same CLB are equal as CLB is LUT based. It means if a path α traverses certain CLB we may include any path of the CLB subgraph (connecting the subgraph root with its proper leaf) into the path α when investigating delay of the path α.

Derive reduced disjoint sum of products (reduced DSoP) for each CLB traversed by the path α. Reduced DSoP is formed from the CLB DSoP with including either products that contain the input variable corresponding to the beginning of the path α or products that contain the internal variable corresponding to the output of the previous CLB traversing with the path α.

For the chosen path α in Fig. 3 we have the following reduced and non reduced DSoPs (Fig. 2).

Reduced DSoP of CLB_1: $x_4 \bar{x}_5 \vee \bar{x}_4 x_5$, non reduced DSoP of CLB_1 is the same.

Reduced DSoP of CLB_4: $\bar{x}_2 w_1$, non reduced DSoP of CLB_4: $x_2 w_2 \vee \bar{x}_2 w_1$.

Reduced DSoP of CLB_5: $\bar{x}_1 w_4$, non reduced DSoP of CLB_5: $x_1 w_3 \vee \bar{x}_1 w_4$.

Move along the path α from the output of the circuit to the beginning of the path α. Substitute reduced DSoPs of the corresponding CLBs instead of internal nodes traversed and remove brackets. If obtained sum of products has internal variables corresponding to the outputs of CLBs that are not traversed with the path α, then substitute instead of these variables the corresponding non reduced DSoPs and remove brackets till a set of internal variables becomes empty. Denote derived set of products as K_α.

Notice that we will consider any set of products here and further also as sum of these products (SoP).

For the path α in our example we have the result of the first substitution: $\bar{x}_1 \bar{x}_2 w_1$. Then after the second substitution we have got K_α: $\bar{x}_1 \bar{x}_2 x_4 \bar{x}_5 \vee \bar{x}_1 \bar{x}_2 \bar{x}_4 x_5$. Take into consideration that if we would derive Equivalent Normal Form (ENF) from the circuit considered [11] then each literal of the sum of products would be supplied with sequence of numbers of elements corresponding to the proper circuit path.

Let products that contain literal ENF marked with the path α be called connected with the path α.

If we exclude in ENF sequences representing paths from all literals then K_α obtained above is a set of products connected with the path α.

Theorem 1. A set K_α contains all products connected with the path α. K_α is DSoP.

Proof. Each product from K_α contains either literal x_i or literal $\bar{x}_i$ that corresponds to the beginning of the path α and path α itself as each product is obtained with substitutions along the path α. By the construction K_α contains all products corresponding to the path α. K_α is DSoP as changing in any DSoP some internal variable for corresponding DSoP originates also DSoP [12]. The theorem is proved.

Take into account that each BDD path connecting its root with 1 (0) terminal node originates the product. If the path traverses the internal node marked with *i* and solid (dashed) edge then the originated product contain x_i ($\bar{x}_i$).

Any path connecting two internal BDD nodes originates the product in the similar way.

For example the path of Fig. 1 (thick lines) originates product $\bar{x}_1 \bar{x}_2 \bar{x}_4 x_5$ and part of this path connecting nodes 2 and 5 originates product $\bar{x}_2 \bar{x}_4$.

Theorem 2. For each product from K_α there exists the path from the root of BDD to its 1 terminal node that originates this product.

Proof. From covering Circuit BDD with CLBs follows that we have got all products of DSoPs as a result of all substitutions of the proper CLB DSoPs instead of circuit internal variables. It means that we have got all products of DSoP for each function of the system. For the path α we execute part of substitutions, excluding internal variables to obtain all products connected with the path α. Consequently each product connected with the path α is among the BDD products corresponding to the one output circuit considered. The theorem is proved.

One test pattern from a test pair detecting robust PDF of both rising and falling transitions [13] has to turn into 1 product K from K_α possibly together with other products from K_α. In our case the circuit behavior is represented with DSoP. Consequently this test pattern turns into 1 only product K. This product contains either x_i, or $\bar{x}_i$.

Let $\underline{K}$ be obtained from K with changing literal $x_i(\bar{x}_i)$ for the inversion literal. Call $\underline{K}$ as an addition of K.

Another test pattern of a test pair has to turn into 1 $\underline{K}$ and into 0 DSoP derived from BDD [13] of the circuit considered. Denote this DSoP as D_d. Let u be minimal cube covering v_1 ,v_2 and k_u product representing u.

To detect robust PDF it is necessary to provide the condition: product k_u is orthogonal to products of D_d excluding product K.

Take into consideration that input variable x_i corresponding to the beginning of the path α is correlated (in general case) to several nodes of BDD covered by the CLB traversed with the path α. These nodes are marked with the same variable x_i. To find test pair we may choose any of them [8]. Denote the chosen node as ***v***.

For example if we would consider the path traversing the same CLBs that the path α, but beginning from the input variable x_5 we have got two nodes of BDD covered by CLB_1 and marked with the variable x_5.

Let ε be path from the BDD root into chosen node ***v***. It originates the product k_ε. Derive from K_α the products corresponding to the path ε that is products containing k_ε. Exclude from them variables of k_ε. Denote the result as K^*.

In the example considered a set of products corresponding to the path ε is as follows: $k_\varepsilon = \bar{x}_1\bar{x}_2$, $K^* = x_4\bar{x}_5 \vee \bar{x}_4 x_5$.

Divide a set K^* into two subsets. Products of one subset have the literal x_i, and another – the literal $\bar{x}_i$. Exclude from them these literals and denote obtained subsets $K^*_{x_i}$, $K^*_{\bar{x}_i}$ correspondingly. These subsets represent functions implemented in nodes that are incidental to right (solid) and left (dashed) edges of the node ***v*** (Fig. 4). These functions are different by the construction of BDD (in this paper we always means ROBDD)

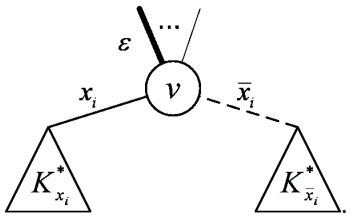


Figure 4. Fragment of BDD

In the example we have: $K^*_{x_i} = \bar{x}_5$, and $K^*_{\bar{x}_i} = x_5$.

Subsets $K^*_{x_i}$, $K^*_{\bar{x}_i}$ are determined with the pole **v** and do not depend on the chosen path ε (Fig. 4).

As the subsets represent different functions then there exists Boolean vector γ on which they take different values. Let γ turns into 1 $K^*_{x_i}$ and into 0 $K^*_{\bar{x}_i}$, k^* is a product representing γ.

In the example considered subsets $K^*_{x_i}$, $K^*_{\bar{x}_i}$ depend on the only variable x_5. Then we have: $\gamma = \overset{x_5}{0}$, $k^* = \bar{x}_5$.

First add literal x_i to k^* and then the product k_ε. Denote the result K', $K' = k^* x_i k_\varepsilon$ is absorbed by the only product K from K_α. It follows from the theorem 1 and the construction of Boolean vector γ. Let $\underline{K'}$ be addition of K' relative to x_i, then $\underline{K'}$ is absorbed by $\underline{K}$. It follows from the construction of Boolean vector γ and $\underline{K'}$. In the example: $K' = \bar{x}_1\bar{x}_2 x_4\bar{x}_5$, $\underline{K'} = \bar{x}_1\bar{x}_2\bar{x}_4\bar{x}_5$. Here K, $\underline{K}$ coincide with K', $\underline{K'}$.

Two products are orthogonal if their cubes don't intersect.

A product k is orthogonal to the sum of products (SoP) if k is orthogonal to each product of the SoP.

Theorem 3. $\underline{K'}$ is orthogonal to D_d.

Proof. We have to show that $\underline{K'}$ is orthogonal to products of K^* that are obtained from K_α with using path ε in above mentioned way. Remind that $\underline{K'}$ contains subproduct k_ε. Product $\underline{K'}$ is orthogonal to set of products $K^*_{\bar{x}_i}$ under construction. Product $\underline{K'}$ is also orthogonal to

set of products $x_i K^*_{x_i}$ hence $\underline{K'}$ is orthogonal to K^*. The theorem is proved.

Let Boolean vectors $\gamma^*, \underline{\gamma}^*$ turns into 1 products $K', \underline{K'}$ correspondingly and these vectors don't differ with variables that are absent in the products $K', \underline{K'}$. Let u be minimal cube covering these vectors and k_u – product representing this cube.

Theorem 4. Boolean vectors $\gamma^*, \underline{\gamma}^*$ comprise test pairs detecting robust PDF of the path α for both rising and falling transitions.

Proof. Vector γ^* turns into 1 product K (K from K_α) as K absorbs K' and vector $\underline{\gamma}^*$ turns into 1 product $\underline{K}$ as $\underline{K}$ absorbs $\underline{K'}$. Except $\underline{\gamma}^*$ turns into 0 D_d. Product k_u is orthogonal to products of D_d that does not contain k_ε as subproduct. Except (by the construction) k_u is orthogonal to a set of products $K^*_{\bar{x}_i}$ and a set of products $K^*_{x_i}$ without the product K. It means [13] that on vectors $\gamma^*, \underline{\gamma}^*$ PDF of α manifests itself as robust for both falling and rising transitions. The theorem is proved.

B. Algorithm of deriving test pair

Remind that each path connecting two nodes of BDD is related to the product originated by the path. Two paths are compatible if their originated products are not orthogonal.

The algorithm is partly (steps 1-5) based on results represented in [8] and connected with finding test pattern for single stuck-at fault of the circuit obtained by covering BDD with CLBs. We mean single stuck-at fault of the CLB pole directly connected with a circuit input.

1. Consider a path that begins from the node in which edge corresponding to x_i (solid edge) goes from the node **v** and ends into 1 terminal node of the BDD. Denote the path as η.

2. Look through paths that begin in the node in which edge corresponding to $\bar{x}_i$ (dashed edge) goes from the node **v** and ends into 0 terminal node of BDD in order to find path compatible with η. If we find such path then go to step 5. Otherwise return to step 1. If all paths η have looked through go to step 3.

3. Consider a path that begins from the node in which edge corresponding to $\bar{x}_i$ (dashed edge) goes from **v** and ends into 1 terminal node of BDD. Denote the path as η.

4. Look through paths that begin in the node in which edge corresponding to x_i (solid edge) goes from **v** and ends into 0 terminal node of BDD in order to find path compatible with η. If we find such path then go to step 5. Otherwise return to step 3.

5. Obtained path denote as ζ. Conjunction of products originated by the paths η, ζ, represents γ.

6. Derive Boolean vectors $\gamma^*, \underline{\gamma}^*$ in above mentioned way. These vectors have to turn into 1 the proper products $K', \underline{K'}$. Except, the Boolean vectors $\gamma^*, \underline{\gamma}^*$ do not differ with variables that are absent in products $K', \underline{K'}$.

In the example: $\gamma^* = \begin{matrix} x_1x_2x_3x_4x_5 \\ 0\ 0\ 0\ 1\ 0 \end{matrix}$, $\underline{\gamma}^* = \begin{matrix} x_1x_2x_3x_4x_5 \\ 0\ 0\ 0\ 0\ 0 \end{matrix}$, $K' = \bar{x}_1\bar{x}_2x_4\bar{x}_5$, $\underline{K'} = \bar{x}_1\bar{x}_2\bar{x}_4\bar{x}_5$. In the products $K', \underline{K'}$ variable x_3 is absent. In the Boolean vectors $\gamma^*, \underline{\gamma}^*$ this variable takes the 0 value.

In Fig. 5 thick lines represent paths corresponding to $\gamma^*, \underline{\gamma}^*$.

To detect robust PDF of α for rising and falling transitions we need triplets v_1, v_2, v_1 or v_2, v_1, v_2. Integrating triplets of all circuit paths we derive test T for all path delay faults.

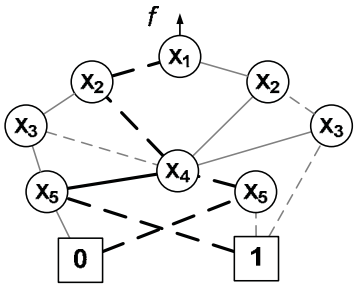


Figure 5. Representing $\gamma^*, \underline{\gamma}^*$

Theorem 5. Test T detects any PDF of a circuit (single and multiple).

Proof. As test T consists of pairs on which PDF manifests itself as robust and includes test pair for each path of a circuit then each single PDF is detectable with test T. As each single PDF manifests itself as robust on the proper pair of test T consequently any multiple PDF is detectable at the expense of single PDF comprising multiple fault. The theorem is proved.

Notice that all results derived for BDD easily may be spread to Shared BDD.

IV. EXPERIMENTAL RESULTS

For the experiments we used the benchmarks LGSynth'91 [14].

In Table 1, the names of the benchmarks are given in the first column, the numbers of inputs and outputs are given in columns two and three, respectively.

In section MUX-map, the results are given for a direct mapping of BDDs by multiplexors as described in [6]. The number of nodes in BDD (NoN), the number of paths (NoP), and the PDF coverage (PDFC) are given in corresponding columns. This technique doesn't provide the 100% PDF covering. To provide 100% PDF covering for circuits in the frame of this technique it is necessary an additional input [7]. Using an additional input increases the number of nodes in BDD (and consequently the number of multiplexors) and the number of paths [7] for the same benchmark.

In section LUT-map, the results are given for the technique described above for 3 inputs CLBs. The number of CLBs in the circuit (NoC), the number of paths (NoP), and the PDF coverage (PDFC) are given in corresponding columns.

TABLE I. EXPERIMENTAL RESULTS

Name	in	out	MUX-map			LUT-map		
			NoN	*NoP*	*PDFC*	*NoC*	*NoP*	*PDFC*
5xpl	7	10	90	273	89.0	69	175	100.0
C17	5	2	12	22	68.1	6	12	100.0
alu2	10	6	259	873	86.9	246	929	100.0
b9	41	21	237	1773	64.6	141	380	100.0
clip	9	5	256	954	79.4	235	597	100.0
conl	7	2	20	47	74.4	9	18	100.0
count	35	16	251	2248	66.1	199	642	100.0
il	25	13	60	137	74.4	41	85	100.0
15	133	66	313	44198	61.3	169	941	100.0
t481	16	1	34	4518	86.1	26	1226	100.0
tcon	17	16	34	40	100.0	16	32	100.0
9sym	9	1	35	328	72.5	27	195	100.0
f51m	8	8	72	326	99.3	58	139	100.0
z4ml	7	4	66	175	77.1	50	118	100.0
x2	10	7	75	188	72.3	61	251	100.0

Experimental results showed that the number of CLBs and the number of paths are as a rule less then the number of multiplexers and the number of paths for the same benchmark.

V. CONCLUSION

Special combinational circuits are investigated. They are derived by covering BDDs with LUT based CLBs. It is found out that PDF of each path of such circuits manifests itself as robust. Except for delays of rising and falling transitions of the same path there exist triplets v_1, v_2, v_1 or v_2, v_1, v_2 detecting these delays. Triplets are found with BDD analyses based on finding test pattern for single stuck-at fault of CLB input directly connected with a circuit input. Experimental results showed that lengths of tests for PDFs of investigated circuits and the sizes of these circuits as a rule less than ones in circuits implemented with multiplexors. Investigated circuits do not demand additional input to provide 100% testability for robust PDFs.

REFERENCES

[1] C.J. Lin and S.M. Reddy, "On Delay Fault Testing in Logic Circuits", IEEE Trans. on Computer-Aided Design, vol. 6, no. 5, pp. 694– 701, September 1987.

[2] M. L. Bushnell and V. D. Agrawal, "Essentials of ElectronicTesting for Digital, Memory, And Mixed-Signal", VLSI Circuits. Hingham, MA, USA: Kluwer Academic Publishers, 2000, p. 432

[3] P. Ashar, S. Devadas, and K. Keutzer, "Gate-delay-fault testability properties of multiplexor-based networks," in Proc. Int. Test Conf., 1991, pp. 887-896.

[4] P. Ashar, S. Devadas, and K. Keutzer, "Testability properties of multilevel logicnetworks derived from binary decision diagrams," in Proc. Adv. Res. VLSI. Univ. California, Santa Cruz, 1991, pp. 33-54.

[5] P. Ashar, S. Devadas, and K. Keutzer, "Path-delay-fault testability properties of multiplexor-based networks," Integration, VLSI J., vol. 15, no. 1,pp. 1-23, 1993.

[6] B. Becker, "Testing with decision diagrams," Integration, VLSI J., vol. 26, pp. 5-20, 1998.

[7] R. Drechsler, J. Shi, and G. Fey, "Synthesis of Fully Testable Circuits From BDDs", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 23, no. 3, march 2004, pp. 1-4.

[8] A. Matrosova, E. Lukovnikova, S. Ostanin, A. Zinchyk, E. Nikolaeva, "Test Generation for Single and Multiple Stuck-at Faults of a Combinational Circuit Designed by Covering Shared ROBDD with CLBs", Proc. of the 22nd IEEE Intern. Symp., Rome, Italy, 2007, pp. 206-214.

[9] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," IEEE Trans. Comput. , vol. C-35, pp. 677-691, Aug. 1986.

[10] S. Minato, N. Ishiura and S. Yajima, "Shared binary decision diagram with attributed edges for efficient Boolean function manipulation", Proc. 27th IEEE/ACM DAC, June 1990, pp. 52-57.

[11] D.B. Armstrong, "On Finding a Nearly Minimal Set on Fault Detection Tests for Combinational Logic Nets", IEEE Trans. Electronic Computers, EC-15, 1966, pp. 66-73.

[12] A. Matrosova, "Random simulation of logical circuits", Avtomatika i Telemechanika, N1, 1995, pp. 156-164.

[13] A. Matrosova, V. Lipsky, A. Melnikov, V. Singh, "Path Delay Faults and ENF," Proceeding of EW&DT Symposium, Russia, St. Petersburg, September, 2010, pp. 164-167.

[14] S. Yang, "Logic Synthesis and Optimization Benchmarks User Guide", Tech. Rep., Microelectron. Center of North Carolina, 1991.

Extending BS-1149.1 for Interconnect Online BIST

Somayeh Sadeghi-Kohan, Ghazaleh Vazhbakht, Parisa Shaafi Kabiri, Zainalabedin Navabi
Electrical and Computer Engineering Department
School of Engineering Colleges, Campus 2
University of Tehran, Iran
{somayeh, ghazal, pani, navabi}@cad.ece.ut.ac.ir

Abstract— **Loss of signal integrity in today's deep sub-micron designs puts communication links at a higher risk of permanent or more frequent intermittent faults and results in reducing the performance and reliability. This paper presents an online interconnect BIST method that is implemented by a simple extension to the boundary scan standard. The proposed extensions to the boundary scan structure facilitate online testing methodology with negligible hardware overhead. The extensions are fully compatible with the IEEE std.1149.1 standard and do not interfere with the normal operation of boundary scan. It effectively reuses existing boundary scan structure to act as signature generator, error detector and error locator; therefore the proposed method includes fault detection and diagnosis phases.**

Keywords-communication; boundary scan; online test; hybrid serial/parallel communication; signature

I. INTRODUCTION

With the advances in VLSI technology and moving towards deep sub-micron technologies, nowadays multi-processor SoC platforms include hundreds of embedded processors with a wide range from heterogeneous components with irregular block sizes to homogeneous components with regular block sizes [1-2]. Therefore, communications are the dominant factor which affect the performance of an SoC [15] and its power consumption [1, 3, 15]. Reliability of SoCs increasingly depends on the error free operation of such interconnects. So testing of SoCs, implies not only the logic cores but also the interconnect architecture.

Shrinking of the feature size, increasing of the interconnect wires length and rapid increasing of the SoCs' functional frequency make signal integrity a major concern which must be considered. Using deep sub-micron SoCs magnifies the capacitive coupling between adjacent wires. This leads to crosstalk noise and affects the signal integrity, by providing glitch, delay, and hence deviation of the functionality and performance from the expected behavior. Decreasing or eliminating capacitive coupling is a challenge during fabrication. It is impossible to check and fix all crosstalk effects during circuit design by only design rule checking (DRC), validation, and analysis [4]. On the other hand, electromagnetic (EM) noise and voltage drop have to be considered during normal operation of a circuit. Also it is possible that the data transferring on other interconnects ruin the transferred data through links of a specific interconnect. In other words the environment around the circuit produces some noises which can ruin the transferred data through interconnect. These faults arise during normal operation of system. The authors in [5] prove that there are some faults that cannot be tested during offline testing and therefore online testing of interconnections is inevitable.

Most of the crosstalk effects are visible in GHz clock frequencies; consequently, at-speed testing is essential for adequately testing high performance chips. Also, there is always a gap between functional frequency of circuits and the frequency of ATEs, and this gap grows with technology scaling. Moreover, ATE with high speed, large pin count, large memory, and good timing resolution would be expensive. Therefore a mechanism for at-speed interconnects testing other than using ATE is necessary.

An accepted cost effective way of testing SoCs, is IEEE std.1149.1 boundary scan standard. This standard adds some hardware structures around cores on a chip to make them more controllable and observable [6]. One of the drawbacks of using this method is the inability of detecting timing related faults, since the method is only applicable in offline mode where the test frequency is much lower than functional frequency of the circuit. In conclusion, the idea of employing online test infrastructures for at speed testing of interconnects seems unavoidable.

This paper addresses the issue of interconnect testing in online mode. Our proposed technique, called online IBIST (Interconnect Built-In Self-Test), reuses the existing on chip test structure. Our method splits the communication link into separated parts and uses the transferred data during normal operation of the circuit to generate signatures in each part for each block transfer. This operation is performed on both sides of a communication link. By modifying the boundary scan structure, the signatures are easily exchanged and compared.

The rest of the paper is organized as follows: Section II presents the background and some of the related works. Section III illustrates the basic idea of the proposed method. Section IV discusses the incorporation of the basic idea into IEEE std.1149.1 standard. Section V demonstrates experimental results in terms of defect coverage and hardware overhead. Section VI concludes the paper.

II. Related Works

Several works are reported in the literatures on introducing fault models for interconnects. Maximum aggressor (MA) [8] is one of the most referred fault models. This model only considers the crosstalk faults. Another proposed fault model is called multiple transitions (MT) [9]. In contrast to crosstalk faults, this model covers the effect of inductance between wires; hence, it is more accurate. The shortcoming of this model is the increase in number of test patterns.

In order to decrease test time, several researches are done with the aim of reducing the number of test patterns required for detecting the considered faults [10]. Authors in [11] proposed a test generation method for signal integrity faults on long interconnects.

Many researches [12, 13] used a combination of linear feedback shift register (LFSR) and multiple input signature register (MISR) for interconnect testing. The fault coverage of their techniques is usually low. Also some optimization approaches for interconnect BIST are introduced [14, 15]. The authors in [14] used a technique based on graph coloring and genetic search and suggested a new LFSR/MISR structure for interconnect testing, which has a high fault coverage. The Logic/Interconnect BIST method presented in [15] is used for offline testing of both logic cores and their interconnections. There are embedded LFSRs and MISRs in logic cores to ease the procedure of their testing. The embedded LFSRs are reconfigurable and just by adding some AND/OR gates to their outputs they can generate test patterns for testing the interconnections. Additionally, the embedded MISRs compact the responses during testing. The method has a low hardware overhead and efficiently uses existing infrastructures of the system.

There are several works [16, 17-19] which incorporate deterministic test patterns into interconnect BISTs. In [17], a modified walking and counting sequence is proposed that replaces each 0 with 100 and each 1 with 011. Consequently, it detects more faults. The authors did not discuss the structure of modified BIST.

A novel technique for interconnect testing is introduced in [20] which does not require test vectors. Instead, impulses are sent through the links and their reflections are listened. If the delay between the impulse and its reflection is more than a determined threshold, the link is marked as a faulty link. This method can only detect stuck-at faults.

Another method for interconnect testing is a test-per-clock fault detection, diagnosis and identification [22]. In this method IETD-MISR (Internal Exclusive-OR linear feedback MISR composed of T-type and D-type flip-flops) is used for signature compaction. It has been proved that the MISR implemented using D and T flip-flops, has better detection properties than a classic MISR. A test structure using the presented MISR is introduced that detects, locates and identifies interconnect faults. Fault type identification is performed by signature normalizing and therefore, decreases signature aliasing.

Several researchers used BS-1149.1 standard for interconnect testing. They suggested several modifications to the structure of boundary scan cells that make them capable of generating test patterns for crosstalk and bridging faults [4, 9, 21]. These methods add a number of gates, multiplexers and flip-flops to the standard boundary scan cells, and the user defined instructions makes the boundary scan structure able to test interconnects in offline mode.

An online test technique is proposed in [5] which adds two flip-flops on the receiver side and samples the receiving data two different times. An XOR gate is then used to compare the two samples. This method cannot detect stuck-at faults, and also misses the long delay lengths.

III. Basic Idea

For online interconnect testing, we use transferred data at the end of a test session (e.g., completion of a block transfer) to compute a signature on each side of a parallel link. Afterward, source and destination of the link send a copy of locally computed signature to the other side. The interfaces on both sides compare the received signature with their own computed ones. A mismatch between these signatures indicates that there is a fault on the interconnect. An overall view of the proposed method is depicted in Fig. 1.

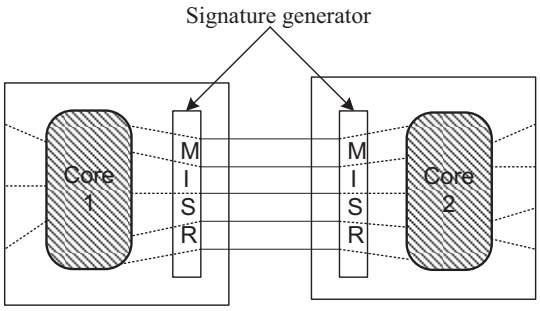


Figure 1. Overall view of online testing of parallel links

The receiver uses a MISR to generate a signature from the incoming data; therefore signature generation does not distract the normal parallel data transferring. To reduce the hardware overhead, the idea can be embedded into the boundary scan infrastructures. This results in an interconnect BIST called IBIST.

Signature exchanging is done by means of serial links. For every parallel communication link, two serial lines in both directions carry signatures generated from parallel data. The inequality between locally computed signature and the received one reveals the existence of a fault on the parallel link. As we will show later, the location of fault (faulty line) can easily be determined. Exchanging of signatures and comparing them with the expected ones overlaps with the next parallel data transferring. Indeed, this method can effectively perform fault detection and fault location by using some area and time redundancies.

There is a compromise between the complexity of data compaction circuit (signature generation) and fault coverage of this method. The simplest method for signature generation is using the last data passing through the parallel link as the signature. So, this data is sent to the receiver side via both parallel and serial links. Fig. 2 shows such an implementation. Eight consecutive data (a block transfer) is sent through the

parallel communication link. The last data of each block is redundantly sent via a test-enhanced serial link. This test session can disclose the existence of the represented faults on the figure. The additional hardware used for this implementation is just the components used for the serial link. Although this implementation has low hardware overhead, it needs too many test sessions to detect all faults. In fact, the mean time to detect a fault is relatively high and the fault coverage is not good.

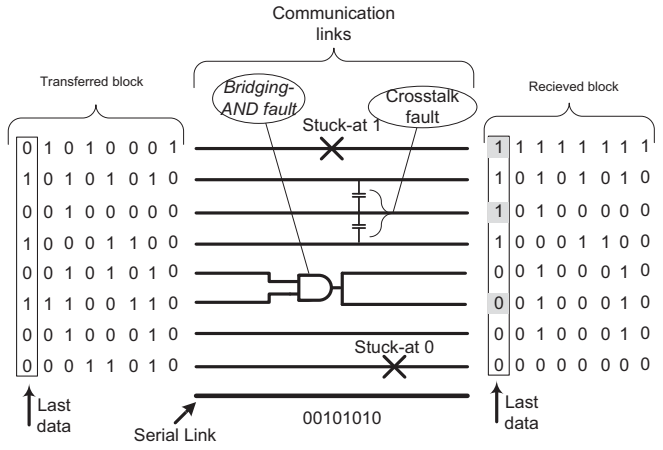


Figure 2. Using last parallel data as the signature

The second approach to generate the signature is to concatenate the parities of all lines in a parallel link together. Thus, each bit of the signature shows the parity of a line for a block transfer in the parallel link. Therefore the locations of faults are easily specified. Fig. 3 shows the interface of the sender and receiver for this approach. Obviously, this implementation imposes more hardware than the previous one. Sender needs a register to hold the signature, and at the receiver side in addition to the previous method's components, a set of XOR gates is needed to generate the signature from receiving data. Fig. 4 shows how this approach can detect faults on the interconnects. This approach reveals the existence of more faults than the previous one. Through the rest of this paper we use this method for signature generation.

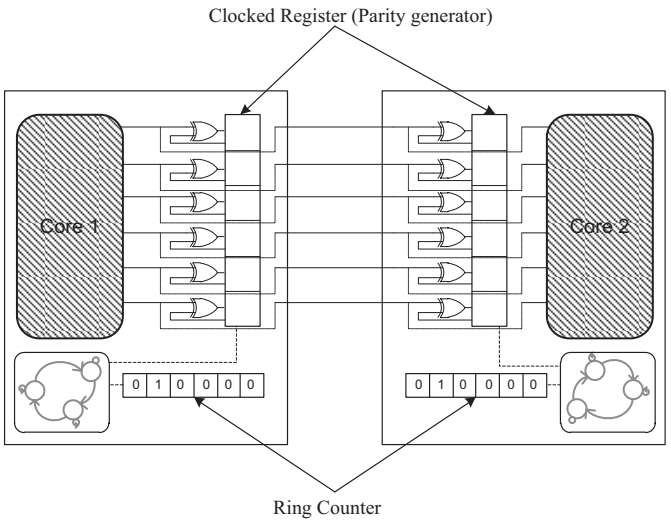


Figure 3. The general structure of sender and receiver when parity is used as the signature generation method

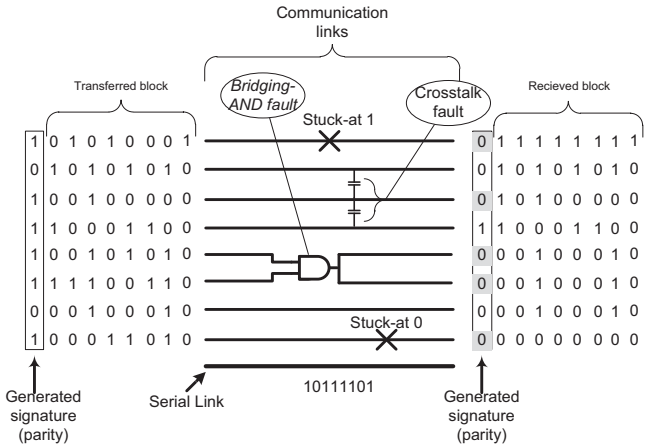


Figure 4. Using the concatenation of parities for each line of a parallel link as the signature

IV. Incorporating Proposed Idea into IEEE std.1149.1 Standard

In this section, the structure of the method is demonstrated. Then, enhancing the IEEE std.1149.1 standard for online testing is discussed.

A. Adding Serial Links

In the proposed method, outputs of the boundary scan chain are divided into several groups with regard to test time and hardware overhead. Dividing the boundary scan chain leads in test time reduction but increases hardware overhead. Our method uses serial output of capture flip-flops in boundary scan cells and also adds two extra serial links for each group between source and destination. The serial input of each group is the serial output of the previous group, or the serial data from the other side of the communication. Two added serial links are used for transferring the generated signatures. Data on the serial link is loaded into the capture flip-flops chain at each clock cycle.

Fig. 5 shows the general structure of the extended boundary scan cells. Several 2-to-1 multiplexers are added to the boundary scan registers to select between the serial output of the previous group or the serial data from the other core. These multiplexers mark the boundaries of different port groupings. This partitioning is a trade-off between increase in hardware overhead and reduction in the mean time to detect. For each multiplexer, a 1-to-2 de-multiplexer is needed to incorporate into the boundary scan structure of the other core for sending the shifted data to the serial input of the next group, or to the other side of the communication. As shown in Fig. 5, for each group, two serial links are added beside the parallel links which are connected to the boundary scan chain.

B. Incorporating Different Phases of Test Procedure into Boundary Scan Architecture

To perform online testing, a test controller is needed to issue appropriate control signals and manage the test methodology phases. The proposed test method has three

phases of signature preparation, signature exchanging and signature verification. Here we will show how to incorporate the controlling operations for each phase into the TAP controller of the boundary scan.

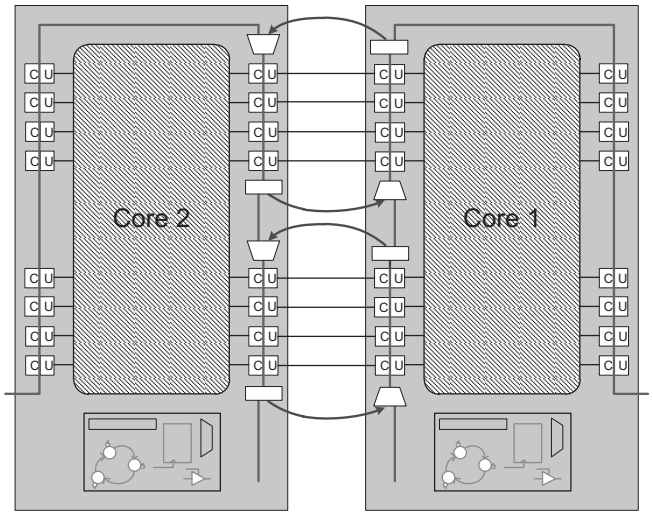


Figure 5. The general structure of extended boundary scan

1) Signature Preparation: Signature preparation is the first phase of the controller. A signature is generated from online data transferred through an interconnect by using capture flip-flops of boundary scan registers or other flip-flops located in the boundary scan cells that imitate the BIST structure. The following operations are performed in this phase:

- Updating signature in each data transmission session
- Capturing signature into capture flip-flops of boundary scan register
- Loading signature into update flip-flops of boundary scan register

The controller recognizes starting point of each data transmission session based on the communication protocol at both sides and issues appropriate signals to update the collected signature. Signature generation can be performed by a specified MISR outside of the boundary scan register, or by a modified boundary scan cell. Two different modifications to the standard boundary scan cells are proposed here to allow boundary scan cells to perform signature generation process, these methods are as follow:

Type I: The first modified boundary scan cell is depicted in Fig. 6. As shown in this figure, an XOR gate is added to the boundary scan cells, which is used for computing the parity of transferred data on each link. Using this structure, the parity of the actual data for each block transfer is collected in capture flip-flops. The controller issues the *ClockBR* signal at each communication transfer, and the updated parity is loaded into the capture flip-flops. Since the signature is already stored in capture flip-flops, there is no need for capturing it again at the end of the test session. The signature exchanging happens in shifting phase by capture flip-flops during the next block transfer. Thus, the signature for the next block cannot be constructed due to the fact that the capture flip-flops are busy with shifting the previous block's signature. Therefore the noise or fault on this block transfer remains undetected and the mean time to detect faults increases.

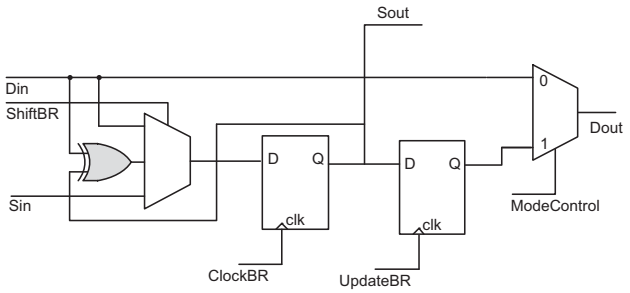


Figure 6. The first modification to standard boundary scan cell to make the cell a signature generator

Type II: The second type of the modified boundary scan cell is depicted in Fig. 7. As shown, a flip-flop is added to the structure of the boundary scan cells which utilizes an XOR gate to construct the signature. The generated signature is captured into the capture flip-flops at the end of a block transfer or a test session, and then it is transferred to the other side of the communication link. Note that the signature exchanging phase of the previous test session overlaps with the signature preparation phase of the current session.

When the block transfer or test session is finished, controller issues capturing signals and the generated signature is loaded into the capture flip-flops. Then the controller issues updating signal to save the signature in update flip-flops.

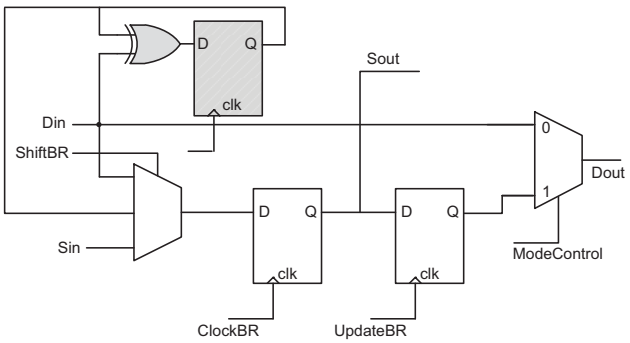


Figure 7. The second modification to standard boundary scan cell to make the cell a signature generator

2) Signature Exchanging: Since the generated signature will be needed later for comparison, before the exchanging starts, capture flip-flops, which contain signature, are loaded into the update flip-flops. The controller then initiates exchanging phase using shift mode and the shifting operation starts when *shiftDR* is issued. The signature in capture flip-flops is shifted from the source to the destination and vice versa. The signatures are transferred through the embedded serial links.

3) Signature Verification: The exchanged signatures should be verified on the the other side of the communication. The verification phase is done by comparing the signature in the capture flip-flops with the one stored in update flip-flops. A

mismatch would indicate a fault on the interconnect. As this comparison is done at the source and destination simultaneously, they both detect the fault at the same time. Fig. 8 shows the modified boundary scan cell that is able to perform the comparison operation. As shown, an XOR gate is added to the output of the capture and update flip-flops.

Both signature generation and signature comparison operations can combine into a modified boundary scan cell. Fig. 9 shows a signature generator/comparator cell constructed from the modified boundary scan cell Type 1.

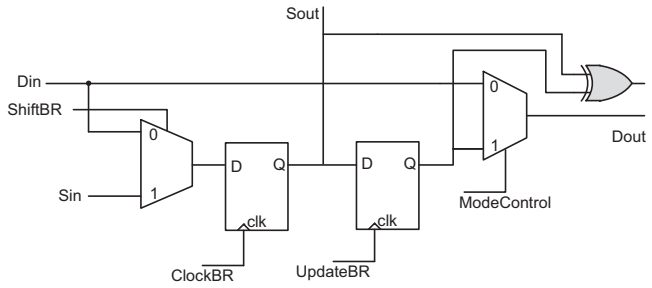


Figure 8. The modification to standard boundary scan cell to make the cell a signature comparator

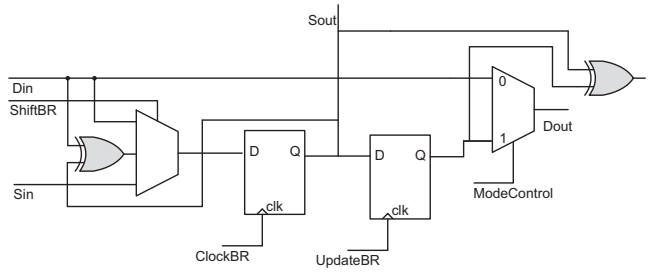


Figure 9. The proposed signature generator/comparator boundary scan cell

C. *Implementation of Controller*

Several implementation methods can be suggested for the proposed test methodology. Here, we present three different implementation methods.

Implementation 1: This implementation uses mandatory boundary scan instructions. The flow of operations which are performed through the *sample* instruction is highly compatible with the proposed method. Fig. 10 shows the standard TAP controller of boundary scan. *Sample* is loaded in the instruction register in the *capture_IR* state of the TAP controller. Then it takes at least two clock cycles to go back to *Run_Test_Idl*e state. Controller stays in this state while the signature is being generated. Subsequently, in the *Capture_DR* state the generated signature is loaded into capture flip-flops. At least after two clock cycles, the generated signature is loaded into the update flip-flops in the *update_DR* state of the TAP controller. At this time, the signature generation phase is completed. The signature exchanging phase starts after at least three clock cycles in the *shift_DR* state of the TAP controller.

For directing the TAP controller toward appropriate states, a simple controller is employed which acts as a little ATE and issues sequencing data for TMS signal. Since, several clock cycles are needed to switch between different phases, the test session time increases. Therefore, some of the transferred blocks do not participate in the test procedure and the average time to detect faults increases. The general view of this implementation is shown in Fig. 11.

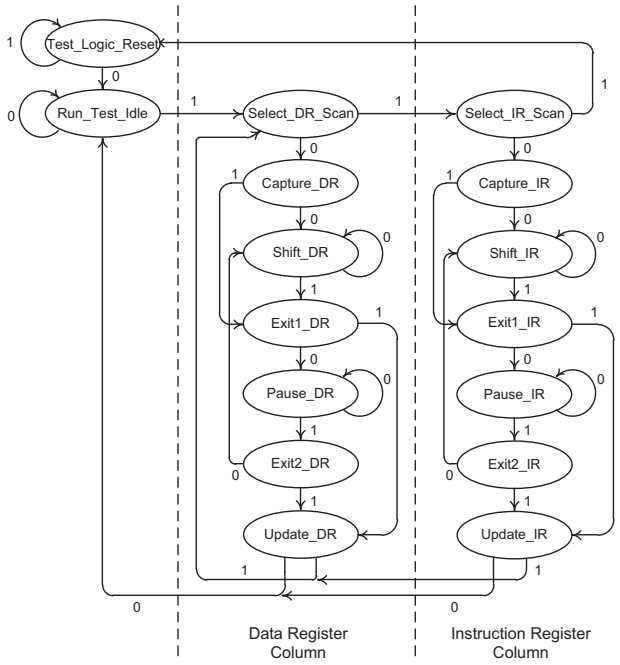


Figure 10. The TAP controller of the boundary scan standard

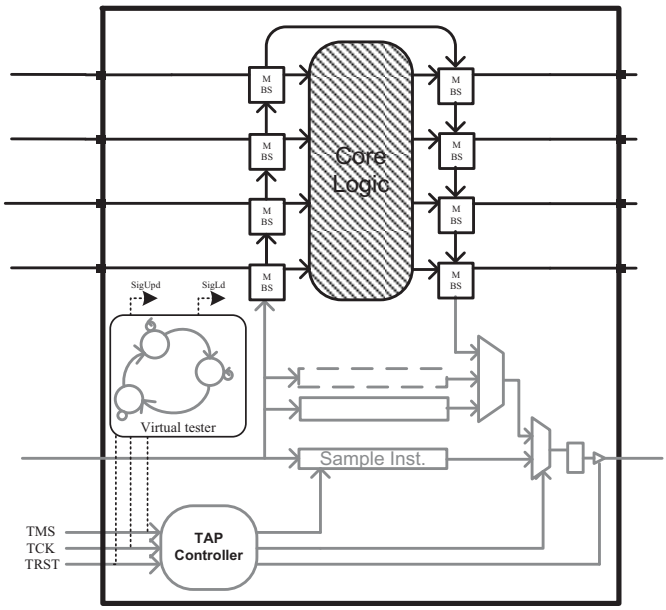


Figure 11. General view of incorporating proposed method into boundary scan standard with implementation 1

Implementation 2: To reduce the test session time that is the problem of the previous implementation, Implementation 2 uses user-defined instructions. The *RunIBIST* instruction is added as a user-defined instruction to the boundary scan structure in order to perform test procedure. To start the online

test, this instruction is loaded into the instruction register in a same way as all other instructions. Then the embedded controller is activated by the decoder in the *update_IR* state of the TAP controller, and the TAP controller goes into the *Run_Test_Idle* state. It is also possible to deactivate the TAP controller and the decoder to reduce power consumption in this step. Embedded controller is a simple controller that works with system clock and initiates the signature generation phase based on the communication protocol. For example, if *request* and *acknowledge* signals are used in a communication protocol, asserting the *request* signal activates the controller on the receiver side, and *acknowledge* does the same operation on the sender side. The advantage of this implementation is that handshaking signals, in addition to the data signals, are also being tested. For example, if noise occurs on the *request* signal, controller at the receiver side updates its signature, and the sender does not. So, the test session at the receiver side ends earlier, and its controller goes to signature exchanging phase. Since the signature generation phase on the sender side has not yet completed, both sides of communication detect the faulty handshaking. A similar situation works for the *acknowledge* signal. Same as the previous implementation, we need a simple controller to issues TMS signal to handle TAP controller and loads the new instruction into instruction register and controls desired states. The overall structure of this implementation is shown in Fig. 12.

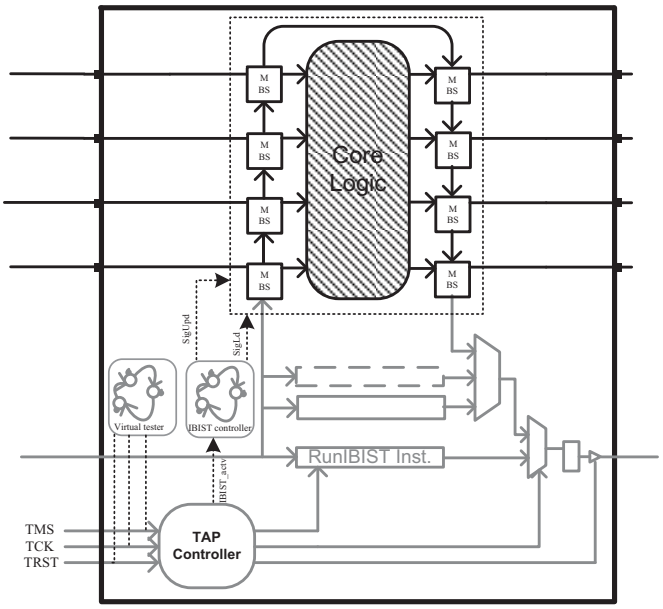


Figure 12. General view of incorporating proposed method into boundary scan standard with implementation 2

Implementation 3: In this Implementation, a separate controller is used for controlling the test procedure. The TAP controller is not activated, in this implementation. The special-purpose controller issues appropriate signals to complete three testing phases. The controller uses the system clock, and as with the controller in Implementation 2, the controlling operation is protocol dependent. Since this controller does not interfere with the TAP controller, there is no need for a controller to drive the TMS signal.

V. RESULTS AND COMPARIONS

This section represents, the comparison between the proposed method and three other methods [4, 5, 9]. Then the results for fault coverage and hardware overhead are presented.

In [5], a method called Dual Sampling is proposed for interconnect testing. This method tests delay faults on interconnect links. It adds two flip-flops, an XOR gate and a buffer in order to sample and compare the receiving data at two different time instances. Bridging faults and stuck-at faults are not detectable with this method.

Another method proposed in researches for interconnect testing, is one that changes boundary scan structure for testing crosstalk faults on communication links [9]. This method performs an offline interconnect testing by adding two multiplexers, a T- Flip-Flop, an AND gate to each boundary scan cell, and two instructions to the boundary scan structure. As with the dual sampling, this method cannot detect stuck-at faults and bridging faults.

In [4], a method for at-speed interconnect testing is introduced, that tests the crosstalk faults. This method uses a BIST-based enhanced boundary scan structure. Bridging faults and stuck-at faults are not detectable in this method.

In comparison with the previous methods, our proposed method performs an online interconnect testing. As discussed before, this method can easily specify the location of faults. In contrast with the other methods, stuck-at faults and bridging faults are also detectable using this method. Table I shows a comparison of the above methods.

TABLE I. COMPARISON OF PROPOSED METHOD WITH PREVIOUSLY REPORTED METHODS

Features / Methods	Sender/Receiver synchronization in detecting fault	Online Testing	Fault detection for handshaking signals	Fault detection for crosstalk faults	Fault detection for bridging faults	Fault detection for stuck-at faults
Our Method	✓	✓	✓	✓	✓	✓
Method in [5]	×	✓	✓	✓	×	×
Method in [9]	×	×	×	✓	×	×
Method in [4]	×	×	×	✓	×	×

A. Fault Coverage

Table II shows fault coverage of the proposed method for a 32-bit communication link. The communication link is divided into 4, 8 and 16 bit groups, and for each partitioning, the fault coverage is reported for the first block transfer of random data. Since the test data for the specific applications are not available, usually researches use random data for evaluating their online test methods; therefore random data is used for fault simulation in this paper.

As shown in Table II, when the communication link is divided into more groups, the length of each group is shorter; therefore, less data are incorporating in signature generation. Note that the number of transmitted data in a block transfer is

equal to the number of the links in each group due to the fact that signature exchanging phase should overlap with the next block transfer. Since the signature generation method is based on parity calculation, dividing in more groups results in a better fault coverage. Also it should be noted that after two or three block transfers, all types of faults are detected. Therefore the average time to detect a fault improves.

TABLE II. FAULT COVERAGE OF THE PROPOSED METHOD

Fault Type / Part Length	Stuck-at	Bridging (OR, AND)	Sate-dependent	Crosstalk
4-bit	81%	71%	77%	59%
8-bit	75%	65%	73%	55%
16-bit	72%	63%	66%	49%

As mentioned before, random data is used for fault simulation, and single and multiple faults are injected in the circuit. In the proposed method, the number of faults has no effect on fault coverage, so the same results as these of Table I are derived for multiple faults.

The crosstalk faults are activated through specific data patterns, so if these patterns do not transfer through faulty links, they remain undetected. The low fault coverage for this fault type is due to this reason.

Table III, presents fault coverage of the proposed method in comparison with the other three methods in [5], [4] and [9]. These results are for the first block transfer of an 8-bit communication link. As shown in this table, the proposed method can detect all types of faults in contrast with the other three methods.

TABLE III. COMPARING FAULT DETECTION CAPABILITIES OF DIFFERENT METHODS

Fault type / Methods	Stuck-at	Crosstalk	State-dependent	Bridging
Our Method	75%	55%	73%	65%
Method in [5]	0%	40%	0%	0%
Method in [9]	0%	100%	0%	0%
Method in [4]	0%	100%	0%	0%

The minimum number of clock cycles for detecting 90 % of faults is shown in Table IV. For all types of partitioning, after 5 block transfers nearly all faults are detected.

TABLE IV. MINIMUM REQUIRED TIME TO DETECT ALL FAULTS

	4-bit	8-bit	16-bit
Number of Clock Cycles	20	40	80

B. Hardware Overhead

Table V, shows the number of 2-input AND gates added into the boundary scan structure in different methods.

TABLE V. THE NUMBER OF GATES ADDED TO BOUNDARY SCAN CELL IN DIFFERENT METHODS

	Number of added 2-input AND gates	
	Receiver Side	**Sender Side**
Our Method	14	14
Method in [9]	28	18
Method in [4]	11	10
Method in [5]	18	-

As shown in this table, compared with the offline interconnect testing methods presented in [9] and [4], the proposed online method has an acceptable hardware overhead. Although the method of [5] has a smaller hardware overhead, it has a lower fault coverage, as shown in the Table III.

In Table VI, hardware overhead of the proposed method for different circuits is presented. As shown, there is a trade-off between the number of partitions that the communication link is divided into and the hardware required for test structure. The increase in number of partitions reduces the average time to detect faults. However, each partition adds two ports to the circuit and increases the number of multiplexers and de-multiplexers.

Hardware overhead of different methods for SAYEH processor [23] is reported in Table VII. As shown, the results are almost the same. Each of these methods, in addition to the hardware added into the boundary scan cells, adds also a test controller and user-defined instructions to this architecture. The controller of the proposed method is very simple, and, as discussed in Implementation 3, there is no need to define new instructions in this methodology. The controller of the method presented in [4] is the most complex one, and although this method performs offline interconnect testing, its hardware overhead is high. In the proposed method, the communication link is divided into 8-bit groups.

TABLE VI. HARDWARE OVERHEAD OF THE PROPOSED METHOD

Part Length / Circuit Under Test	4-bit	8-bit	16-bit
Adding Machine	11%	8.5%	8%
SAYEH Processor	3%	2%	~2%

TABLE VII. COMPARING THE HARDWARE OF DIFFERENT METHODS FOR SAYEH PROCESSOR

	Our Method	Method in [5]	Method in [9]	Method in [4]
SAYEH Processor	~2.1%	~1.4%	~2.2%	~2.1%

VI. CONCLUSION

We proposed an online interconnect BIST based on a hybrid serial/ parallel communication. The methodology is implemented with boundary scan structure; therefore the hardware overhead is negligible. The modified boundary scan structures are used to generate and compare compacted data on both sides of the communication. Because of this our method can locate faults as well as detecting them.

ACKNOWLEDGMENT

This project was partially funded by Iran Telecommunication Research Center (ITRC).

REFERENCES

[1] P. Magarshack and P. G. Paulin, "System-on-chip beyond the nanometer wall", In Proc. DAC, 2003, pp. 419-424.

[2] L. Benini and G. De Micheli, "Networks on chips: A new SoC paradigm", Computer, Vol. 35, No. 1, 2002, pp. 70-78.

[3] N. Karimi, S. Sadeghi, Z. Navabi, "Network-on-chip concurrent error recovery using functional switch faults", In Proc. of 11thWorkshop on RTL and high level testing (WRTLT'10), Shanghai, Dec. 2010.

[4] Nourmandi-pour, A. Khadem-zadeh, A.M. Rahmani, "An IEEE 1149.1-based BIST method for at-speed testing of inter-switch links in network on chip", Elsevier J. Microelectronic. Vol. 41, No. 7, 2010. pp. 417–429.

[5] Y. Zhao , S. Dey , L. Chen, "Double sampling data checking technique: an online testing solution for multisource noise-induced errors on on-chip interconnects and buses", In IEEE Trans on VLSI Systems, vol. 12, no. 7, July 2004, p.746-755.

[6] Z. Navabi, "Digital system test and testable design with HDLs", ISBN 978-1-4419-7547-8, Springer, 1st edition, January 2011.

[7] S. Sadeghi-Kohan, F. Javaheri, S. Mahmoudi and Z. Navabi, "Multi level test package: a package for C/C++ gate level fault simulation of system level design", In Proc. of 11th Workshop on RTL and high level testing (WRTLT'10) and Journal of Shanghai Normal University, Shanghai, Dec. 2010.

[8] M. Cuviello, S. Dey, X. Bai and Y. Zhao, "Fault modeling and simulation for crosstalk in system-on-chip interconnects", In Proc. of the International Conference on Computer Aided Design, 1999, pp. 297-303.

[9] M. H. Tehranipour, N. Ahmed and M. Nourani, "Multiple transition model and enhanced boundary scan architecture to test interconnects for signal integrity", In Proc. Of IEEE Int. Conf. Comput. Des.: VLSI Comput. Process. (2003) 554-559.

[10] Z. Ying, Huawei1 and L. Xiaowei1, "MT compacted set for interconnect crosstalk on SOC", In Proc. of IEEE eight Workshop on RTL and high level testing, 2007, pp. 125-129.

[11] A. Attarha and M. Nourani, "Test pattern generation for signal integrity faults on long interconnects", In Proc. of the 20th IEEE VLSI Test Symposium, 2002, pp. 336-341.

[12] J. Koeter, S. Sparks, "Interconnect testing using BIST embedded in IEEE 1149.1 designs", In Proc. of Int. ASIC Conf. , September, 1991, pp. P11-2.1- P11-2.4.

[13] Chan, J. C., "An improved technique for circuit board interconnect test", In Proc. of IEEE Trans. on Inst. and Meas., Vol. 41, No. 5, pp. 692-698, 1992.

[14] C. P. Ravikumar and S. Chopra, "Testing interconnects in a system chip," In Proc. of Int. Conf. VLSI Design, 2000, pp. 388–391.

[15] K. Sekar and S. Dey, "LI-BIST: A low-cost self-test scheme for SoC logic cores and interconnects", In Proc. of IEEE VLSI Test Symp., 2002, pp. 417-422.

[16] A. Hławiczka, K. Gucwa, T. Garbolino, M. Kopec, "Can a D flip-flop based MISR compactor reliably detect interconnect faults?", In Proc. of IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems, April 2005.

[17] C.-A. Chen, S.K. Gupta, "BIST/DFT for performance testing of bare dies and MCMs", In Proc. of Electro'94, May 1994, pp. 803-812.

[18] X. Bai, S. Dey, and J. Rajski, "Self-test methodology for at-speed test of crosstalk in chip interconnects", In Proc. of Design Automation Conf., 2000, pp. 619–224.

[19] T. Garbolino, A. Hławiczka, A. Kristof, "A new idea of test-per-clock interconnect BIST structure", In Proc. of East-West Design & Test Workshop (EWDTW'04), Sept, 2004, pp. 23-29.

[20] Ulf Pillkahn, "Structural test in a board self test environment", In Proc. Of IEEE Int. Test Conf. (ITC'2000), Atlantic City, NJ, USA, Oct 3-5, 2000, pp. 1005-1012.

[21] N, Chunping, R. Zheping and Y. Zongzhong, "Five modified boundary scan adaptive test generation algorithms", J. Syst. Eng. Electron. Vol. 1, 2006, pp. 760-768.

[22] M. Kopec, T. Garbolino, K. Gucwa, A. Hlawiczko, "Test-per-clock detection, localization and identification of interconnect faults", In Proc. of eleventh IEEE European Test Symposium (ETS'06), 2006.

[23] Z. Navabi, VHDL: Modular Design and Synthesis of Cores and Systems, McGraw Hill, 1998.

A STUDY OF INSTRUMENT REUSE AND RETARGETING IN P1687

Farrokh Ghani Zadegan, Urban Ingelsson, Erik Larsson
Linköping University

Gunnar Carlsson
Ericsson

ABSTRACT

Modern chips may contain a large number of embedded test, debug, configuration, and monitoring features, called instruments. An instrument and its instrument data, instrument access procedures, may be pre-developed and reused and instruments may be accessed in different ways through the life-time of the chip, which requires retargeting. To address instruments reuse and retargeting, IEEE P1678 specifies a hardware architecture, a hardware description language, and an access procedure description language. In this paper, we investigate how P1687 facilitates instrument access procedure reuse and retargeting.

REUSE AND RETARGETING IN TEST STANDARDS

Design reuse is attractive because it reduces the IC design time and efforts, allows outsourcing portions of a large design to other companies, and therefore helps to manage the complexity of large designs through modular design approaches. However, when it comes to testing and verification, the benefits of design reuse cannot be completely utilized. For example, when a pre-developed test feature is reused and embedded inside another block of logic or another IC, the test patterns developed for that test feature—or in general the access procedures that describe how to use an on-chip feature—must be redeveloped. There are tools available to concatenate bit strings corresponding to the test patterns for each discrete IC on a PCB to perform tests at board-level. Such tools do not yet exist for operating on on-chip test features.

In recent years, the need for reusing the access procedures for embedded test, debug, and monitoring features, called on-chip instruments, has become more apparent, since modern chips may contain a large number of such embedded instruments. This abundance of on-chip instruments inside on-chip modules or IP blocks—developed by their respective designers for test and debug purposes—has given rise to the idea of reusing that wealth of on-chip instruments. Even though designed for a particular time in the life cycle of a chip, instruments can be accessed throughout its whole life cycle. To efficiently reuse instruments, it is necessary to automatically retarget the instrument data—or the access procedures described for a given instrument at its terminals—to the higher design levels up to the chip pins through an instrument access infrastructure.

If we want to make on-chip instrument access infrastructure available for off-chip usage, the ubiquitous IEEE 1149.1 (JTAG) [1] Test Access Port (TAP) is an attractive alternative. To implement and use JTAG, a set of languages are employed: to describe the JTAG circuitry there is the Boundary Scan Definition Language (BSDL), and to describe the JTAG TAP operations there are Serial Vector Format (SVF) [2] and Standard Test and Programming Language (STAPL) [3]. BSDL, however, is neither efficient nor sufficient to describe all types of instruments or the instrument access infrastructure. As an example, there are some SERDES implementations for which the Test Data Register (TDR) length varies depending on the performed operation. A variable length TDR is not supported by BSDL and therefore, a separate JTAG instruction should be used for each operation. Many JTAG instructions will lead to a long instruction register (IR) and complex decoding logic which may slow down the test clock (TCK). Similarly, SVF and STAPL do not lend themselves well to the design reuse practice inside the chip. For example, SVF does not support aliases or enumerations to make the code easy to reuse and to maintain. In particular, SVF and STAPL prove inefficient when we try to retarget the access procedures at the instrument's terminals to the chip terminals. As an example, currently there is no standard way for third-party vendors to provide procedures for accessing instruments inside their designed IP blocks—which are to be embedded in a larger design. Therefore, the test engineer has the responsibility of generating access procedures at the terminals of the IP block and translating those IP-level procedures to the system level. That is, even if we have the access procedures for a given instrument in the form of bit strings at its terminals, it is difficult to retarget those procedures to higher levels, since current EDA tools do the bit string concatenation only at the board level and not the chip-level. Another

alternative to describe both the instruments and the access procedures is to use the Core Test Language (CTL) of IEEE 1500. However, IEEE 1500 does not describe the chip-level network, and retargeting in IEEE 1500 is provided only from core terminals to core wrapper terminals.

Since current standards are limited in terms of describing instruments, reuse, and retargeting, IEEE P1687 [4] introduces two new languages, Instrument Connectivity Language (ICL) and Pattern Description Language (PDL), to standardize the access and control of on-chip instruments. In this paper, we introduce P1687 and compare the ICL/PDL pair with BSDL/SVF regarding their utility in reuse and retargeting of instrument access procedures.

P1687 specifies JTAG as off-chip to on-chip interface to the instrument access infrastructure (the P1687 *network*) and is informally called Internal JTAG (IJTAG). P1687 includes (1) specifications on the hardware that interfaces the on-chip instruments to the outside world (see the Appendix), (2) ICL which describes the instrument's port functions and logical connection to other instruments and to the JTAG TAP, and (3) PDL which describes how an instrument should be operated. The idea in introducing ICL and PDL is to provide an adequate and standardized description of the P1687 network, instruments, and instrument access procedures, to enable ICL and PDL interpreter tools to automate the retargeting of access procedures.

In comparing ICL/PDL with BSDL/SVF, note that ICL and BSDL both describe on-chip features (accessed through the JTAG TAP). However, ICL is used in conjunction with BSDL (when using a P1687 network through a JTAG TAP) and does not replace it. PDL, STAPL and SVF all describe how to use on-chip features. We make the comparison with SVF rather than with STAPL because SVF is currently a de facto standard at the board level to operate the JTAG TAP and interchange test data between EDA tools.

For this comparison, we start by describing a very simple scenario in which a temperature sensor is accessed through the JTAG TAP, using firstly BSDL/SVF and secondly ICL/PDL. Subsequently, we extend to multiple sensors, serially connected on the JTAG scan-path and make the scan path configurable (flexible) through the use of P1687-specific components.

A SMALL EXAMPLE: ACCESSING A SIMPLE INSTRUMENT

In this section, we illustrate how to access an embedded instrument using BSDL/SVF and ICL/PDL. We make use of a temperature sensor (see Figure 1) which, when enabled, makes the temperature available at its terminals as a 4-bit number, after 10 system clocks.

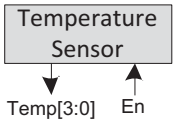


FIGURE 1 SHOWS THE TEMPERATURE SENSOR

Assume that we need to access our temperature sensor through the JTAG TAP as shown in Figure 2. One option is using BSDL to describe the JTAG circuitry, and SVF to describe how to access the sensor. P1687 provides another option which is using ICL in conjunction with BSDL to describe the network, and PDL to describe the access procedure for the sensor. We will compare the two alternatives in the following subsections.

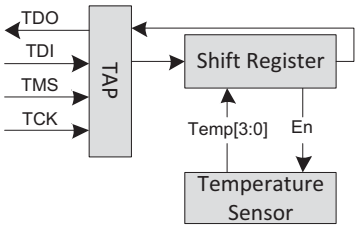


FIGURE 2 SHOWS THE CONNECTION OF THE TEMPERATURE SENSOR TO THE JTAG TAP THROUGH A SHIFT AND UPDATE REGISTER.

HOW IT IS DONE BY USING BSDL/SVF

Figure 2 shows a partial view of the JTAG circuitry that we have considered for this example. To keep the illustration easy to read, the rest of the mandatory components such as instruction register, instruction register decoder, etc. are not shown (see the Appendix for a more detailed view of a JTAG circuitry). Clock and control signals (e.g. shift enable) are not shown in Figure 2. Listing 1 shows the partial BSDL description of the JTAG circuitry shown in Figure 2. In Listing 1, we have replaced some lengthy parts of the code which are irrelevant to this discussion with (. . .).

LISTING 2 SHOWS THE SVF FILE

```
01 SIR 4 TDI (7);                    ! 4-bit IR scan => Loading READINSTR □("0111"□)
02 SDR 4 TDI (1);                    ! 4-bit DR scan => Activating the sensor □("0001"□)
03 STATE DRPAUSE;                    ! Going to state DRPAUSE
04 RUNTEST 10 SCK ENDSTATE DRPAUSE;  ! Waiting for 10 system clocks
05 SDR 4 TDI (0);                    ! Shifting out the temperature
06 STATE IDLE                        ! Going to state Run-Test/Idle
```

LISTING 1 THE PARTIAL BSDL DESCRIPTION FOR THE DESIGN IN FIGURE 2

```
01 entity SingleInstrumentChip is
02 generic (PHYSICAL_PIN_MAP : string := "DIP22_PACKAGE");
03
04 port (...);
05
06 use STD_1149_1_1990.all;
07
08 attribute PIN_MAP of SingleInstrumentChip : entity is PHYSICAL_PIN_MAP;
09 constant DIP22_PACKAGE : PIN_MAP_STRING :=  "...";
10
11 attribute TAP_SCAN_IN    of TDI : signal is true;
12 attribute TAP_SCAN_MODE  of TMS : signal is true;
13 attribute TAP_SCAN_OUT   of TDO : signal is true;
14 attribute TAP_SCAN_CLOCK of TCK : signal is (20.0e6, BOTH);
15
16 attribute INSTRUCTION_LENGTH of SingleInstrumentChip : entity is 4;
17 attribute INSTRUCTION_OPCODE of SingleInstrumentChip : entity is
18                         "BYPASS       (1111, 0000) , " &
19                         "EXTEST       (0001, 1001) , " &
20                         "SAMPLE       (0010, 1010) , " &
21                         "INTEST       (0011, 1011) , " &
22                         "HIGHZ        (0100, 1100) , " &
23                         "CLAMP        (0101) , " &
24                         "READINSTR    (0111, 1000)    ";
25
26 attribute INSTRUCTION_CAPTURE of SingleInstrumentChip : entity is "0001";
27 attribute INSTRUCTION_DISABLE of SingleInstrumentChip : entity is "HIGHZ";
28 attribute INSTRUCTION_GUARD   of SingleInstrumentChip : entity is "CLAMP";
29
30 attribute REGISTER_ACCESS of SingleInstrumentChip : entity is
31                         "BOUNDARY (EXTEST, INTEST, SAMPLE)," &
32                         "BYPASS   (BYPASS, HIGHZ, CLAMP)," &
33                         "INSTR[4]   (READINSTR)" ;
34                                        .
35                                        .
36                                        .
37 end SingleInstrumentChip;
```

For our discussion, the interesting points in Listing 1 are how a JTAG instruction called `READINSTR` is defined (Line 24) and how the 4-bit TDR that this instruction accesses is specified as `INSTR[4]` (Line 33). This 4-bit TDR represents the shift and update register in Figure 2.

Listing 2 shows a sample SVF script that reads our sensor according to the procedure stated above. We developed this SVF script assuming (as in Figure 2) that there are no other JTAG devices on the scan path, i.e. the TDI and TDO terminals of the TAP are directly connected to the external tester. Please note how the `READINSTR` instruction is loaded in Line 1 of Listing 2.

How It Is Done by Using ICL/PDL

Although ICL has some overlap with BSDL, it is not supposed to replace it. In fact, the JTAG TAP and boundary scan related circuitry will still be in the scope of JTAG. Therefore, for our small temperature sensor, we still need to use the BSDL in Listing 1 to describe the instruction opcode used for placing the TDR (shift register in Figure 2) between the TDI and TDO terminals. Listing 3 shows the ICL code that describes a P1687 network consisting of the sensor and a shift register, as well as how this network is interfaced to the JTAG TAP. For the sake of brevity, we have not shown and port-mapped the clock and control signals—i.e. ShiftEn, CaptureEn, and UpdateEn—which are required for the operation of the TDR_Sensor module.

LISTING 3 SHOWS ICL DESCRIPTION OF THE P1687 NETWORK IN FIGURE 2

```
01    Module Sensor {
02          DataInPort        en;
03          DataOutPort       temp[3:0];
04          }
05
06    Module TDR_Sensor {
07          ScanInPort        si;
08          ScanOutPort       so {  Source      SR[0];
09                                  LaunchEdge  Falling; }
10          DataInPort        pi[3:0];
11          DataOutPort       po {  Source SR[0]; }
12
13          SelectPort        en;
14
15          ScanRegister      SR[3:0]{ ScanInSource   si;
16                                     CaptureSource  pi; }

17          }
18
19    Module SingleInstrumentChip_ICL {
20          Instance TDR_Sensor_1 Of TDR_Sensor {
21                InputPort en = Tap1.en_TDR_Sensor;
22                InputPort pi = Sensor1.temp;
23                }
24          Instance Sensor1 Of Sensor {
25                InputPort en = TDR_Sensor_1.SR[0];
26                }
27          AccessLink Tap1 Of STD_1149_1 {
28                BSDL_Entity SingleInstrumentChip_ICL;
29                READINSTR {
30                      ScanPath {  TDR_Sensor_1; }
31                      ActiveSignals { en_TDR_Sensor ;}
32                      }
33                }
34          }
```

Listing 3 contains three modules, one describing the interface of the sensor (Lines 1-4), one describing the shift register that connects the sensor's terminals to the scan path (Lines 6-17), and the chip level module (Lines 19-34). The description of the sensor's interface contains the I/O ports shown in Figure 2, i.e. the *en* input terminal and the *temp* output terminal. The description of the shift register is more detailed since it contains both connection to the scan path (Lines 7-9) and connection to the sensor (Lines 10-12). The main component inside the TDR_Sensor module is *ScanRegister* (Lines 15-16) which is among the *primitive building blocks* specified by ICL.

For *ScanRegister*, the scan-in source and the parallel capture ports are described to connect it to the serial (scan) and parallel terminals of the TDR_Sensor module.

The chip level module, i.e. SingleInstrumentChip_ICL, puts it all together by instantiating the other modules (Lines 20-26) and connecting them appropriately to the BSDL description of the JTAG circuitry (Lines 27-33).

Listing 4 shows the PDL code that accesses the sensor as we had specified above. The PDL commands are categorized either as *setup* commands or as *action* commands. Setup commands configure the environment for the action commands, and action commands perform actual operations that make the setup commands take effect. For example, commands such as iTarget, iWrite, iRunLoop, and iRead are setup commands, whereas iPDLLevel and iApply are action commands which are immediately performed.

LISTING 4 SHOWS THE PDL CODE FOR READING THE TEMPERATURE SENSOR

```
01    iPDLLevel 0;
02    iTarget Sensor;
03
04    iProc Read_Temperature() {
05          iWrite      en    1;
06          iApply;
07
08          iRunLoop    10    -sck;
09          iApply;
10
11          iRead       temp;
12          iWrite      en    0;
13          iApply;
14    }
15
16    iTarget     SingleInstrumentChip_ICL;
17    iCall SingleInstrumentChip_ICL.Sensor1.Read_Temperature();
```

We will now examine the code in Listing 4 in detail. The iPDLLevel command (Line 1) chooses the Level-0 flavor of PDL which can be seen as a sequential set of actions without any flow control, which is sufficient for our simple example. The iTarget command (Line 2) specifies the ICL module (see Listing 3) for which the following commands are specified. The iProc command is used to specify a series of PDL commands for a given instrument as a group, which simplifies multiple repetitions of those commands and makes it possible to call them at higher levels. In our example, we only specify one procedure for reading the temperature from the sensor (Lines 4-14). To use the sensor, we enable the sensor (Lines 5-6), wait for 10 system clocks (Lines 8-9), and read the temperature while returning the *en* signal back to zero to make the sensor ready for the next read (Lines 11-13). It can be seen that multiple setup commands can be queued and applied concurrently with a single iApply command. The iTarget command in Line 16 tells the PDL interpreter that the next command is for the SingleInstrumentChip_ICL module, and the following iCall command (Line 17) retargets the access procedures for the temperature sensor, from its terminals (parallel access) to the boundary of the chip, i.e. the JTAG TAP (sequential access).

EXPANDING THE SMALL EXAMPLE

So far, for our small design, ICL and PDL required more lines of code compared to BSDL/SVF. However, for slightly larger designs, ICL/PDL show better maintainability and ease of use, which will be detailed in this section.

USING MULTIPLE INSTANCES OF OUR TEMPERATURE SENSOR

The first extension to consider is to use two instances of our sensor, as would be the case when there is a need to read temperatures of different areas of a chip. The impact of this on the BSDL code (for both BSDL/SVF and ICL/PDL scenarios) is that the length of the custom TDR `INSTR` which is described in Line 33 of Listing 1 as four, should be doubled since now there are two shift registers of length four on the scan path (that is, when the `READINSTR` JTAG command is loaded).

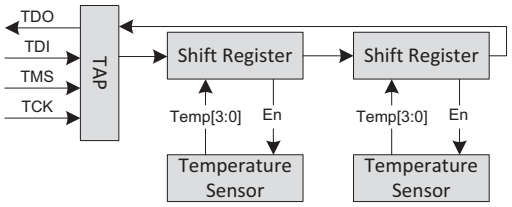


FIGURE 3: USING TWO INSTANCES OF THE SENSOR

Listing 5 shows the updated SVF code for reading the temperature from Sensor1. Since the length of scan path for the `INSTR` TDR has changed, any SDR command corresponding to `INSTR` operations should be updated, as can be seen in Line 2 and Line 5 of Listing 5. For reading both sensors, the length of the scan path, specified after each SDR command, does not change but the data bits that are to be scanned will change from 10 □(i.e. “00010000”□) to 11 (i.e. “00010001”□) to enable both sensors. It can be seen from this example that changes in the hardware or the access procedure can be tricky to be applied correspondingly to the SVF code, mainly because there is no link between the hardware description (BSDL) and the access procedure description (SVF).

LISTING 5 SHOWS THE UPDATED SVF CODE FOR FIGURE 3

```
01 SIR 4 TDI (7);                    ! 4-bit IR scan => Loading READINSTR □("0111"□)
02 SDR 8 TDI (10);                   ! 8-bit DR scan => Activating Sensor1 □("00010000"□)
03 STATE DRPAUSE;                    ! Going to state DRPAUSE
04 RUNTEST 10 SCK ENDSTATE DRPAUSE;  ! Waiting for 10 system clocks
05 SDR 8 TDI (0);                    ! Shifting out the temperature
06 STATE IDLE                        ! Going to state Run-Test/Idle
```

We will now examine the changes required for the P1687 alternative. Listing 6 shows the updates that are required in the ICL description. Since the same sensor and shift register components are used twice, the only part of code that should be modified is the chip level module that should instantiate two instances of the sensor and shift register modules (Lines 20-33), and add both shift registers to scan path (Lines 37-38).

LISTING 6 SHOWS THE UPDATED PDL CODE FOR FIGURE 3

```
                    .
                    .
                    .
19   Module SingleInstrumentChip_ICL {
20         Instance TDR_Sensor_1 Of TDR_Sensor {
21               InputPort en = Tap1.en_TDR_Sensor;
22               InputPort pi = Sensor1.temp;
23               }
24         Instance Sensor1 Of Sensor {
25               InputPort en = TDR_Sensor_1.SR[0];
26               }
27         Instance TDR_Sensor_2 Of TDR_Sensor {
28               InputPort en = Tap1.en_TDR_Sensor;
29               InputPort pi = Sensor2.temp;
30               }
31         Instance Sensor2 Of Sensor {
32               InputPort en = TDR_Sensor_2.SR[0];
33               }
34               AccessLink Tap1 Of STD_1149_1 {
35               BSDL_Entity SingleInstrumentChip_ICL;
36               READINSTR {
37                     ScanPath {  TDR_Sensor_1;
38                                 TDR_Sensor_2; }
39                     ActiveSignals { en_TDR_Sensor ;}
40                     }
41               }
42         }
```

As for the PDL code, no changes are required if we are only interested in reading Sensor1, otherwise, another iCall command, similar to the one in Line 17 in Listing 4, should be added for Sensor2.

It seems to us that from a developer's point of view, the PDL code is easier to read and to maintain as the complexity of the access procedures grows. Compared to SVF, PDL features aliases and enumerations which make the PDL code human readable. Moreover, PDL is a good means for documenting how to use an instrument.

USING A VARIABLE LENGTH SCAN-PATH

In the previous example shown in Figure 3, the shift-registers for the instruments were always on the scan-path which is not desirable in chips that have hundreds or more instruments, in particular if an access is made only to one or a subset of those instruments in a given access schedule. In such cases having all the instruments on the scan-path might unnecessarily incur a large access time overhead. Furthermore, in certain scenarios such as when instruments are located in different power islands, a single scan-path containing all the instruments will be broken when an island goes to a low-power mode [5]. Therefore, another interesting extension to our design will be to use the P1687-specified SIB module (see the Appendix on P1687 for basic information on SIBs) to add flexibility to the scan-path. Figure 4 shows how SIB components are added to the scan-path for our example.

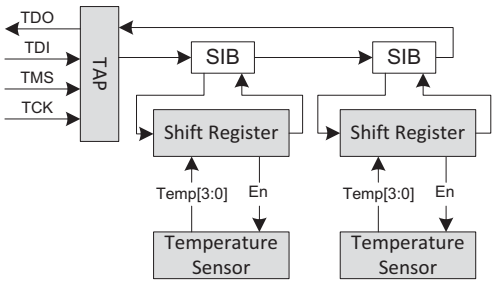


FIGURE 4: ACCESSING THE SENSORS INDIVIDUALLY

Again, Line 33 of the BSDL code (Listing 1) should be modified such that the length of `INSTR` is set to two, since now the initial scan path for INSTR consists of two SIBs. As for the SVF code for reading Sensor1, this example requires an additional step before accessing the shift registers. In the additional step, we need to open the SIBs on the scan path. Opening the SIBs requires a separate scan sequence (i.e. SDR command) in the SVF code, as can be seen in Listing 7, which opens the appropriate SIB to access Sensor1.

Regarding the ICL code modifications, we need to describe the SIB module and apply modifications to

LISTING 7 SHOWS THE UPDATED SVF CODE FOR FIGURE 4

```
01    SIR 4 TDI (7);                    ! 4-bit IR scan => Loading READINSTR □(“0111”□)
02    SDR 2 TDI (2);                    ! 2-bit DR scan => Opening SIB1 by shifting "10"
03    SDR 6 TDI (34);                   ! 6-bit DR scan => Activating Sensor1 by shifting "100010"
04    STATE DRPAUSE;                    ! Going to state DRPAUSE
05    RUNTEST 10 SCK ENDSTATE DRPAUSE;  ! Waiting for 10 system clocks
06    SDR 6 TDI (0);                    ! Shifting out the temperature
07    STATE IDLE                        ! Going to state Run-Test/Idle
```

the chip level module to instantiate the SIB components and describe the scan path correspondingly. Listing 8 shows the partial ICL code for the design in Figure 4. The Sensor and TDR_Sensor modules are not modified and therefore are not shown in Listing 8. Here again, we have not shown and port-mapped the clock and control signals for the SIB module. The PortGroup command used in the SIB module, guides the PDL interpreter in retargeting the access procedures. Besides instantiating and port-mapping the components required for the design in Figure 4, ScanPath (Line 59) is also updated to reflect that the scan-path is now through the SIBs.

LISTING 8 SHOWS THE UPDATED ICL CODE FOR FIGURE 4

```
                              .
                              .
                              .
19    Module SIB {
20          ScanInPort        si;
21          ScanInPort        fso;
22          ScanOutPort       so {  Source      SIB;
23                                  LaunchEdge Falling; }
24          SelectPort        en;
25          ToSelectPort      to_en;
26
27          ScanRegister      sr {  ScanInSource      mux1;
28                                  CaptureSource     1'b0;
29                                  ResetValue        1’b0; }
30          ScanMux                 mux1  sr {
31                                        1'b0 : si;
32                                        1'b1 : fso;
33                                  }
34          PortGroup         tap_side {si, so, en}
35          PortGroup         instrument_side {fso, to_en}
36          }
37
38    Module SingleInstrumentChip_ICL {
39          Instance sib1 Of SIB {
40                InputPort   en = Tap1.en_TDR_Sensor;
41                InputPort   fso = TDR_Sensor_1.so;
42                }
43          Instance TDR_Sensor_1 Of TDR_Sensor {
44                InputPort en = sib1.to_en;
45                InputPort si = sib1.so;
46                InputPort pi = Sensor1.temp;
47                }
48          Instance Sensor1 Of Sensor {
49                InputPort en = TDR_Sensor_1.SR[0];
50                }
51          /* sib2, TDR_Sensor_2, and Sensor2 are
52             instantiated and port-mapped as above
53                                  .
54                                  .
55                                  .          */
56          AccessLink Tap1 Of STD_1149_1 {
57                BSDL_Entity SingleInstrumentChip_ICL;
58                READINSTR {
59                      ScanPath {  sib1; sib2; }
60                      ActiveSignals { en_TDR_Sensor ;}
61                      }
62                }
63          }
```

Regarding the PDL code, again no change is required. It is expected in P1687 that the PDL interpreter will take care of opening the SIBs on the scan path and performing the scan operations required to access Sensor1, and therefore, retargeting of the access procedure of the sensors (from the sensor’s terminals, through the P1687 network, and to the JTAG TAP) is automatically performed.

From the above examples, it is becoming evident that without the help of a procedural description language such as P1687’s PDL, developing access procedures for on-chip instruments from the JTAG TAP (i.e. from an outside point of view) by using SVF is becoming difficult to maintain as the complexity and size of the on-chip instrument network increases.

In the above simple examples, we only used a small subset of ICL commands. However, ICL goes well beyond BSDL’s ability to describe the interface to a variety of instruments from simple ones to complex instruments such as MBIST engines, memories, and SERDESs, as well as scenarios such as multiple TAPs on a chip and direct (parallel) interfaces between instruments.

To summarize, by using P1687, the modular design approach can be utilized for test, monitoring, and etc. purposes as well by (1) reusing the access procedures for a given instrument (which can be among the deliverables for an IP block), and (2) retargeting those access procedures to any higher level of design and to any test access mechanism.

CHALLENGES IN ADOPTING P1687

In our discussion of accessing an embedded instrument from the JTAG TAP, we mentioned that ICL and PDL facilitate reuse and retargeting of instrument access procedures. In addition to EDA tools required for interpreting the ICL and PDL codes, we expect two additional tools, (1) a network construction tool for automatic generation of optimized ICL code which will address potential overheads, and (2) a SIB handling tool [5] for optimized operation of SIBs used in a variable length scan path.

As for the P1687 network construction tool, it should be noted that one should take into account the

instrument access time overhead, which might be considerable for large designs. In [6] the access time overhead is discussed for the scenario where the JTAG TAP is used to access the on-chip instruments. For example, chaining all instrument shift registers into a long scan-path, is not the best P1687 network design practice due to the prohibitively high access time overhead [6]. Moreover, such a long scan path is vulnerable to manufacturing defects such as stuck at faults in the chain that could potentially render the whole chain useless. P1687 specifies the SIB module to be used to form a variable length scan path so that it becomes possible to include/exclude instruments as needed. However, a long flat [6] scan-path might also prove inefficient when some instruments are to be accessed more frequently than the others—a situation that can be handled, again by using SIBs to add hierarchical levels to the P1687 network [7]. Hierarchical network design may also solve the issue with the instruments being in different power islands [5]. Such a network construction tool is not yet available from the industry.

Regarding the SIB handling tool, given a well designed P1687 network with a variable length scan path, it is still required that based on the power constraints, resource conflicts, etc., SIBs are operated (i.e. opened and closed) in a way that the constraints are satisfied and the access time overhead is minimized. This smart handling [5] of SIBs is another area which requires support from the EDA industry.

BIBLIOGRAPHY

[1] IEEE Std 1149.1-2001, IEEE Standard Test Access Port and Boundary-Scan Architecture, 2001.

[2] (1999) Serial Vector Format Specification. [Online]. http://www.asset-intertech.com/support/svf.pdf

[3] (1999, Aug) STANDARD TEST AND PROGRAMMING LANGUAGE (STAPL). [Online]. http://www.jedec.org/standards-documents/results/STAPL

[4] IJTAG. [Online]. http://grouper.ieee.org/groups/1687/

[5] Al Crouch. (2011) IEEE P1687 Internal JTAG (IJTAG) taps into embedded instrumentation. [Online]. http://www.asset-intertech.com/pressroom/whitePapers/IEEE_P1687_IJTAG_Whitepaper.pdf

[6] F. Ghani Zadegan, U. Ingelsson, G. Carlsson, and E. Larsson, Access Time Analysis for IEEE P1687, doi: 10.1109/TC.2011.155, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5999657&isnumber=4358213.

[7] F.G. Zadegan, U. Ingelsson, G. Carlsson, and E. Larsson, "Design automation for IEEE P1687," in *proc. Design, Automation & Test in Europe Conference & Exhibition (DATE) 2011*, Grenoble, 14-18 March 2011.

APPENDIX

The on-chip P1687 network is interfaced to the JTAG TAP by using a special TDR called (Level-0) Gateway module. One interesting feature in the P1687 specification is the concept of a variable length scan-path which can be achieved by using a module called Segment Insertion Bit or SIB for short. A SIB is a 1-bit shift and update register on the scan-path which can be programmed to insert another segment of P1687 scan-path into the current (active) scan-path—hence the name Segment Insertion Bit. It is possible to build a multitude of different hierarchical P1687 networks by using SIBs. The Gateway itself can be composed of one or more SIBs. Figure 5 shows a small P1687 network and its connection to the JTAG TAP. The shown network also illustrates the concept of a variable length scan-path achieved by using the SIB modules.

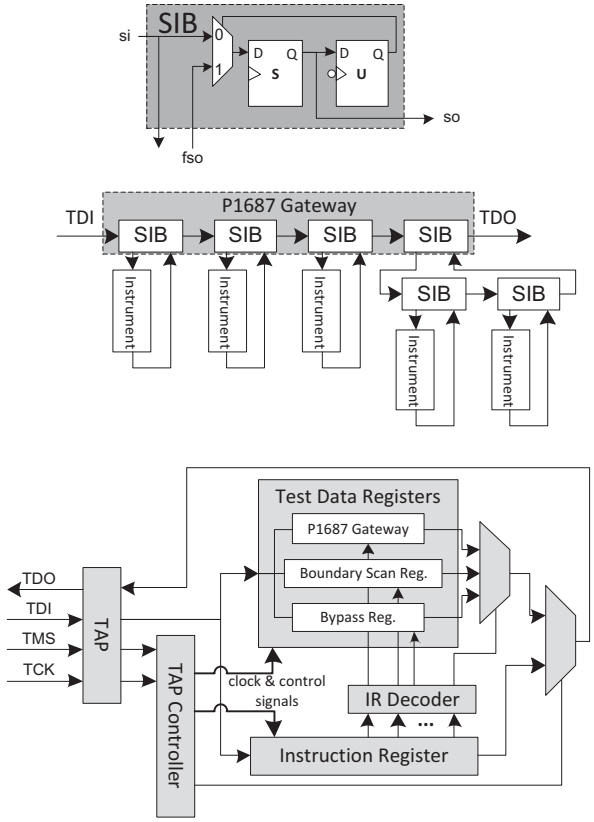


FIGURE 5 SHOWS A SAMPLE JTAG CIRCUITRY, A SAMPLE P1687 NETWORK, AND HOW THEY ARE INTERFACED

A Test Compaction Oriented Don't Care Identification Method

Hiroshi YAMAZAKI† Motohiro WAKAZONO††
Toshinori HOSOKAWA‡ Masayoshi YOSHIMURA‡‡

†Graduate School of Industrial Technology, Nihon University, 1-2-1 Izumicho, Narashino, Chiba 275-8575, Japan
††College of Industrial Technology, Nihon University, 1-2-1 Izumicho, Narashino, Chiba 275-8575, Japan
‡College of Industrial Technology, Nihon University, 1-2-1 Izumicho, Narashino, Chiba 275-8575, Japan
‡‡Graduate School of Information Science and Electrical Engineering, Kyushu University
744 Motooka, Nishi-ku, Fukuoka, Fukuoka 819-0395, Japan

E-mail: †cihi10017@g.nihon-u.ac.jp, ††gamgam129@hotmail.com
‡hosokawa.toshinori@nihon-u.ac.jp, ‡‡yosimura@ait.kyushu-u.ac.jp

Abstract *In recent years, the growing density and complexity of VLSIs have led to an increase in the number of test patterns and fault models. These test patterns must be of high quality and low cost. One test generation method satisfying these requirements is the don't care identification technique. However, conventional don't care identification techniques are not very effective for application-specific fields such as test compaction. In this paper, we propose a test compaction oriented don't care identification method. The experimental results for ITC'99 and ISCAS'89 benchmark circuits show that a given test set can be efficiently compacted by the proposed method.*

keywords**:** *don't care identification, care bit distribution, test compaction, essential faults*

1. Introduction

In recent years, the growing density and complexity of very-large-scale integration (VLSI) circuits has caused an increase in the number of test patterns and fault models. Not only all detectable stuck-at faults but also defects modeled by other faults, such as bridging faults [1] and transition faults [2][3], need to be detected. Therefore, additional test patterns are necessary to detect the defects of new fault models. Because the test cost is generally proportional to the number of test patterns, the test cost increases with the increase in the number of test patterns.

Test compaction [4] is one of the methods solving the problem of the increased number of test patterns. Test compaction reduces the number of test patterns by merging a test pattern with other compatible test patterns [5].

Some of the primary input values in a set of test patterns may be changed to opposite logic values without losing fault coverage. Such input values can be regarded as don't care (X). Generally, every primary input value of test patterns is specified to be either 0 or 1. Some of the input values in test patterns generated by the Automatic Test Pattern Generator (ATPG) may be X-bits.

The X-identification (XID) method proposed in [6] identified many don't care inputs of test patterns in a given test pattern set. In XID, X-bits were identified in earlier test patterns. XID doesn't take the X-bit distribution for test patterns into consideration. Therefore, an X-identification method may be less effective for test pattern compaction and test power reduction. In reference [7], a distribution for a test pattern-controlling X-Identification (DTC-XID) method was proposed to solve the problem of test power reduction. DTC-XID averages the number of detected faults for each test pattern. As the result, only the almost equivalent percentages of X-bits are identified in each test pattern. Although DTC-XID is effective for test power reduction, DTC-XID does not consider the X-bit distribution for primary inputs (PI). Therefore, DTC-XID may be less effective for test pattern compaction.

In this paper, we propose a test compaction oriented X-identification method. The proposed method consists of two techniques. One technique reduces the number of test patterns to detect hard-to-detect faults. Hard-to-detect faults are defined as faults whose number of detections is less than N. The other technique controls the X-bit distribution for primary inputs.

This paper is organized as follows. Section 2 shows definitions and a basic algorithm for the X-identification methods [6]. Section 3 shows an overview of the proposed test compaction oriented X-identification method. Section 4 describes the details of the X-identification method for hard-to-detect faults. Section 5 describes the details of the method to control the X-bit distribution for each primary input. Section 6 shows the experimental results for ISCAS'89 and ITC'99 benchmark circuits. Finally, Section 7 concludes the paper and discusses future work.

2. Background and Motivation

2.1. Don't Care Bits

Some of the specified primary input values in a set of test patterns may be changed to the opposite logic values without losing fault coverage. Such input values can be regarded as don't care (X). The specified input value X can be set to the logic value "0" or "1". The don't care bit is denoted as "X" or "x" in a test pattern.

2.2. Formulation of X-Identification

In this paper, a test pattern set T is given and generated by the ATPG. Every primary input value in a test pattern is specified to be either 0 or 1. Given a circuit C and a fully specified test pattern set T, we compute the test pattern set T', including some Xs (don't cares), where the test pattern set T' has the following properties.

(1) T' covers T.
(2) T' contains as many Xs as possible.
(3) The fault coverage of T' is equal to that of T.

We show a simple example of an X-identification for a single stuck-at fault. Suppose that test pattern set T in Table I(a) is generated for the circuit shown in Fig. 1. Test pattern set T' in Table I(b) is one of the solutions. Test pattern t_1 detects faults $a/0$, $b/0$ and $c/1$, where s/v denotes the stuck-at $v \in \{0, 1\}$ fault on the signal line s. Fault $a/0$ has to be detected by t_1, because any other test pattern cannot detect $a/0$. In contrast, fault $c/1$ does not have to be detected by t_1 because t_3 detects $c/1$, too. Hence the value

Table I. Test pattern set

(a) T

	abc
t_1	110
t_2	101
t_3	010
t_4	011

(b) T′

	abc
t'_1	11X
t'_2	101
t'_3	010
t'_4	X11

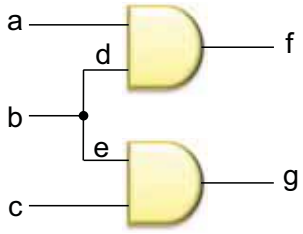


Fig. 1. Example circuit

0 at input c of t_1 becomes an X. Similarly, the value 0 at input a of t_4 becomes an X. Thus, test pattern set T' in Table I(b) is obtained.

2.3. Basic X-Identification Algorithm

Fig. 2 shows a basic procedure to obtain T' from T, where C is a given circuit and T is a given test pattern set. In Step 1, a fault simulation is performed for each test pattern t_i. In Step 2, the essential faults [8] of test pattern t_i are collected. From the result of the fault simulation, the primary input values of t_i sufficient for detection of the essential faults in Step 3 are calculated. The primary input values fix the care bits of t'_i, and the other values are X-bits. As t'_i may detect faults other than the essential faults, a fault simulation is performed for t'_i in Step 4. From the result of the fault simulation, a list of undetected faults is made. From Steps 2 to 4, the initial test pattern set T' and its undetected fault set can be obtained. T' is the test pattern set that can detect all essential faults.

Since some faults are not detected by T', some of the Xs in t'_i are returned to the original values of t_i such that all of the detectable faults are detected. In Step 5 the faults undetected by t'_i but detectable by t_i are detected. Let a set of the collected faults be G. Then, the logic values of t_i sufficient for detection of the faults in G in Step 6 are calculated and added to the values of the primary inputs of t'_i. In Step 7, a fault simulation is performed for test pattern t'_i to update the undetected fault set. As a result, we obtain the test pattern set T' that detects all faults and contains many Xs.

```
Procedure X-search(C,T)
 Circuit C ;  Test_set  T ;
{
      for each testpattern t_i in T {
            fault_similation(t_i) ;                    Step 1.
      }
      for each testpattern t_i in T {
            F = collect_essential_fault(t_i) ;         Step 2.
            t'_i = find_value(F) ;                     Step 3.
            fault_simulation(t'_i) ;                   Step 4.
      }
      for each testpattern t_i in T {
            G = collect_undetected_fault(t_i) ;        Step 5.
            t'_i = find_value(G) ;                     Step 6.
            fault_simulation(t'_i) ;                   Step 7.
      }
      return T' composed of t'_i ;
}
```

Fig. 2. Conventional X-identification algorithm

2.4. Motivation

The *X*-identification methods proposed in [6][7] do not consider the X-bit distribution for primary inputs. Therefore, the *X*-identification method [6][7] may be less effective for test compaction.

Fig. 3 shows an example of test compaction for the test pattern set. The number of *X*-bits for each primary input in the test pattern set is not uniform. The X-concentrated primary input defined as the number of *X*-bits for each primary input is also not uniform. Before test compaction, t'_1 to t'_5 denote the test pattern ID. After test compaction, ct'_1 to ct'_4 denote the test pattern ID. The primary input IDs are denoted as p_1 to p_7. Test patterns t'_1 to t'_5 have no *X*-bits at primary inputs p_1 and p_2. As the result, test compaction reduces the number of test patterns from 5 to 4.

Fig. 4 shows an example of the test compaction of the test pattern set. The number of *X*-bits for each primary input in the test pattern set is uniform. Every primary input has two *X*-bits. Test compaction reduces the number of test patterns from 5 to 3. As described above, the distribution of the *X*-bits for each primary input may affect the efficiency of test compaction.

3. Test Compaction Oriented Don't Care Identification

In this section, we propose test compaction oriented X-identification. The proposed method consists of two techniques. One technique is X-identification for hard-to-detect faults. Hard-to-detect faults are defined as faults whose number of detections is less than *N*. The details of this technique are described in Section 4. Another technique is X-identification to equalize the

Test pattern set with X-concentrated primary input

	p_1	p_2	p_3	p_4	p_5	p_6	p_7
t'_1	1	0	0	x	x	0	x
t'_2	0	0	x	1	1	x	1
t'_3	1	0	x	0	x	0	x
t'_4	0	x	0	x	0	x	x
t'_5	1	1	1	0	x	x	1

Test compacted test pattern set

	p_1	p_2	p_3	p_4	p_5	p_6	p_7
ct'_1	1	0	0	0	x	0	x
ct'_2	0	0	x	x	1	x	1
ct'_3	0	x	0	x	0	x	x
ct'_4	1	1	1	0	x	x	1

Fig. 3. Example of test compaction on test pattern set with X-concentrated primary input

Test pattern set with X-unconcentrated primary input

	p_1	p_2	p_3	p_4	p_5	p_6	p_7
t'_1	1	0	0	x	1	0	x
t'_2	x	0	x	1	1	x	1
t'_3	1	0	x	0	x	0	1
t'_4	x	1	0	1	0	x	x
t'_5	1	x	1	0	x	0	1

Test compacted test pattern set

	p_1	p_2	p_3	p_4	p_5	p_6	p_7
ct'_1	1	0	0	1	1	0	1
ct'_2	1	0	1	0	x	0	1
ct'_3	x	1	0	1	0	x	x

Fig. 4. Example of test compaction on test pattern set with X-unconcentrated primary input

number of Xs for each primary input. The details of this technique are described in Section 5. Fig. 5 shows the test compaction oriented X-identification algorithm.

(Step 1)

A fault simulation is performed for a circuit by each test pattern t_i. From the result of the fault simulation, a detected faults list is made.

(Step 2)

Essential faults are collected from the detected faults list made by Step 1. The primary input values of t_i sufficient for detection of the essential faults are calculated. These primary input values are fixed care bits, and the other values are X. Then, T' is obtained.

(Step 3)

As T' may detect faults besides the essential faults, a fault simulation is performed by T'. From the result of the fault simulation, an undetected faults list (UD) is made.

(Step 4)

X-identification is performed for hard-to-detect faults. As the result of the X-identification, test pattern set T' is updated. The details are described in Section 4.

(Step 5)

X-identification is performed to equalize the number of Xs for each primary input for the undetected faults of T'. As the result of the X-identification, test pattern set T' is updated. The details are described in Section 5.

```
Procedure test_compaction_oriented_X-Identification(C,T)
 Circuit C ;   Test_Pattern_Set  T ;
{
     for each testpattern t_i in T {
          fault_similation(t_i ) ;                          Step 1.
     }
     T' = essential_Xidentification(T ) ;                   Step 2.
     UD = collect_undetected_fault(C, T' ) ;                Step 3.
     (T', UD) = hard_to_detect_X-filling(C, T, T', UD) ;    Step 4.
     T' = distribution_X-filling(C, T, T', UD) ;            Step 5.
     return (T' ) ;
}
```

Fig. 5. Algorithm of test compaction oriented X-identification

```
Procedure hard_to_detect_X-filling(C, T, T', UD, N , M)
circuit C ;   First_Test_Pattern_Set T ;
Essential_Fault_Test_Set T' ;  Undetected_Fault_List UD ;
Detect_Number_Of_Timese N ;
Target_Test_Pattern_Number   M;
{
     (HD, S) = Search_hard_to_detect_fault(T, M) ;      Step 4-1.
     for (each t_j∈S ) {
          t'_j  = X_filling(C, t_j , t'_j  ) ;          Step 4-2.
          UD = fault_simulation(C, UD, t'_j ) ;         Step 4-3.
     }
     return(T', UD) ;
}
```

Fig. 6. Hard-to-detect fault X-identification algorithm

4. X-Identification for Hard-to-detect Faults

As stated above, hard-to-detect faults are faults whose number of detections is less than N. It is assumed that many care bits are requested to detect hard-to-detect faults. Test patterns containing many care bits have a low probability of compaction. Therefore, it is assumed that test patterns to detect hard-to-detect faults have a low probability of compaction. In addition, if many test patterns to detect hard-to-detect faults are available, it is assumed that the efficiency of the test compaction becomes low. The X-identification algorithm selects one test pattern from test patterns in T which can detect a fault. This processing is called a test pattern assignment. The X-bit distribution in test pattern set T' depends on the test pattern assignment. A test pattern assignment method to reduce the number of test patterns for detection of hard-to-detect faults is proposed. An X-identification algorithm with the test pattern assignment is also proposed.

In this section, we describe a heuristic algorithm for hard-to-detect fault X-identification (Step 4 of Fig. 5). Fig. 6 shows the X-identification algorithm for hard-to-detect faults.

First, a fault simulation is performed for undetected fault set UD in a circuit by test pattern set T. As the result of the fault simulation, the hard-to-detect fault set HD is obtained. The detection time of each fault in HD is less than N. Test pattern set S is M test patterns which are selected in descending order of detection time of hard-to-detect fault from the test pattern set T (Step4-1). M can be specified by the user. The processing from Steps 4-2 to 4-3 is iterated for each test pattern in S. The test pattern $t_j \in S$ is processed in descending order of the number of detected faults in HD. First, $t'_j \in T'$ corresponding to $t'_j \in S$,T is applied, at which time t'_j detects essential faults. Next, care bits are assigned to primary inputs such that t'_j can detect all the faults in HD that are detected by t_j (Step 4-2). Thus, t'_j is updated. A fault simulation is performed for the faults in UD by using t'_j, and the detected faults are deleted from UD (Step 4-3). The test pattern set T' that can detect essential faults, hard-to-detect faults, and the undetected fault set UD of T' are returned.

5. Care Bit Distribution for Control X-Identification

In this section, a don't care identification technique is shown to control the X-bit distribution for each primary input in a test set. This technique is used in Step 5 of Fig. 5. The inputs of this technique are the test pattern set T for which all primary input values have been specified, the test pattern set T' to detect essential faults containing X-bits, and the undetected faults list UD. The output of this technique is test pattern set T' containing X-bits that detect all the faults in UD. Thus, T' can detect all faults detected by T.

In Section 5.1 we describe the evaluation function for care bit distribution control, and in Section 5.2 we describe a matching method of faults and test patterns. We describe the care bits of the primary input distribution control X-identification algorithm in Section 5.3.

5.1. Evaluation Function

Here, we present the evaluation function for control of the care bit distribution. First, we denote formula (1) for $W(p_j)$, which is the number of care bits for primary input p_j in test pattern set T' that includes some Xs.

$$W(p_j) = \sum_{i=1}^{TP} b(t'_{i,}\, p_j) \qquad (1)$$

TP is the number of test patterns in T'. If the primary

input value of p_j in $t'_i \in T'$ is a care bit, $b(t'_i,p_j)$ returns 1. If t'_i is a don't care bit, $b(t'_i,p_j)$ returns 0.

The calculated results of $W(p_j)$ for T' are shown in Table II. Here, x denotes a don't care bit, c denotes a care bit, t'_1 to t'_5 denote the test patterns, and p_1 to p_7 denote the primary inputs.

As an example, the number of care bits for the primary input p_1 is calculated. The primary input p_1 includes the care bits in test patterns t'_2 and t'_4, as shown in Table II. Therefore, $w(p_1)=2$ is calculated by formula (1).

5.2. Matching Method of Faults and Test Patterns

In this section, the matching method of faults and test patterns is described. Because it is necessary to detect an undetected fault f by at least one test pattern, a test pattern in T must be selected to detect the undetected fault f. The test pattern in T that has the minimum values of the cost function is selected. The cost function is defined by formula (2).

$$MW(t'_i, t''_i) = \sum_{j=1}^{I} W(p_j) \times d(t'_i, t''_i, p_j) \qquad (2)$$

In formula (2), $t'_i \in T'$ is an element in test pattern set T' that detects essential faults and hard-to-detect faults, t''_i is a test pattern corresponding to $t_i \in T$ and detects fault f. Thus, the care bits of t''_i are the values assigned to detect only fault f.

I denotes the number of primary inputs. If the value of the primary input p_j in a test pattern t''_i is a care bit, and the value of p_j primary inputs in a test pattern t'_i is X-bits, $d(t'_i, t''_i, p_j)$ returns 1. Otherwise, $d(t'_i, t''_i, p_j)$ returns 0.

Table III shows the care bits to detect fault f. Fault f is detected by test patterns, t_1, t_3, and t_5. $W(p_j)$ and $MW(t'_i, t''_i)$ are also shown in Table III.

Table II. Care bits in each test pattern and $W(p_j)$

	p_1	p_2	p_3	p_4	p_5	p_6	p_7
t'_1	x	x	c	c	x	c	x
t'_2	c	x	x	c	c	x	c
t'_3	x	x	c	c	x	x	x
t'_4	c	x	c	c	c	x	x
t'_5	x	x	c	c	x	c	x
$W(P_j)$	2	0	4	5	2	2	1

Table III. Care bits to detect fault f

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	$MW(t'_i,t''_i)$
t''_1	c	c	c	x	x	c	c	3
t''_3	x	c	c	c	x	x	c	1
t''_5	c	c	x	x	x	c	x	2
$W(P_j)$	2	0	4	5	2	2	1	

As an example calculation, we consider $MW(t'_1, t''_1)$, which is the value of the cost function required for test pattern t'_1 and t''_1 to detect fault f. If test pattern t''_1 detects fault f, the values of primary inputs p_1, p_2, p_3, p_6, and p_7 are care bits. $MW(t'_1, t''_1)$ is calculated as follows.

$$\begin{aligned} MW(t'_1,t''_1) = {} & W(p_1)\times d(t'_1, t''_1, p_1) \\ & + W(p_2)\times d(t'_1, t''_1, p_2) \\ & + W(p_3)\times d(t'_1, t''_1, p_3) \\ & + W(p_4)\times d(t'_1, t''_1, p_4) \\ & + W(p_5)\times d(t'_1, t''_1, p_5) \\ & + W(p_6)\times d(t'_1, t''_1, p_6) \\ & + W(p_7)\times d(t'_1, t''_1, p_7) \\ = {} & (2\times 1)+(0\times 1)+(4\times 0) \\ & +(5\times 0)+(2\times 0)+(2\times 0)+(1\times 1) \\ = {} & 2+0+0+0+0+0+1 = 3 \end{aligned}$$

The cost value of t'_1 and t''_1 is 3. The value of primary inputs p_3 and p_6 of test pattern t'_1 are care bits, as shown in Table II. Therefore, $d(t'_1,t''_1,p_3)$ and $d(t'_1,t''_1,p_6)$ are 0. Similarly, $MW(t'_i, t''_i)$ is calculated for test pattern t_3 and t_5. $MW(t'_3, t''_3)$ and $MW(t'_5, t''_5)$ are 1 and 2, respectively. The results of the values of the cost function reveal the minimum value is $MW(t'_3, t''_3)$. Therefore, test pattern t_3 to detect fault f is selected.

5.3. Care Bits of Primary Input Distribution Control X-Identification Algorithm

Fig. 7 shows the X-identification algorithm with the X-bit distribution for primary inputs. Steps 5-1 to 5-7 are iterated for each fault in UD. An undetected fault f is selected from an undetected fault set UD (Step 5-1). A test pattern set DT_f, which detects f, is obtained (Step 5-2). The care bits to detect t_i are calculated for all test patterns in DT_f. Let T'' be a test pattern set having care bits to detect only f. A test pattern t''_i is obtained by identifying the primary input values assigned to detect undetected fault f for test pattern t_i in DT_f. $MW(t'_i, t''_i)$ is calculated using all test patterns in T'' (Step 5-4). Then the minimum

```
Procedure distribution_X-filling(C, T, T'', UD )
circuit C ;    Initial_Test_PatternSet T ;
Hard_To_Detect_Fault + Essential_Fault_Test_Pattern_Set T'' ;
Undetected_Fault_List UD ;
{
    for (each fault in UD) {
        f  = collect_no_detect_fault(UD) ;              Step 5-1.
        DT_f = collect_detect_test_pattern(f ,T ) ;     Step 5-2.
        T''  = find_value(DT_f ) ;                      Step 5-3.
        caluclation_const(T', T'' ) ;                   Step 5-4.
        t''_i = collect_low_cost_pattern(T', T'' ) ;    Step 5-5.
        t'_i = x_filling(t''_i ) ;                      Step 5-6.
        UD = fault_simulation(C, UD, t'_i ) ;           Step 5-7.
    }
    return T' ;
}
```

Fig. 7. X-bit distribution for primary inputs with the X-identification algorithm

value of $MW(t'_i, t''_i)$ is selected. If the values of the primary input p_j in t'_i and t''_i are an X-bit and a care bit, the X-bit value of the primary input p_j in t'_i is updated to a care bit (Step 5-6). A fault simulation is performed for undetected fault set UD by t'_i and the detected faults are found in UD (Step 5-7). Finally, T' is returned.

6. Experimental Results

In this section, we describe the evaluation results of the proposed method. The evaluation items are the don't care identification rates and the number of test patterns after test compaction. The proposed method was compared with XID [6] and DTC-XID [7]. The evaluation circuits were ITC'99 benchmark circuits and ISCAS'89 benchmark circuits. The initial test pattern set T was generated with the ATPG tool "TetraMAX™" from Synopsys. The target fault model was a single stuck-at fault model. Two test pattern sets were prepared for each circuit. One was a compacted test pattern set. The other was an uncompacted test pattern set.

In the experiment, the detection time of hard-to-detect faults was less than 3. In Proposed (1), M was 0% of the size of the test pattern set. In Proposed (2), M was 1% of the size of the test pattern set.

Table V shows the don't care identification rate of the uncompacted test pattern set. In Table V, *#X-vari* denotes the variance of the primary inputs, *#X-min* denotes minimum percentage of X-bits identified, *%X-ave* denotes the don't care identification rate. The don't care identification rates of Proposed (1) and Proposed (2) are higher than those of XID and DTC-XID in all circuits.

Table VI shows the don't care identification rate of the compacted test pattern set. The don't care identification rate of the compacted test pattern is lower than the uncompacted test pattern in all circuits. The don't care identification rates of Proposed (1) and Proposed (2) are approximately 1% lower than those of XID in b14, b20, b21, and b22.

Table VII shows the number of test patterns in the test pattern set after test compaction [5][8][9]. The test pattern set before test compaction was obtained by X-identification for the uncompacted test pattern set. The numbers of test patterns of Proposed (1) and Proposed (2) are lower than those of XID and DTC-XID in all circuits except s35932. The numbers of test patterns of Proposed (1) and Proposed (2) are one test pattern higher than that of XID in s35932. Comparing Proposed (1) and Proposed (2), the number of test patterns of Proposed (2) is lower than that of Proposed (1) in s13207, b20, and b21. For the other circuits, the number of test patterns of Proposed (1) is lower than that of Proposed (2). These results indicate that the parameter used in Step 4 and the number of test patterns to detect hard-to-detect faults for these circuits were not effective.

Table VIII shows the number of test patterns in the test pattern set after test compaction. The test pattern set before test compaction was obtained by X-identification for the compacted test pattern set. The numbers of test patterns of Proposed (1) and Proposed (2) are lower than those of XID and DTC-XID in all circuits except b15. In b15, XID has the smallest number of test patterns.

7. Conclusion

In this paper, we proposed a test compaction oriented don't care identification method. The experimental results showed that the number of test patterns is reduced for most circuits. The experimental results also showed that the number of test patterns to detect hard-to-detect faults affects the number of final test patterns. Future work includes improving the test compaction oriented don't care identification and studying a don't care identification for other fault models.

References

[1] Yuzo Takamatsu, Tomoko Shiosaka, Teruhiko Yamada and Koji Yamazaki, "A Fault Model and Test Generation for Bridging Faults in CMOS Circuits," IEICE D-I, Vol. J81-D-I, No. 6, pp. 872-879, June 1998 (in Japanese).

[2] Sying-Jyan Wang and Yan-Ting Chen, "Low Capture Power Test Generation for Launch-off-Capture Transition Test Based on Don't-Care Filling," Circuits and Systems, 2007, ISCAS 2007. IEEE International Symposium, 27-30, pp. 3683-3686, May 2007.

[3] Angela Krstić,Kwang-Ting Cheng, Cheng, "Delay Fault Testing for VLSI Circuits," Kluwer Academic Publishers, 1998.

[4] Niraj Jha and Sadeep Gupta, "Testing of Digital Systems," Cambridge University Press, 2002.

[5] Kohei Miyase, Seiji Kajihara, Sudhakar M. Reddy, "A Method of Static Test Compaction Based on Don't Care Identification," IPSJ journal, Vol. 43, No. 5, pp. 1290-1293, May 2002.

[6] K. Miyase, S. Kajihara, "XID: Don't Care Identification of Test Patterns for Combinational Circuits," IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, Vol. 23, No. 2, pp. 321-326, Feb. 2004.

[7] Kohei Miyase, Kenji Noda, Hideaki Ito, Kazumi Hatayama, Takashi Aikyo, Yuta Yamato, Hiroshi Furukawa, Xiaoqing Wen,Seiji Kajihara, "Effective IR-Drop Reduction in At-Speed Scan Testing Using Distribution-Controlling X-Identification," IEEE/ACM International Conference on Computer-Aided Design, pp. 52-58, 2008.

[8] Seiji Kajihara, Irith Pomeranz, Kozo Kinoshita and Sudhakar M. Reddy, "Cost-Effective Generation of Minimal Test Sets for Stuck-at Faults in Combinational Logic Circuits," 30th ACMIEEE Design Automation Conference, pp. 102-106, 1993.

[9] Masayuki Arai, Satoshi Fukumoto, Kazuhiko Iwasaki, Takahisa Hiraide, Takashi Aikyo, "Test Data Compression Using TPG Reconstruction for BIST-Aided Test," proc. IEEE 6th Workshop on RTL and High Level Testing,211-8588, 2005.

[10] Jeremy Lee, Mohammad Tehranipoor, "LS-TDF: Low-Switching Transition Delay Fault Pattern Generation," VTS, 26th IEEE VLSI Test Symposium (VTS 2008), pp. 227-232, 2008.

[11] Kohei Miyase, Kenta Terashima, Seiji Kajihara, Xiaoqing Wen and Subhakar M. Reddy, "On Improving Defect Coverage of Stuck-at Fault Tests," In Proc. the 14th Asian Test Symposium (ATS '05), pp. 216~223, 2005.

[12] D. Brelaz, "New Methods to Color the Vertices of a Graph, " Communications of the ACM, 22, pp. 251-256, 1979.

Table V. X-bit rates of uncompacted test pattern sets

Circuits	#PIs	XID			DTC-XID			Propose(1)			Propose(2)		
		#X-vari	#X-min	%X-ave	#X-vari	#X-min	%X-ave	#X-vari	#X-min	%X-ave	#X-vari	#X-min	%X-ave
s13207	700	2145	266	94.3	1894	266	95.8	1536	301	**96.0**	1563	298	95.9
s15850	611	1031	284	94.0	1106	230	94.3	756	333	**94.7**	783	323	94.6
s35932	1763	16	5	86.1	13	0	87.9	9	15	**88.7**	9	14	88.5
s38417	1664	1498	936	97.2	1101	958	97.5	725	1016	**97.8**	771	1008	97.7
s38584	1462	1849	51	96.3	1552	0	96.5	970	273	**96.7**	1020	272	**96.7**
b14	277	78202	94	82.1	76544	3	80.0	56149	294	**82.7**	56644	289	82.4
b15	485	14110	70	90.1	15844	32	89.6	8634	253	**91.9**	8669	248	**91.9**
b20	522	61611	902	87.0	60665	891	85.5	41923	1132	**87.6**	43025	1112	87.3
b21	522	68620	1056	87.9	67145	1019	86.6	48045	1277	**88.3**	48702	1255	88.1
b22	767	57468	1278	91.2	60325	1145	90.4	40205	1546	**91.8**	41449	1523	91.5

Table VI. X-bit rates of compacted test pattern sets

Circuits	#PIs	XID			DTC-XID			Propose(1)			Propose(2)		
		#X-vari	#X-min	%X-ave	#X-vari	#X-min	%X-ave	#X-vari	#X-min	%X-ave	#X-vari	#X-min	%X-ave
s13207	700	1790	1	88.4	1402	6	91.8	1331	8	**92.2**	1335	7	**92.2**
s15850	611	484	5	77.8	429	6	80.6	378	8	**81.5**	380	8	81.2
s35932	1763	7	0	56.2	4	0	58.6	3	0	**59.4**	3	0	**59.4**
s38417	1664	570	11	81.5	510	18	82.7	470	20	**83.1**	473	20	83
s38584	1462	566	2	82.8	509	0	82.9	434	6	**83.6**	433	6	**83.6**
b14	277	21521	1	**76.0**	22383	0	73.9	21286	1	74.4	21248	1	74.4
b15	485	4593	2	85.0	5760	2	82.5	4461	4	**85.5**	4491	4	**85.5**
b20	522	18441	18	**72.8**	18204	19	72	17000	23	72.4	16998	23	72.5
b21	522	20503	19	**72.6**	20279	19	70.9	19281	26	71.5	19326	26	71.6
b22	767	19354	18	**71.6**	18755	30	70.5	17781	30	70.8	17729	30	70.8

Table VII. Test compaction results of uncompacted test pattern sets

Circuits	#tests	#compacted tests			
		XID	DTC-XID	Propose(1)	Propose(2)
s13207	587	281	271	266	**265**
s15850	566	167	162	**145**	146
s35932	81	**38**	64	39	39
s38417	1178	191	155	**136**	145
s38584	878	206	192	**168**	172
b14	1303	908	907	**876**	879
b15	895	524	549	**492**	493
b20	2155	1186	1202	1101	**1088**
b21	2316	1275	1291	1202	**1199**
b22	2759	1068	1075	**972**	980

Table VIII. Test compaction results of compacted test pattern sets

Circuits	#tests	#compacted tests			
		XID	DTC-XID	Propose(1)	Propose(2)
s13207	276	272	267	**263**	266
s15850	130	122	126	**120**	**120**
s35932	22	**22**	**22**	**22**	**22**
s38417	133	130	133	127	**126**
s38584	151	149	151	**137**	138
b14	714	705	704	707	**703**
b15	426	**344**	367	354	354
b20	744	723	723	**716**	718
b21	755	731	735	**723**	724
b22	636	628	635	628	**627**

SR-Quasi-Equivalents: Yet Another Approach to Secure and Testable Scan Design

Katsuya Fujiwara[1], Hideo Fujiwara[2], and Hideo Tamamoto[1]

[1] Faculty of Engineering and Resource Science
Akita University
Akita, 010-8502, JAPAN
{fujiwara, tamamoto}@ie.akita-u.ac.jp

[2] Faculty of Informatics
Osaka Gakuin University
Osaka, 564-8511, JAPAN
fujiwara@ogu.ac.jp

***Abstract*—Scan design makes digital circuits easily testable, however, it can also be exploited to be used for hacking the chip. We have reported a secure and testable scan design approach by using extended shift registers called "*SR-equivalents*" that are functionally equivalent but not structurally equivalent to shift registers [14-17]. In this paper, to further extend the class of SR-equivalents we introduce a *wider* class of circuits called "*SR-quasi-equivalents*" which still satisfy the testability and security similar to SR-equivalents. To estimate the security level, we clarify the cardinality of each equivalent class in SR-quasi-equivalents for several linear structural circuits, and also present the actual number of SR-quasi-equivalents obtained by the enhanced program SREEP.**

Keywords – design-for-testability; scan design; shift register equivalents; shift register quasi-equivalents; security; scan-based side-channel attack.

1. INTRODUCTION

Both testability and security of a chip has become primordial to ensure its reliability and protection from invasion to access important information. However, both may have conflicting requirements for designers. To guarantee quality, designers use design for testability (DFT) methods to make digital circuits easily testable for faults. Scan design is a powerful DFT technique that warrants high controllability and observability over a chip and yields high fault coverage [1], [2]. However, this also accommodates reverse engineering, which contradicts security. For secure chip designers, there is a demand to protect secret data from side-channel attacks and other hacking schemes [3]. Nevertheless, with improved control and access to the chip through DFT, the chip becomes more vulnerable to attacks. Scan chains can be used to steal important information such as intellectual property (IP) and secret keys of cryptographic chips [4], [6]. Despite all these, security chips can be made more susceptible to errors, and thus, not secure, if they are faulty. Therefore, testability is as important as security for secure IC designers to guarantee the quality of security and functionality of the chip. Hence, there is a need for an efficient solution to satisfy both testability and security of digital circuits.

To solve this challenging problem, different approaches have been proposed [4-13]. All the approaches except [13] add extra hardware outside of the scan chain. Disadvantages of this are high area overhead, timing overhead or performance degradation, increased complexity of testing, and limited security for the registers part among others. In [14 - 17], We have reported a secure and testable scan design approach by using extended shift registers called "*SR-equivalents*" that are functionally equivalent but not structurally equivalent to shift registers. The proposed approach is only to replace part of the original scan chains to SR-equivalents, which satisfy both testability and security of digital circuits. This method requires very little area overhead and no performance overhead. Moreover, no additional keys and controller circuits outside of the scan chain are needed, thus making the scheme low-cost and efficient.

In this paper, to further extend the class of SR-equivalents we introduce a *wider* class of circuits called "*SR-quasi-equivalents*" which still satisfy the testability and security similar to SR-equivalents. The security level of the secure scan architecture based on those SR-quasi-equivalents is determined by the probability that an attacker can identify the configuration of the SR-quasi-equivalent used in the circuit, and hence the attack probability approximates to the reciprocal of the cardinality of the class of SR-quasi-equivalents. We clarify the cardinality of each equivalent class in SR-quasi-equivalents for several linear structured circuits, and also present the actual number of SR-quasi-equivalents obtained by the program SREEP [18].

2. SR-EQUIVALENT CIRCUITS

Consider a k-stage shift register shown in Figure 1. For the k-stage shift register, the input value applied to x appears at z after k clock cycles. Suppose a circuit C with a single input x, a single output z, and k flip-flops as shown in Figure 2. If the input value applied to x of C appears at the output z of C after k clock cycles, the circuit C behaves as if it is a k-stage shift register.

A circuit C with a single input x, a single output z, and k flip-flops is called ***functionally equivalent*** to a k-stage shift register (or ***SR-equivalent***) if the input value applied to x at any time t appears at z after k clock cycles, i.e., z(t+k) = x(t) for any time t.

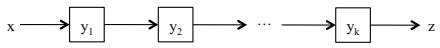


Figure 1. k-stage shift register SR

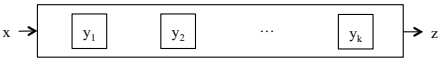
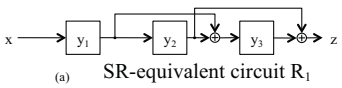


Figure 2. k-stage SR-equivalent circuit C

Figure 3 illustrates an example of 3-stage SR-equivalent circuit R_1. The table in Figure 3 can be obtained easily by symbolic simulation. As shown in the table, z(t+3)=x(t), i.e., the input value applied to x appears at z after k=3 clock cycles, and hence the circuit is SR-equivalent. Although the

input/output behavior of R_1 is the same as that of the 3-stage shift register, the internal state behavior of R_1 is different from the shift register. For the shift register SR, the input sequence (x(t), x(t+1), x(t+2)) which transfers SR to the state (y_1(t+3), y_2(t+3), y_3(t+3)) is (x(t), x(t+1), x(t+2)) = (y_3(t+3), y_2(t+3), y_1(t+3)). The initial state (y_1(t), y_2(t), y_3(t)) can be identified as (y_1(t), y_2(t), y_3(t)) = (z(t+2), z(t+1), z(t)) from the output sequence (z(t), z(t+1), z(t+2)). However, for the SR-equivalent circuit R_1, the input sequence which transfers R_1 to the state (y_1(t+3), y_2(t+3), y_3(t+3)) is (x(t), x(t+1), x(t+2)) = (y_3(t+3)⊕y_2(t+3), y_2(t+3), y_1(t+3)) from Figure 3, and the initial state (y_1(t), y_2(t), y_3(t)) can be identified as (y_1(t), y_2(t), y_3(t)) = (z(t+2), z(t+1), z(t)⊕z(t+1)) from the output sequence. Therefore, without the information on the structure of R_1 one cannot control/observe the internal state of R_1. From this observation, replacing the shift register with an SR-equivalent circuit makes the scan circuit *secure*.

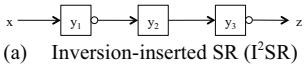


(a) SR-equivalent circuit R_1

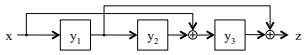
x	y_1	y_2	y_3	z
x(t)	y_1(t)	y_2(t)	y_3(t)	z(t) = y_2(t)⊕y_3(t)
x(t+1)	x(t)	y_1(t)	y_1(t)⊕y_2(t)	z(t+1) = y_2(t)
x(t+2)	x(t+1)	x(t)	x(t)⊕y_1(t)	z(t+2) = y_1(t)
x(t+3)	x(t+2)	x(t+1)	x(t+1)⊕x(t)	z(t+3) = x(t)

(b) Behavior of R_1 by symbolic simulation

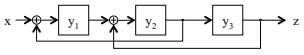
Figure 3. Example of SR-equivalent circuit



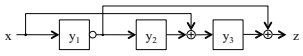
(a) Inversion-inserted SR (I^2SR)



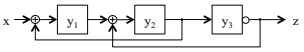
(b) Linear feed-forward SR (LF^2SR)



(c) Linear feedback SR (LFSR)



(d) Inversion-inserted linear feed-forward SR (I^2LF^2SR)



(e) Inversion-inserted linear feedback SR (I^2LFSR)

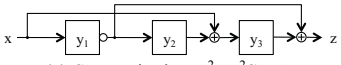
Figure 4. Five types of linear structured circuits

The SR-equivalent circuit shown in Figure 3 is a linear feed-forward shift register. SR-equivalent circuits can also be realized by a linear feedback shift register and/or by inserting inverters as shown in Figure 4. SR-equivalent circuits can be realized not only by linear feed-forward/feedback shift registers with/without inverters but also by more general circuits.

In [15], we showed the number of k-stage SR-equivalent circuits for each type of circuits. They are summarized in Table I. From those cardinalities of SR-equivalents, the complexity or the difficulty of identifying the structure of SR-equivalent circuits increases more than exponentially as the stage of SR increases. Hence, very high security can be realized by using SR-equivalent circuits.

TABLE 1. CARDINALITY OF EACH CLASS

	# of circuits in the class	# of SR-equivalents in the class
I^2SR	2^{k+1} - 1	2^k-1
LF^2SR	$2^{k(k+1)/2}$ - 1	$2^{k(k-1)/2}$ - 1
LFSR		
I^2LF^2SR	$(2^{k(k+1)/2}-1)(2^{k+1}-1)$	$(2^{k(k-1)/2}-1)(2^k-1)$
I^2LFSR		



(a) SR-equivalent I^2LF^2SR, R_2

x	y_1	y_2	y_3	z
x(t)	y_1(t)	y_2(t)	y_3(t)	z(t)=1⊕y_1(t)⊕y_3(t)
x(t+1)	x(t)	1⊕y_1(t)	x(t)⊕y_2(t)	z(t+1)=1⊕y_2(t)
x(t+2)	x(t+1)	1⊕x(t)	x(t+1)⊕1⊕y_1(t)	z(t+2)=y_1(t)
x(t+3)	x(t+2) =y_1(t+3)	1⊕x(t+1) =y_2(t+3)	x(t+2)⊕1⊕x(t) =y_3(t+3)	z(t+3)=x(t)

x(t) = 1⊕y_1(t+3)⊕y_3(t+3)
x(t+1) = 1⊕y_2(t+3)
x(t+2) = y_1(t+3)

(b) Equations for state-justification

x	y_1	y_2	y_3	z
x(t)	y_1(t)	y_2(t)	y_3(t)	z(t)=1⊕y_1(t)⊕y_3(t)
x(t+1)	x(t)	1⊕y_1(t)	x(t)⊕y_2(t)	z(t+1)=1⊕y_2(t)
x(t+2)	x(t+1)	1⊕x(t)	x(t+1)⊕1⊕y_1(t)	z(t+2)=y_1(t)
x(t+3)	x(t+2)	1⊕x(t+1)	x(t+2)⊕1⊕x(t)	z(t+3)=x(t)

y_1(t) = z(t+2)
y_2(t) = 1⊕z(t+1)
y_3(t) = 1⊕z(t)⊕z(t+2)

(c) Equations for state-observation

Figure 5. State-justification and state-observation for R_2

3. SR-QUASI-EQUIVALENT CIRCUITS

For an SR-equivalent circuit, the following two problems are important in order to utilize the SR-equivalent circuit as a

scan shift register in testing. One problem is to generate an input sequence to transfer the circuit into a given desired state. This is called ***state-justification problem***. The other problem is to determine the initial state by observing the output sequence from the state. This is called ***state-observation problem.***

A circuit C with a single input, a single output, and k flip-flops is called to be ***scan-controllable*** if for any internal state of C a transfer sequence (of length k) to the state (final state) can be generated only from the connection information of C, independently of the initial state.

A circuit C with a single input, a single output, and k flip-flops is called to be ***scan-observable*** if any present state (initial state) of C can be identified only from the output sequence (of length k) and the connection information of C, independently of the initial state and the input sequence.

A circuit C is called to be ***scan-testable*** if C is *scan-controllable* and *scan-observable*.

Any SR-equivalent circuit is scan-controllable and scan-observable, and hence scan-testable.

Consider a 3-stage I^2LF^2SR, R_2, given in Figure 5(a). This I^2LF^2SR is SR-equivalent. By using symbolic simulation, we can derive equations to obtain an input sequence (x(t), x(t+1), x(t+2)) that transfers R_2 from any state to the desired final state ($y_1(t+3)$, $y_2(t+3)$, $y_3(t+3)$) as illustrated in Figure 5(b). Similarly, as illustrated in Figure 5(c), we can derive equations to determine uniquely the initial state ($y_1(t)$, $y_2(t)$, $y_3(t)$) from the output sequence. Hence, R_2 is scan-testable.

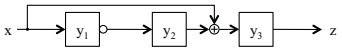
Next, let us first try to relax the definition of scan-testability. First, suppose to relax the scan-controllability by removing "independence of the initial state" as follows. A circuit C is called to be ***quasi-scan-controllable*** if for any internal state of C a transfer sequence of length k to the final state can be generated from a given initial state and the connection information of C. However, this quasi-scan-controllability does not make the state-justification easy because of the dependence of initial state. So, we don't adopt this relaxation. Next, let us try to relax the definition of scan-observability as follows. A circuit C is called to be ***quasi-scan-observable*** if any present state (initial state) of C can be identified from the *input-output sequence* (of length k) and the connection information of C. In this case, since it is easy to apply any input sequence to C, this quasi-scan-observability makes state-observation easy. So, we adopt this relaxation and extend scan-testability as follows. A circuit C is called to be ***quasi-scan-testable*** if C is *scan-controllable* and *quasi-scan-observable*.

Based on the above new concept of "quasi-scan-testability," we introduce a new class of circuits as follows.

A circuit C with a single input x, a single output z, and k flip-flops is called ***functionally quasi-equivalent*** to a k-stage shift register (or ***SR-quasi-equivalent***) if the input value applied to x at any time t appears at z after k clock cycles with exclusive-OR of some inputs and/or constant 1, i.e.,

$$z(t+k) = x(t) \oplus c_0 \oplus c_1x(t+1) \oplus c_2x(t+2) \oplus \ldots \oplus c_kx(t+k)$$

where c_0 , c_1, c_2, … , c_k are 0 or 1. The ordered set of coefficients (c_0 , c_1, c_2, … , c_k) is called the ***characteristic coefficient*** of the SR-quasi-equivalent circuit C.



(a) SR-quasi-equivalent I^2LF^2SR, R_3

x	y_1	y_2	y_3	z
$x(t)$	$y_1(t)$	$y_2(t)$	$y_3(t)$	$z(t)=y_3(t)$
$x(t+1)$	$x(t)$	$1\oplus y_1(t)$	$x(t)\oplus y_2(t)$	$z(t+1)=x(t)\oplus y_2(t)$
$x(t+2)$	$x(t+1)$	$1\oplus x(t)$	$x(t+1)\oplus 1\oplus y_1(t)$	$z(t+2)=x(t+1)\oplus 1\oplus y_1(t)$
$x(t+3)$	$x(t+2)$	$1\oplus x(t+1)$	$x(t+2)\oplus 1\oplus x(t)$	$z(t+3)= x(t+2)\oplus 1\oplus x(t)$

(b) Symbolic simulation

x	y_1	y_2	y_3	z
$x(t)$	$y_1(t)$	$y_2(t)$	$y_3(t)$	$z(t)=y_3(t)$
$x(t+1)$	$x(t)$	$1\oplus y_1(t)$	$x(t)\oplus y_2(t)$	$z(t+1)=x(t)\oplus y_2(t)$
$x(t+2)$	$x(t+1)$	$1\oplus x(t)$	$x(t+1)\oplus 1\oplus y_1(t)$	$z(t+2)=x(t+1)\oplus 1\oplus y_1(t)$
$x(t+3)$	$x(t+2)$ $=y_1(t+3)$	$1\oplus x(t+1)$ $=y_2(t+3)$	$x(t+2)\oplus 1\oplus x(t)$ $=y_3(t+3)$	$z(t+3)= x(t+2)\oplus 1\oplus x(t)$

$x(t) = 1\oplus y_1(t+3)\oplus y_3(t+3)$
$x(t+1) = 1\oplus y_2(t+3)$
$x(t+2) = y_1(t+3)$

(c) Equations for state-justification

x	y_1	y_2	y_3	z
$x(t)$	$y_1(t)$	$y_2(t)$	$y_3(t)$	$z(t)=y_3(t)$
$x(t+1)$	$x(t)$	$1\oplus y_1(t)$	$x(t)\oplus y_2(t)$	$z(t+1)=x(t)\oplus y_2(t)$
$x(t+2)$	$x(t+1)$	$1\oplus x(t)$	$x(t+1)\oplus 1\oplus y_1(t)$	$z(t+2)=x(t+1)\oplus 1\oplus y_1(t)$
$x(t+3)$	$x(t+2)$	$1\oplus x(t+1)$	$x(t+2)\oplus 1\oplus x(t)$	$z(t+3)= x(t+2)\oplus 1\oplus x(t)$

$y_1(t) = z(t+2) \oplus x(t+1)\oplus 1$
$y_2(t) = z(t+1)\oplus x(t)$
$y_3(t) = z(t)$

(d) Equations for state-observation

Figure 6. Example of SR-quasi-equivalent circuit

We can easily see that any SR-quasi-equivalent circuit C satisfies the following two properties: (1) for any internal state of C a transfer sequence (of length k) to the state (final state)

can be generated only from the connection information of C, independently of the initial state, i.e., C is scan-controllable; (2) any present state (initial state) of C can be identified from the input-output sequence (of length k) and the connection information of C, i.e., C is quasi-scan-observable, where k is the number of flip-flops. Hence, we have the following.

Any SR-quasi-equivalent circuit is scan-controllable and quasi-scan-observable, and hence quasi-scan-testable.

Consider a 3-stage I^2LF^2SR, R_3, given in Figure 6(a). This I^2LF^2SR is SR-quasi-equivalent. By using symbolic simulation, we can obtain an output sequence (z(t), z(t+1), z(t+2), z(t+3)) and the output $z(t+3)=x(t) \oplus 1 \oplus x(t+2)$ as shown in Figure 6(b). Therefore, R_3 is SR-quasi-equivalent. By using symbolic simulation, we can derive equations to obtain an input sequence (x(t), x(t+1), x(t+2)) that transfers R_3 from any state to the desired final state ($y_1(t+3)$, $y_2(t+3)$, $y_3(t+3)$) as illustrated in Figure 6(c). Similarly, as illustrated in Figure 6(d), we can derive equations to determine uniquely the initial state ($y_1(t)$, $y_2(t)$, $y_3(t)$) from the input/output sequence.

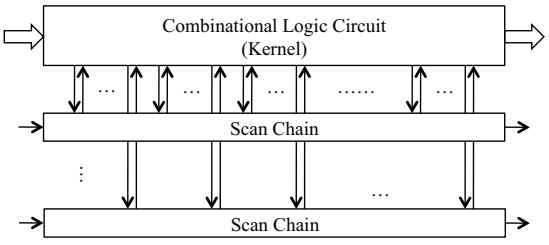
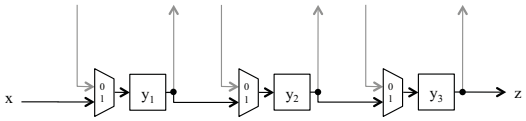
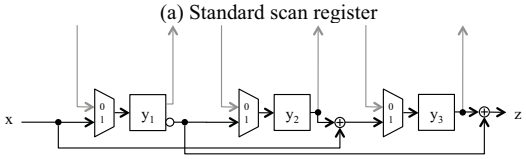


Figure 7. Scan-designed circuit



(a) Standard scan register



(b) Modified scan register (SR-quasi-equivalent)

Figure 8. Standard and modified scan registers

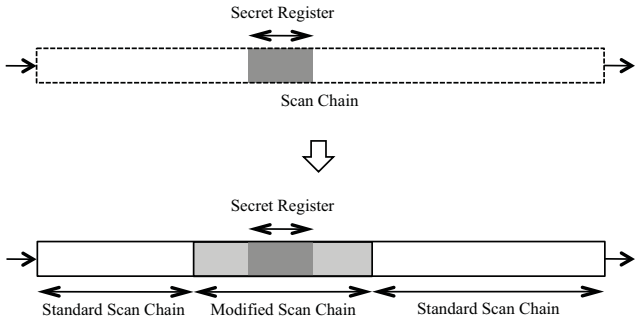


Figure 9. Replacement of scan chain by modified scan chain

4. APPLICATION TO SCAN DESIGN

A scan-designed circuit consists of a single or multiple scan chains and the remaining combinational logic circuit (*kernel*) as illustrated in Figure 7. A scan chain is regarded as a circuit consisting of a shift register with multiplexers that select the normal data from the combinational logic circuit and the shifting data from the preceding flip-flop as shown in Figure 8(a). Here, we replace the shift register with a modified SR-quasi-equivalent scan register as shown in Figure 8(b).

However, to reduce the area overhead as much as possible, not all scan chains are replaced with modified scan chains. As shown in Figure 9, only parts of scan chains necessary to be secure are replaced with modified SR-quasi-equivalent scan chains that cover secret registers to be protected, and the size of the modified scan chains is large enough to make it secure. The size of modified scan chain can be determined by the expected security level computed from the cardinality of SR-quasi-equivalent circuits that will be described in the following section. The delay overhead due to additional Exclusive-OR gates influences only scan operation, and hence there is no delay overhead for normal operation.

5. CARDINALITY OF SR-QUASI-EQUIVALENTS

When we consider a secure scan design, we need to assume what the attacker knows and how he can potentially make the attack. Here, we assume that the attacker does not know the detailed information in the gate-level design, and that the attacker knows the presence of test pins (scan in/out, scan, and reset) and modified scan chains. However, he does not know the structure of modified scan chains (the connection information, position of XOR and NOT, and the size).

Based on the above assumption, we consider the security to prevent scan-based attacks.

A circuit C with a single input x, a single output z, and k flip-flops is called ***scan-secure*** if the attacker cannot determine the structure of C.

Consider two SR-quasi-equivalent circuits C_1 and C_2. We can easily see that *C_1 and C_2 have the same characteristic coefficient if and only if they are functionally equivalent.* Suppose that C_1 and C_2 have different structures but the same characteristic coefficient. Then, we cannot distinguish C_1 and C_2 merely from the input/output relation because they are functionally equivalent. Therefore, C_1 and C_2 are scan-secure.

The characteristic coefficient of any SR-quasi-equivalent circuit C can be identified by applying input sequences to C and observing the output responses from C (though it might be time-consuming and the complexity increases exponentially in the worst case).

Here, we partition the whole set of SR-quasi-equivalent circuits with k flip-flops into equivalent classes based on characteristic coefficient. Since the size of coefficient is k+1, the number of equivalent classes is 2^{k+1}. The first equivalent class of the characteristic coefficient, 00…0, is the set of SR-equivalent circuits.

TABLE II. CARDINALITY OF EACH EQUIVALENT CLASS IN SR-QUASI-EQUIVALENTS OBTAINED BY ANALYSIS

EQUIVALENT CLASS	I^2SR	LF^2SR	I^2LF^2SR	I^2LFSR	LFSR	TOTAL
00…00	2^K-1	$2^{K(K-1)/2}-1$	$(2^{K(K-1)/2}-1)(2^K-1)$	$(2^{K(K-1)/2}-1)(2^K-1)$	$2^{K(K-1)/2}-1$	$2(2^{K(K+1)/2})-2^K-1$
00…01 ~ 01…11	0 ~ 0	$2^{K(K-1)/2}$ ~ $2^{K(K-1)/2}$	$2^{K(K-1)/2}(2^K-1)$ ~ $2^{K(K-1)/2}(2^K-1)$	0 ~ 0	0 ~ 0	$2^{K(K+1)/2}$ ~ $2^{K(K+1)/2}$
10…00	2^K	0	$(2^{K(K-1)/2}-1)2^K$	$(2^{K(K-1)/2}-1)2^K$	0	$2(2^{K(K+1)/2})-2^K$
10…01 ~ 11…11	0 ~ 0	0 ~ 0	$2^{K(K-1)/2}2^K$ ~ $2^{K(K-1)/2}2^K$	0 ~ 0	0 ~ 0	$2^{K(K+1)/2}$ ~ $2^{K(K+1)/2}$
TOTAL	$2^{K+1}-1$	$2^{K(K+1)/2}-1$	$(2^{K(K+1)/2}-1)(2^{K+1}-1)$	$(2^{K(K-1)/2}-1)(2^{K+1}-1)$	$2^{K(K-1)/2}-1$	

The security level of the secure scan architecture based on those SR-quasi-equivalents is determined by the probability that an attacker can identify the structure of the SR-quasi-equivalent circuit used in the circuit, and hence the attack probability approximates to the reciprocal of the cardinality of the class of SR-quasi-equivalents. Since the attacker can identify the characteristic coefficient of SR-quasi-equivalents, we need to clarify the cardinality of each equivalent class in SR-quasi-equivalents to estimate the attack probability.

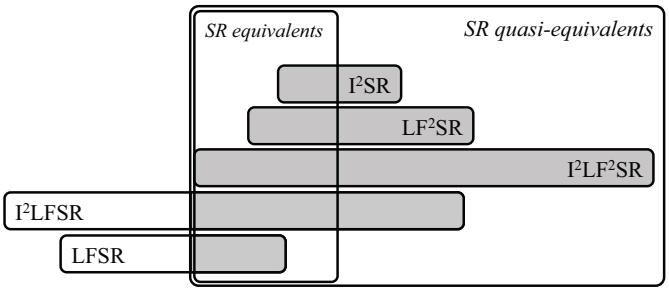


Figure 10. Covering relation among classes

The cardinality of each equivalent class in five types of linear structured circuits (I^2SR, LF^2SR, I^2LF^2SR, I^2LFSR, LFSR) is summarized in Table II. The second row is the equivalent class of the characteristic coefficient 00…00, and this is the same as the SR-equivalents (see Table I). The fourth row is the equivalent class of 10…00 such that $z(t+k) = x(t) \oplus 1$. The last row is the total number of each type of linear structured circuit. They coincide with the total number of circuits in the class for I^2SR, LF^2SR, and I^2LF^2SR (see Table I). This means any circuit of type I^2SR, LF^2SR, and I^2LF^2SR is SR-quasi-equivalent. On the other hand, as for I^2LFSR only two equivalent classes (00…00 and 10…00) are SR-quasi-equivalents. As for LFSR, there is no SR-quasi-equivalent circuit except SR-equivalent circuits.

In [15, 16], we reported a program called SREEP (Shift Register Equivalents Enumeration and Synthesis Program).

TABLE III. CARDINALITY OF EACH EQUIVALENT CLASS FOR K=4 OBTAINED BY SREEP

	I^2SR	LF^2SR	I^2LF^2SR	I^2LFSR	LFSR	TOTAL
00000	15	63	945	945	63	2,031
00001 ~ 01111	0 ~ 0	64 ~ 64	960 ~ 960	0 ~ 0	0 ~ 0	1,024 ~ 1,024
10000	16	0	1,008	1,008	0	2,032
10001 ~ 11111	0 ~ 0	0 ~ 0	1,024 ~ 1,024	0 ~ 0	0 ~ 0	1,024 ~ 1,024
TOTAL	31	1023	31,713	1,953	63	

To examine the actual cardinalities of equivalent classes in SR-quasi-equivalents, we enhanced the program by adding several facilities in handling SR-quasi-equivalents and its equivalent classes. Table III shows the results obtained by SREEP. The theoretical values obtained by substituting 4 for k for Table II coincides with the actual values in Table III obtained by SREEP [18].

The probability that an attacker can identify the structure of an SR-quasi-equivalent circuit in an equivalent class approximates to the reciprocal of the cardinality of the class. In Table II, they are $O((2^k)^2)$ except $O(2^k)$ for I^2SR. Hence, the number of indistinguishable SR-quasi-equivalent circuits grows much more rapidly than exponentially and hence they are very secure.

From Tables I and II, for each class of linear structured circuits (I^2SR, LF^2SR, I^2LF^2SR, I^2LFSR, LFSR), we have Table IV which illustrates the total number of circuits in the

class, the number of SR-equivalents in the class, and the number of SR-quasi-equivalents in the class.

From Tables II and IV, we have the covering relation among five classes of linear structured circuits (I^2SR, LF^2SR, I^2LF^2SR, I^2LFSR, LFSR), and SR-equivalents and SR-quasi-equivalents as illustrated in Figure 10.

From Figure 10, we can see all the circuits in I^2SR, LF^2SR, and I^2LF^2SR are SR-quasi-equivalent, and hence we can use any of them to organize the secure and testable scan chains, which means it is very easy to design an SR-quasi-equivalent circuit.

TABLE IV. CARDINALITY OF EACH CLASS OF SR-EQUIVALENTS/QUASI-EQUIVALENTS

CLASS	# OF CIRCUITS IN THE CLASS	# OF SR-EQUIVALENTS IN THE CLASS	# OF SR-QUASI-EQUIVALENTS IN THE CLASS
I^2SR	$2^{K+1}-1$	$2^{K}-1$	$2^{K+1}-1$
LF^2SR	$2^{K(K+1)/2}-1$	$2^{K(K-1)/2}-1$	$2^{K(K+1)/2}-1$
I^2LF^2SR	$(2^{K(K+1)/2}-1)(2^{K+1}-1)$	$(2^{K(K-1)/2}-1)(2^{K}-1)$	$(2^{K(K+1)/2}-1)(2^{K+1}-1)$
I^2LFSR	$(2^{K(K+1)/2}-1)(2^{K+1}-1)$	$(2^{K(K-1)/2}-1)(2^{K}-1)$	$(2^{K(K-1)/2}-1)(2^{K+1}-1)$
LFSR	$2^{K(K+1)/2}-1$	$2^{K(K-1)/2}-1$	$2^{K(K-1)/2}-1$

6. CONCLUSION

In our previous work [14-17], we reported a secure and testable scan design approach by using extended shift registers called "SR-equivalents" that are functionally equivalent but not structurally equivalent to shift registers. In this paper, to extend the class of SR-equivalents we have introduced a wider class of circuits called "SR-quasi-equivalents" which still satisfy the testability and security similar to SR-equivalents.

The security level for the secure scan design based on SR-quasi-equivalents is related to the attack probability that approximates to the reciprocal of the cardinality of the class of SR-quasi-equivalents. In this paper, we clarified the cardinality of each equivalent class in SR-quasi-equivalents for several linear structured circuits, and also presented the actual number of SR-quasi-equivalents obtained by the program SREEP [18].

REFERENCES

[1] H. Fujiwara, Y. Nagao, T. Sasao, and K. Kinoshita, "Easily testable sequential machines with extra inputs," *IEEE Trans. on Computers,* Vol. C-24, No. 8, pp. 821-826, Aug. 1975.

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

[3] K. Hafner, H. Ritter, T. Schwair, S. Wallstab, M. Deppermann, J. Gessner, S. Koesters, W. Moeller, and G. Sandweg, "Design and test of an integrated cryptochip," *IEEE Design and Test of Computers*, pp. 6-17, Dec. 1999.

[4] D. Hely, M.-L. Flottes, F. Bancel, B. Rouzeyre, and N. Berard. "Scan design and secure chip." *10th IEEE International On-Line Testing Symposium,* pp. 219–224, 2004.

[5] D. Hely, F. Bancel, M.L. Flottes, and B. Rouzeyre. "Securing scan control in crypto chips." *Journal of Electronic Testing - Theory and Applications*, Vol. 23, No. 5, pp. 457–464, Oct. 2007.

[6] B. Yang, K. Wu, and R. Karri. "Scan based side channel attack on dedicated hardware implementations of data encryptionstandard." *International Test Conference 2004*, pp. 339–344, 2004.

[7] B. Yang, K. Wu, and R. Karri. "Secure scan: A design-for-test architecture for crypto chips." *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 25, No.10, pp. 2287–2293, Oct. 2006.

[8] J. Lee, M. Tehranipoor, and J. Plusquellic, "A low-cost solution for protecting IPs against scan-based side-channel attacks," *24th IEEE VLSI Test Symposium,* pp. 94 - 99, 2006.

[9] J. Lee, M. Tehranipoor, C. Patel, and J. Plusquellic. "Securing designs against scan-based side-channel attacks." *IEEE Trans. on Dependable and Secure Computing*, Vol. 4, No. 4, pp. 325–336, Oct.-Dec. 2007.

[10] S. Paul, R. S. Chakraborty, and S. Bhunia. "VIm-Scan: A low overhead scan design approach for protection of secret key inscan-based secure chips." *25th IEEE VLSI Test Symposium*, pp. 455–460, 2007.

[11] M. Inoue, T. Yoneda, M. Hasegawa, and H. Fujiwara, "Partial scan approach for secret information protection," *14th IEEE European Test Symposium*, pp. 143-148, May 2009.

[12] U. Chandran and D. Zhao, "SS-KTC: A high-testability low-overhead scan architecture with multi-level security integration," *27th IEEE VLSI Test Symposium,* pp. 321-326, May 2009.

[13] G. Sengar, D. Mukhopadhyay, and D. R. Chowdhury. "Secured flipped scan-chain model for crypto-architecture." *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vo. 26, No.11, pp. 2080-2084, November 2007.

[14] H. Fujiwara and M.E.J. Obien, "Secure and testable scan design using extended de Brujin graph," *15th Asia and South Pacific Design Automation Conference*, pp. 413-418, Jan. 2010.

[15] K. Fujiwara, H. Fujiwara, M.E.J. Obien, and H. Tamamoto, "SREEP: Shift register equivalents enumeration and synthesis program for secure scan design," *13th IEEE International Symposium on Design and Diagnosis of Electronic Circuits and Systems,* pp. 193-196, April 2010.

[16] K. Fujiwara, H. Fujiwara, and H. Tamamoto, "SREEP-2: SR-equivalent Generator for Secure and Testable Scan Design," 11th IEEE Workshop on RTL and High Level Testing, pp. 7-12, Dec. 2010.

[17] H. Fujiwara, K. Fujiwara, and H. Tamamoto, "Secure Scan Design Using Shift Register Equivalents against Differential Behavior Attack," *16th Asia and South Pacific Design Automation Conference*, pp.818-823, Jan. 2011.

[18] SREEP: http://sreep.fujiwaralab.net/.

Gracefully Degradable 3D On-Chip Networks Using an Optimized Re-routing Mechanism

Ali Shahabi, Reza Nakhjavani, Saeed Safari and Zainalabedin Navabi
Electrical and Computer Engineering, School of Engineering Colleges, Campus 2
University of Tehran, 1450 North Amirabad
Tehran, Iran
{shahabi, reza.nakhjavani, saeed, navabi}@ut.ac.ir

Abstract— **The decreasing manufacturing yield of integrated circuits, as a result of rising complexity and deep submicron feature size, and the emergence of 3D on-chip network methodology architectures, has necessitated the search for gracefully degradable techniques in order to make such faulty chips reusable. In this paper, a novel deterministic and distributed routing mechanism is presented. First, we introduce an efficient and scalable router for network switches using a fixed size programmable routing table (PRT). Then a heuristic search algorithm is proposed to find a valid configuration for PRT to compensate the effect of faulty links in three dimensional on-chip networks. Several techniques exploited to accelerate the speed of the algorithm are also demonstrated. Experimental results show that, the proposed mechanism is not only able to gracefully degrade almost 90% of physically connected faulty networks, but it also considerably reduces the required search effort as compared to the exhaustive search method. However, the general trend of performance parameters is to get worse when the number of faulty links increases.**

Keywords- Graceful Degradation;Faulty Links; 3D-NoCs; 3D Mesh-based Routing (3DMR); Routing Connectivity; Constraints;

I. INTRODUCTION

Deep submicron feature size in the state of the art fabrication process [1] and the imposed complexity of the modern VLSI circuits [2] have led us to integrate pre-designed and pre-verified Intellectual Property (IP) cores into a single chip, called System-on-Chip (SoC) [3]. To solve performance and power limitations caused by long interconnects in such systems, proposed Network-on-Chip (NoC) architecture are becoming a promising solution [2], [4]. Although, NoC-based approaches provide higher communication bandwidth (and hence higher performance) than the conventional data transfer techniques, a careful mapping method is mandatory to meet the performance requirements. Due to the limited floor-planning options, physical mapping of a large number of cores and also several memory blocks into a single chip using two-dimensional (2D) NoC architectures is very difficult if not impossible [5].

Combination of three-dimensional (3D) ICs which consist of several layers of active devices [6] with NoC design methodology [2], not only opens a wide range of possibilities to obtain more performance improvements, but also is leading to lower power consumption and higher noise immunity [7]. Since three-dimensional NoC (3D-NoC) structure provides both the higher bandwidth and greater number of neighbors for each functional block, it makes the capability of performance improvement possible even without device scaling [6].

Despite the fact that technology scaling enables us to shorten NoC interconnection network, it makes the products more susceptible to the permanent manufacturing faults due to the stochastic variations of the physical design parameters [1]. Since most manufacturing faults affect just a limited area of the chip, it well worth to find new methods to reuse such faulty chips. Graceful degradability which is one of the most popular approaches in this domain, reconfigures the faulty IC in such a way to prevent using the faulty modules and get the work done using the remaining non-faulty ones [8]. This approach usually implies a specific flow that begins by developing required algorithms which result in reconfigurable circuits. Then, diagnosis techniques are exploited to detect and locate the faulty components. And finally, based on the obtained fault pattern, a proper configuration, which bypasses the faulty elements, is programmed into the circuit (e.g. using common built-in test provisions like JTAG).

In this paper we focus on a static reconfiguration technique to gracefully degrade a 3D mesh-based NoC in the presence of several faulty links. Using a fixed size programmable routing table in each switch not only makes the switches scalable but also enables us to avoid using faulty links by reprogramming the lookup tables. The rest of this paper is organized as follows. Several methods in fault-tolerant and degradable 3D-NoC domains are briefly reviewed in Section II. In Section III, our routing mechanism is thoroughly illustrated. In Section IV, we will focus more on the improvement and speedup of the proposed algorithm. Section V provides the experimental results and their interpretation, while Section VI concludes the paper and summarizes our proposed approach.

II. RELATED WORKS

While more dies stack vertically in 3D-NoCs, the length of heat conduction path and the power consumption density will both increase, leading to a significant growth of the temperature within the chip [9]. The poor cooling mechanism usually utilized in such systems, increases the probability of fault occurrence and results in vulnerability of the overall system reliability and performance. To make 3D-NoCs fault tolerant or at least degradable, several techniques have been proposed in the literature. In [10], the authors presented an adaptive routing approach which utilizes the remaining operative parts of a faulty switch. Although this online approach degrades the system gracefully for both permanent

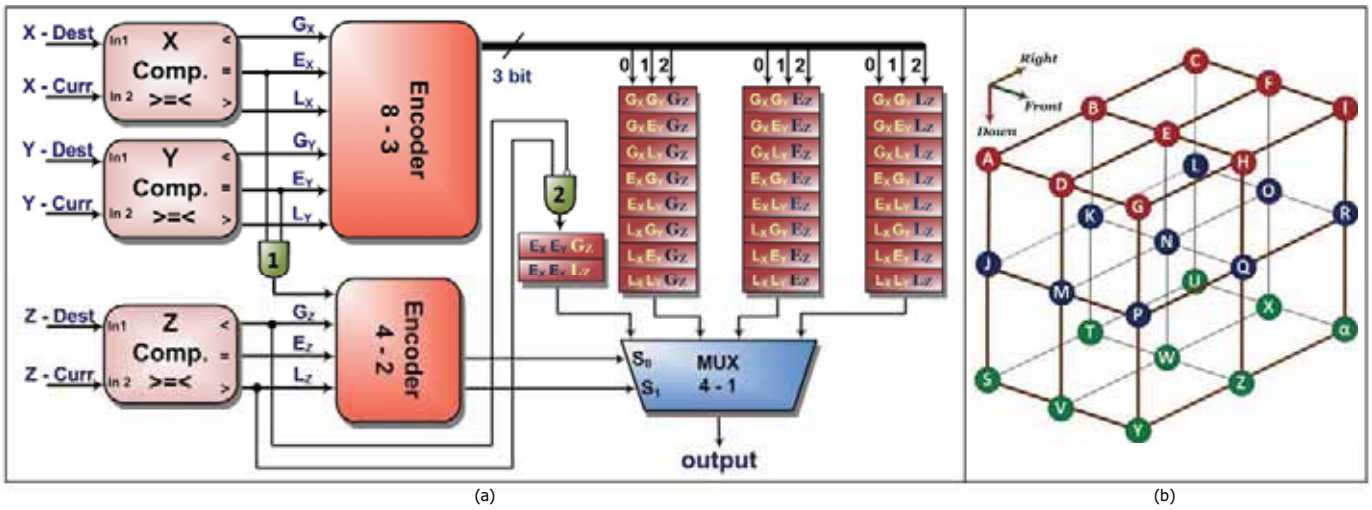


Figure 1. (a) 3D Mesh-based Routing (b) A 3×3×3 3D Mesh-based Network

and transient faults, imposed implementation cost because of the required modification in the router architecture, reduces its applicability.

Three different routing algorithms have been investigated in [11]. It compares 'flooding', 'directed flooding' and 'random walk' routing algorithms in terms of area and power consumption. It has been shown that 'Random walk' algorithm has the same level of fault tolerance as the two others, but it has lower rate of packet retransmission and thus decreases the power consumption. The authors in [12], have proposed a dynamic routing algorithm which manages both static and dynamic permanent failures by incorporating acknowledgment signals and using an stress factor. Complex router architecture and the increased response time are the two most important drawbacks of this technique.

Another offline distributed routing algorithm for 2D-NoCs was introduced in [13]. The algorithm has low area overhead but it just considers local information about the failures and hence somehow it is similar to our work. In [14], a low level defect-tolerance approach for yield improvement was described. The basic idea of this work is utilizing an efficient and effective redundancy by introducing a defect-tolerant vertical link in 3D-NoCs. In spite of great yield improvement, the implementation of this technique needs fabrication process modifications. Mejia et al. [15] have proposed a deterministic segmentation based routing algorithm for 2D-NoC in which the mesh or torus topology is partitioned into independent segments. Although this method gets the advantage of more freedom in defining restrictions in each segment, it ignores global effects of the fault.

III. Our Routing Mechanism

In a typical NoC (either 2D or 3D), switches are responsible to route packets between nodes. Thus each switch should have several ports to communicate with the neighboring switches or its Processing Element (PE). A switch also contains a router to provide a path between the input and output ports. Routing algorithm implementation (to define a path by which a packet reaches to its destination) is one of the crucial aspects of the switch design. The routing algorithms can be classified according to the number of destinations (unicast vs. multicast), routing decision (source vs. distribution), and adaptivity (deterministic vs. adaptive). The details of each method are irrelevant to the scope of this paper and can be found in any text related to the interconnection networks [12]. Simple and efficient hardware implementation of deterministic and distributed routing algorithms for mesh based networks results in their acceptance as the de facto standard for routing in the NoC realm. Hence, in this work, we focus on these algorithms.

A. Proposed Small and Scalable Hardware

As we mentioned in previous works [16], in NoC switches, the routing algorithm can be implemented as a hardwired module or as a Programmable Routing Table (PRT). Although in the former case, no requirement for the post-manufacturing programming is needed, the switch cannot be changed to cope with the physical failures. In the latter case, which adds a PRT programming step to the manufacturing process, the switches can be reconfigured to bypass the faulty elements. A naive way to implement a PRT is to use a lookup table with as many entries as the number of nodes in the network. This approach, called Per-Node Routing (PNR), suffers from several major drawbacks including area overhead, throughput reduction in large networks, and scalability [16].

In 3D mesh based networks, the address of a node can be considered as a triplet (x,y,z) which is the coordinates of the node in 3D-mesh. Each mesh switch, in general, has seven ports: one attached to its local PE (local port) and the others to the neighbors (system ports). When a packet arrives, it should be delivered to the attached PE through the local port, if it is destined for that node. Otherwise, based on the destination address, one of the system ports (*i.e.* Left, Right, Up, Down, Front, and Back) is selected. An extension of the simple conventional 'XY' algorithm, called *'XYZ'*, has been vastly used in 3D mesh topology. *'XYZ'* algorithm, like its ancestor, routes a packet first in the 'X' direction (left or right), then in the 'Y' direction (up or down), and finally in the 'Z' direction (front or back).

Fig. 1(a) illustrates our proposed routing hardware. Three small comparators compare x-, y-, and z-coordinates of the

current node (*CRT*) with those of the destination node (*DST*). The outputs of each comparator assumes three different one-hot coded states (*G* for greater, *E* for equal, and *L* for less). Thus, we have 27 (3^3) different situations for the combination of comparator outputs. The first encoder (8 to 3) gets the outputs of the *X* and *Y* comparators and generates a 3-bit signal indicating which of the 8 different situations based on the *x*- and *y*-coordinates of *CRT* and *DST* has occurred. The output of this encoder will be used to index three same 8-entry, programmable lookup tables. In these three lookup tables, each entry (*i.e.* G_XG_Y, G_XE_Y, G_XL_Y, E_XG_Y, E_XL_Y, L_XG_Y, L_XE_Y, and L_XL_Y) contains a port identifier to indicate which port (from the six system ports) has to be used.

The first AND gate is activated when both *CRT* and *DST* have the same *x*- and *y*-coordinates. The outputs of the *Z* comparator and the first AND gate feed the second encoder (4 to 2) to generate a 2-bit signal that is used by the multiplexer to indicate which of the four lookup tables should be selected. The first lookup table which has just two entries (*i.e.* $E_XE_YG_Z$ and $E_XE_YL_Z$), is selected when the first AND gate is activated. As stated before, this situation happens when *CRT* and *DST* have exactly the same *x*- and *y*-coordinates. Now based on the *z*-coordinate (*i.e.* the output of the second AND gate) one of these two entries is selected. The remaining three lookup tables in this scheme use 8 entries to identify different situations with respect to the *X* and *Y* comparator outputs. Based on G_Z, E_Z, and L_Z (*Z* comparator output) the second, third, or fourth lookup table is chosen, respectively (See the right part of Fig. 1(a)). Apparently, when *x*-, *y*-, and *z*-coordinates of the *CRT* and *DST* become equal, the switch uses its local port.

Unlike PNR-based routers, this routing hardware has a small and fixed structure that does not depend on the number of nodes in the network. Throughout this paper, we refer to this method as 3D Mesh-Based Routing (3DMR).

B. Physical vs. Routing Connectivity

The network topology is considered as a graph with switches being its vertices and links being its edges. If this graph is connected, then the network will be **physically connected**. The network is **routing connected** if for every source (*SRC*) and destination (*DST*) pair of nodes, a packet produced in *SRC* can be routed to reach its *DST*. Whether this is possible or not depends on the routing table of the visited nodes starting from *SRC*. For example improper filled 3DMR (*i.e.* configuration) might cause the packet never reaches its destination. **Valid configuration** term is used throughout this paper to refer to the configurations leading to a routing connected network.

It can be easily proved that every physically connected PNR-based network is routing connectable, that means at least one set of PNR-based PRT configurations exists to make the network routing connected. But it might be possible to have a physically connected 3DMR-based network which is not routing connectable. Consider Fig. 1(b) in which 4 pairs of links (e.g. *ED*, *EF*, *EH* and *EN*) are broken due to the physical faults. No valid configuration can be found for the 3DMR of node *B* because when *B* has a packet destined to *E*, it should send it using *BE* link and when it has a packet destined to *H* it should not use *BE* link. Thus, there is no feasible port identifier for ($E_XE_YG_Z$) entry of *B*'s routing table.

C. Our Heuristic to Fill PRTs

In a 3D-NoC, physical permanent faults might occur in PEs, communication links, and switches. In this work, we focus on degradability just in the presence of faulty communication links. In what follows, we describe an algorithm tries to find a valid configuration (using 3DMR-based PRT) to compensate the effects of link failure(s). One possible method to find a valid configuration is exhaustive search. In this approach we should consider all possible configurations and check to find whether it results in a routing connected network or not. This approach is not feasible even for the small networks, for example in a 3×3×3 network, there are 27 nodes, 26 PRT entries per node, 6 possible system ports per entry, resulting in $6^{27\times26}$ or approximately 1.83×10^{546} different configurations.

In this paper, we propose a heuristic search algorithm to find a valid configuration to make a faulty 3D mesh-based communication network routing connected. This algorithm tries to reduce possible values of PRT entries by using the heuristic of imposing constraints. These constraints are obtained through considering local effect(s) of each failure on the content of the related entries. Since finding a valid configuration for each faulty IC should be done during IC testing process, our emphasis is on the speed of the algorithm not the optimality of the solution. To the best of our knowledge, no fast algorithm has been previously proposed for this purpose. The nature of the constraints is local, so the algorithm cannot globally guarantee that every generated configuration is valid. Thus after finding a solution, an additional check should be conducted to ensure that a path exists between any possible source and destination pair. However, as it will be shown in Section V, the required number of final complete checks is in most cases very small.

1) Reconfiguration Procedure

When a packet arrives at a switch, it is destined for local PE or it should be routed using one of the six system ports. We use *L*, *R*, *D*, *U*, *F*, and *B* to indicate moving in left, right, down, up,

TABLE I. HCs & SCs FOR 3 BIDIRECTIONAL FAULTY LINKS (*EB*, *ED*, *YZ*)

Node	Soft Constraints	Hard Constraints	Helping Nodes
A	R < U, R < D, R < B	R < F	-
B	F < L, F < R	-	A
D	R < D	-	M
E	L < D, B < R	-	N (Left), F (Back)
F	L < U, L < D, L < F	L < B	-
M	U < L, U < F, U < B	U < R	-
N	U < R, U < F, U < B	U < L	-
P	D < L, D < F, D < B	D < R	-
Q	D < R, D < F, D < B	D < L	-
Y	R < U	-	P
Z	L < U	-	Q

front, or back directions, respectively. The decision of where to route the packet is made based on the result of the address comparison between *CRT* and *DST*. The distance of the two switches whose coordinates are (X_{CRT}, Y_{CRT}, Z_{CRT}) and (X_{DST}, Y_{DST}, Z_{DST}) is defined as:

$$\text{dist}(CRT, DST) = (|X_{DST} - X_{CRT}| + |Y_{DST} - Y_{CRT}| + |Z_{DST} - Z_{CRT}|) \quad (1)$$

This distance is the minimum number of hops that the packet should traverse until it reaches its destination. If a routing algorithm routes a packet in such a way that its distance from the destination switch decreases with each move, the algorithm will be live-lock free [16]. The live-lock refers to the situation in which a packet will not reach its destination, although it never gets blocked permanently. We call every move that decreases (increases) the distance of packet from its destination a *positive* (*negative*) move.

Sometimes there are multiple positive moves for a packet. For example, if *DST* is on the left, above, and front of *CRT,* taking each direction will be a positive move. A 3DMR-based routing table has one entry for each possible combination of positive moves, as shown in Fig. 1(a). The contents of this table indicate the relative priority of moving in different directions. For example, if in the entry corresponding to positive moves in up and right and front directions, ($G_XL_YG_Z$) in Fig. 1(a), the port identifier of the right port is given, then the priority of moving in the right direction is more than up and front move. We use "<" operator to show the priority of movements. In this case, we write $U<R$ and $F<R$.

When a node becomes faulty, it will inevitably need some helps from one of its neighbors (called *helper*) to route the packets. Since the switch with a faulty link might be forced to perform a negative move, the distance may increase and if the priorities of the *helper* are not properly set, it might return the packet to the original switch and causes live-lock. Hence, it would be necessary to put some constraints on possible priority combinations that the *helper* can use. As indicated before, the imposed constraints are generated according to the local effect of each fault and might just guarantee routing connectivity in the absence of other faulty links in the network.

```
CONF3DNETWORK(faulty_link_set, current_constraints, approach)
begin
  if ISEMPTY(faulty_link_set) then
    for each configuration conf satisfying all the constraints
      flag = ISCONFIGURATIONROUTTINGCONNECTED(conf)
      if (flag) then
        return conf;
      end if;
    end for;
    return NULL;
  else
    fulty_link = GETTHENEXTFAULTYLINK(faulty_link_set);
    n = node affected by faulty_link;
    for each possible helping node helping_node for n
      new_constraints = current_constraints +
        {imposed constraints by using helping_node as the helper};
      if COULDBESATISFIED(new_constraints, approach) then
        conf = CONF3DNETWORK(faulty_link_set – faulty_link,
                             new_constraints, approach);
        if (conf <> NULL) then
          return conf;
        end if;
      end if;
    end for;
    return NULL;
  end if;
end
```

Legend:
Keywords
variable_names
FUNCTIONNAMES

Figure 2. Reconfiguration Algorithm for 3DMR

2) Putting constraints for one faulty link

To depict our technique, assume we have one broken link (unidirectional) between node *N* and *M* in Fig. 1(b). Since the *NM* link is faulty, node *N* can select one of its four adjacent nodes in other dimensions (*i.e. K*, *Q*, *E*, and *W*) as the *helper*. In this algorithm we do not consider the neighbor in the same dimension as the faulty link (*O*, in this case) as a possible *helper*, because this will force us to consider non-local constraints and make the whole process more complex.

Suppose that we choose node *Q* as *helper*. This will impose some restrictions on 3DMR of *Q* to prevent live-lock situations. Suppose node *N* has a packet destined to node *M*. Since the *NM* link is faulty, *N* will send the packet to *Q*, instead. Now, node *Q* has a packet that should move both left and back. If, in routing table of *Q*, back moves have a higher priority than left moves, *Q* will return the packet to *N* and will cause a live-lock. Thus, for *Q*, the priority of the back move should be less than that of the left move ($B<L$). This constraint is a **Hard Constraint** (HC) for *Q*, because it has to be satisfied to have a live-lock free routing.

Moreover, we can use another type of constraint, called **Soft Constraint** (SC) to improve some network measures like path lengths reduction. Suppose that in Fig. 1(b) *N* is helped by node *Q* to route in the left direction. It is better to force switch *N* to first route the packets using its non-faulty links (*i.e.* right, front, back, up and down) if it is a positive move. This means that switch *N* will use the *helper Q* to route when no other choice exists. Thus we consider the following SCs for node *N*: $\{L<F, L<B, L<U, L<D\}$. $L<R$ is a useless constraint because moving toward both right and left are exclusive and cannot happen simultaneously. As another example, consider a network with three bidirectional faulty links (*i.e. EB*, *ED*, and *YZ* in Fig. 1(b)). TABLE I shows the assigned hard and soft constraints for each node. The nodes missing in the table have no constraints.

When we consider all possible faulty links and extract all required constraints, it would be probable to have a switch whose constraints are conflicting. For example, several faulty links could impose both $L<R$ and $R<L$ simultaneously on the routing table of a switch. Apparently, these are conflicting constraints and cannot be met at the same time. If in our example, we first choose *Q* as *helper* of *N* and after considering other nodes, we get trapped into a conflicting situation. It is very likely that this conflict has happened due to the choosing of *Q* as *helper*. So, we should now check the alternative choice(s) and test the selection of *K*, *E*, or *W* instead of *Q* as helper. Thus our algorithm has a backtracking nature and whenever it encounters some conflicts, it backtracks to the last point in which another alternative exists.

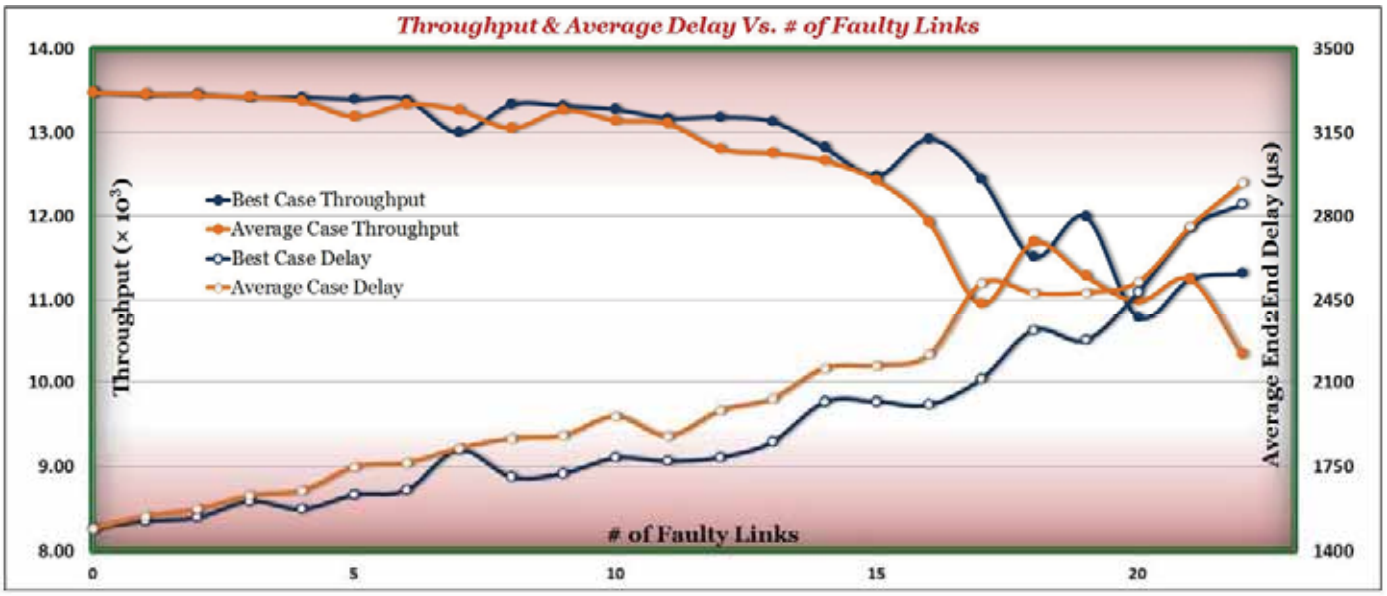


Figure 3. Throughput & Average End-to-End Delay for the Experiments (Best & Average Case)

3) Proposed reconfiguration algorithm

Fig. 2 demonstrates our proposed reconfiguration algorithm. The recursive CONF3DNETWORK gets three inputs: a set of faulty links that have not been handled yet, the constraints resulting from handling previous faults, and the behavior of the algorithm to manage hard and soft constraints (*i.e.* consider just HCs, consider both HCs and SCs, and consider HCs and apply SCs as much as no conflicting occurs). It selects one faulty link from the list and examines all possible *helpers* for the node affected by the faulty link. It repeats this process until no faulty link remains in the list. This situation happens when either all the faults are handled without conflict, or the set of constraints could not be satisfied. In the former case, it checks all the routing table configurations that satisfy the constraints. If this solution is a valid configuration (see Section III.B), it will be returned as result. Otherwise, the algorithm will backtrack to the point of the last choice. In the latter case, the algorithm will consider alternative *helper*(s) for the node affected by current faulty link. If there is no unchecked alternative, current invocation of CONF3DNETWORK returns **NULL**. If the first invocation returns **NULL** then the algorithm has failed to find a live-lock free configuration and thus, faulty network cannot be used even with degraded performance, because there are several unreachable nodes.

IV. IMPLEMENTATION ISSUES

To assess and evaluate the proposed approach, an efficient tool has been developed using C++ in eclipse environment. The inputs of the program are network structure, and desired fault pattern(s). Using proposed search algorithm, this tool tries to find the first, and not the best, valid configuration for the faulty network. As stated before, the time limitation imposed by the IC test process, has forced us to improve the speed of the algorithm not its optimality. In this section several aspects of the implemented tool is briefly described assuming a regular $n \times m \times q$ mesh structure. The extracted results are reported in the next Section.

A. Pseudo-random Fault Generation

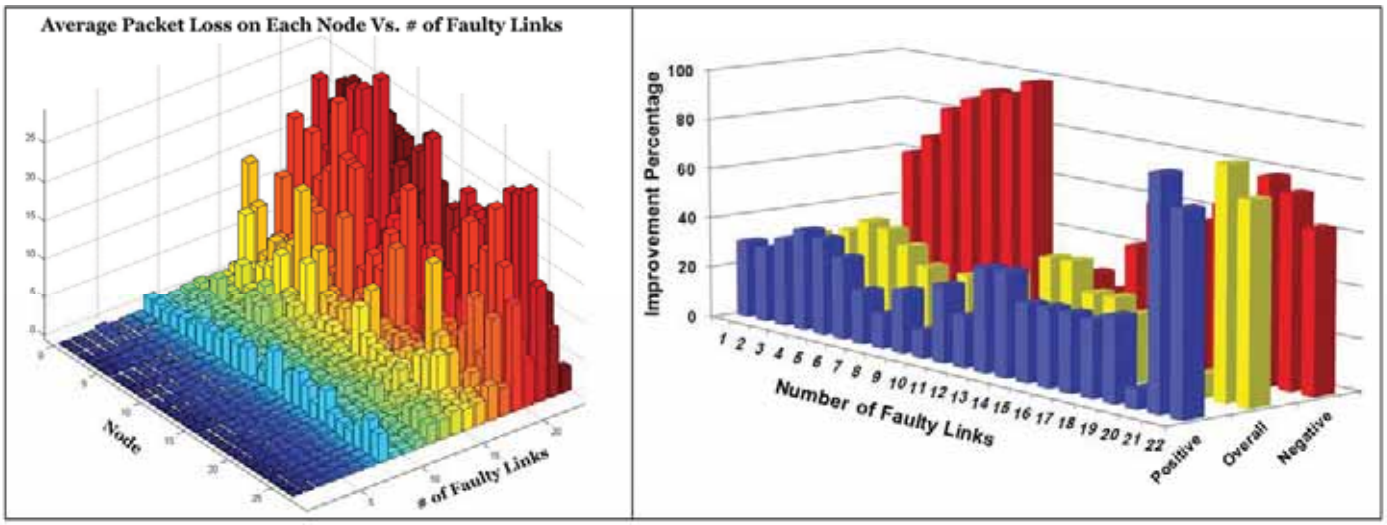
To investigate our algorithm with different faulty 3D mesh-based networks, a pseudo random fault generation mechanism is designed and implemented. The inputs for this part of the program are: the number of faulty unidirectional links (*i.e.* 0 to [2× [q× (3×n×m - (m+n)) - n×m]]), the start pattern of the injected faults, and the number of different fault patterns. Using these inputs, our tool automatically generates unique fault patterns and applies them into the mesh structure separately. In each case the program tries to find the first valid configuration, and reports the results (*e.g.* average link loads, required path check, number of backtracks and so on) for the degraded network. Since each physical fault usually affects a restricted area of the circuit, we incorporate another option into the program in order to make it possible to apply a fault on both direction of a link between any pairs of the (*SRC*, *DST*).

Our pseudo random fault generation mechanism is able to provide all unique fault patterns for a specific mesh structure in a reasonable time (*e.g.* less than half second for a *3×3×3* network). The number of faulty links in addition to the fault pattern (*i.e.* different faulty networks with the same number of faults) can be simply changed in the program to see and compare the corresponding effects on the overall network measures.

B. Algorithm Optimization

In this part, some optimization techniques that are utilized to accelerate our algorithm are explained. The presented algorithm has a backtrack nature and in some situation it takes a significant time to find a valid configuration. One way to minimize the execution time for these types of algorithms is to enforce backtrack tree grows in depth instead of width. Using this technique the required time to find a solution is greatly reduced.

To do this, first, we gathered all nodes affected by the faulty links. Then based on the number of *helpers* that each infected node can use, we sort the node list. At each step the



(a) (b)

Figure 4. (a) Average Pcket Loss on Each Node for the Experiments (b) Effectiveness of Proposed Techniques To Speed-up the Algorithm

node with the minimum number of *helpers* is pushed into the backtrack stack. Using this approach we force the state space tree to enlarge just in depth. Thus, the algorithm not only is able to find a solution for the faulty network (if there is any) sooner, but also it can return NULL in a much less time for the non-routing connectable faulty networks.

As shown in Section III.C.1), our algorithm just considers the local effects of the faults and does not guarantee to return a configuration which is globally routing connected. Thus additional check is mandatory to see whether the returned configuration results in a routing connected network or not (*i.e.* ISCONFIGURATIONROUTINGCONNECTED function in Fig. 2). So the way of implementing this function, has a significant influence on overall performance of the algorithm.

A configuration is valid when a path exists between each pair of (*SRC*, *DST*) based on the modified 3DMR. Obviously the number of required paths to be checked for each configuration equals to 2-permutation from $n \times m \times q$ or $P(n \times m \times q, 2)$. To reduce the required time for each configuration check, two different techniques are embedded into our algorithm. First, before starting the global path checking, all nodes are sorted based on the number of their faulty links. So if the configuration is not really a valid configuration, the algorithm immediately detects it (after considering just a few paths started with the most affected node) and investigates other solutions. In the second technique we avoid to check the paths which are already checked. Thus we select a (*SRC*, *DST*) pair with the maximum distance and then check whether there is a path between them or not. If a path exists between *SRC* and *DST*, all other pairs which are connected using the same path can be marked and eliminated from the checking process. This procedure continues until no pair of (*SRC*, *DST*) remains unchecked or the configuration marked as a non-valid configuration.

C. Script Generation for NS

Network Simulator (NS) is a widely used open source discrete event simulator for networking researches. The obtained results from the previous steps can be used by our tool to provide a NS script. If the traffic pattern for NS-simulation is not defined during TCL script generation, simple mpeg2 traffic is considered. The routing tables of each node in the network are filled based on the 3DMR PRTs extracted from the algorithm for the degraded network. The script is then simulated using NS-2 simulator and a trace file which contains all the network parameters, is reported.

V. EXPERIMENTAL RESULTS

In this section we will evaluate the proposed algorithm. A 3×3×3 mesh-based network with 27 nodes and 54 pairs of bidirectional links has been used as our case study. We injected a uniform traffic pattern (similar to a simple *Mpeg2* decoder traffic) to the network. Each packet consists of four 128-bit flits. We assumed that each switch has 3 virtual channels per port and each virtual channel buffers just one packet. The link capacity is 1 Gbps. We also assumed having a packet generation rate according to a *Poisson* distribution with a rate of one packet every 2.5 μs (*i.e.* 400 Kilo packet per second). To evaluate the performance of the proposed routing mechanism, *NS-2* simulator is used and the results are compared to those of the non-faulty networks. We assumed that occurrence of any manufacturing fault makes a communication link unusable in both directions (See Section IV.A). To more precisely assess the efficiency of the presented technique, almost all fault patterns were applied to the network using the pseudo random fault generation mechanism (depicted in Section IV.A). We considered up to 22 faulty bidirectional links in our experiments.

Fig. 3 demonstrates throughput and end-to-end delay for a network with zero to 22 faulty links. For each case the average and the best case among different fault patterns are reported in this figure. This diagram shows that the general trend of the extracted parameters is to get worse when the number of faulty links increases, due to the limited functional non-faulty resources. However, there are some deviations that can be attributed to the sensitivity of the results to both the number and pattern of the faulty links.

Fig. 4(a) depicts average packet loss on each node for different faulty networks. The third axis is used to show the number of faulty links. This figure also confirms that the average packet loss increases proportionally with the number of faults. Fig. 4(b) shows the effectiveness of the techniques introduced in Section IV.B to accelerate the presented algorithm. Normalized execution times (percentage of improvement due to the algorithm optimization) were reported in this diagram in three separate categories. These categories shown as *Overall*, *Positive*, and *Negative* in Fig. 4(b) are: all the networks used in our experiments, those faulty networks that our algorithm is able to find the corresponding degraded network, and those faulty networks that are either not physically connected or non-routing connectable, respectively.

In TABLE II, the first column shows the number of faulty links. The second column gives the number of backtrack operations to find a valid configuration for the corresponding faulty networks. The most time consuming part of the algorithm is the final routing connectivity check, and the presented techniques are utilized to reduce the number of times that ISCONFIGURATIONROUTINGCONNECTED function should be invoked. The third column of TABLE II shows the number of configuration checks, which in all cases are very small, compared to exhaustive search. The next two columns provide the average path length and the average link load of the networks under the uniform traffic. The last column shows the ability of our algorithm to find a valid configuration for each faulty network. For example the 10th row indicates that in almost 90% of physically connected networks with 15 faulty links, our proposed algorithm is able to find a valid configuration for the corresponding degraded network.

VI. CONCLUSION

The emergence of the state of the art deep submicron fabrication process and the increased complexity of the current designs, have made the products more and more susceptible to the manufacturing faults. Reconfiguring the faulty chip to bypass the faulty modules and get the work done using the remaining non-faulty ones (usually called graceful degradability) is one of the most popular approaches in fault-tolerant design area. In this paper we have considered the problem of the faulty links in 3D mesh-based NoCs and have proposed a heuristic search algorithm to reconfigure the faulty networks in order to make such faulty chips reusable. This approach is based on using an efficient and scalable hardware and a fixed-size programmable routing tables in each network switch. The experimental results show that the algorithm not only is able to reconfigure almost 90% of physically connected faulty networks, but also it considerably reduces the required search effort as compared to the exhaustive search method. Although, the degraded networks show acceptable measures, the general trend of performance parameters is to get worse when the number of faulty links increases.

TABLE II. EXTRACTED SIMULATION RESULTS FOR OUR EXPERIMENTS

# of Faulty Links	Req. Checks	Backtrack Opr.	Avg. Path Length	Avg. Load	Success Percentage
0	1	0	2.7692	18	100%
1	1	0	2.8034	18.5660	100%
2	1	0	2.8205	19.0385	100%
3	1	0	2.9117	20.0392	70.46%
5	1	0	2.9402	21.0612	80.34%
7	21	4	3.2051	23.9362	75.93%
10	1	0	3.1054	24.7727	65.54%
12	1	0	3.0855	25.7857	80.77%
15	4	2	3.3476	30.1282	89.65%
17	1	0	3.4672	32.8919	94.56%
18	26	7	3.7236	36.3056	95.73%
19	26	7	3.7607	37.7143	89.25%
20	66196	27460	3.9459	40.7353	92.28%
21	115253	31892	4.1766	44.4242	90.94%
22	202162	58739	4.3989	45.25	91.21%

REFERENCES

[1] Semiconductor Industry Association, International Technology Roadmap for Semiconductors, Edition 2010, available: http://www.itrs.net/reports.html.

[2] B. S. Feero and P. P. Pande, "Networks-on-Chip in a Three-Dimensional Environment: A Performance Evaluation," *IEEE Transactions on Computers*, vol. 58, no. 1, pp. 32-45, Jan. 2009.

[3] R. Saleh et al "System-on-Chip: Reuse and Integration," *Proceedings of the IEEE*, vol. 94, no. 6, pp. 1050-1069, Jun 2006.

[4] H. G. Lee, N. Chang, U. Y. Ogras, and R. Marculescu, "On-chip communication architecture exploration," *ACM Transaction on Design Automation of Electronic Systems*, vol. 12, no. 3, pp. 1-20, Aug. 2007.

[5] A. B. Ahmed, A. B. Abdallah, and K. Kuroda, "Architecture and Design of Efficient 3D Network-on-Chip for Custom Multicore SoC," in Proc. *International Conference on Broadband, Wireless Computing, Communication and Applications (BWCCA'10)*, Nov. 2010, pp.67-73.

[6] A. W. Topol et al., "Three-dimensional integrated circuits," *IBM Journal of Research and Development*, vol. 50, no. 4/5, pp. 491–506, 2006.

[7] S. F. Al-Sarawi, D. Abbott, and P. D. Franzon, "A review of 3-D packaging technology," *IEEE Transaction on Components, Packaging, and Manufacturing Technology, Part B: Advanced Packaging*, vol. 21, no. 1, pp. 2-14, Feb 1998.

[8] N. Honarmand, A.Shahabi, H. Sohofi, M. Abbaspour, and Z. Navabi, "High Level Synthesis of Degradable ASICs Using Virtual Binding," in Proc. *VLSI Test Symposium (VTS'07)*, May 2007, pp. 311-317.

[9] C. Chao, K. Jheng, H. Wang, J. Wu, and A. Wu, "Traffic- and Thermal-Aware Run-Time Thermal Management Scheme for 3D NoC Systems," in Proc. *International Symposium on Networks-on-Chip (NOCS'10)*, May 2010, pp. 223-230.

[10] A. Kohler, G. Schley, and M. Radetzki, "Fault Tolerant Network on Chip Switching With Graceful Performance Degradation," *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 6, pp. 883-896, Jun. 2010.

[11] M. Pirretti et al., "Fault Tolerant Algorithms for Network-On-Chip Interconnect," in Proc *IEEE Symposium on VLSI (ISVLSI'04)*, Feb. 2004, pp.46-51.

[12] A. Hosseini, T. Ragheb, and Y. Massoud, "A fault-aware dynamic routing algorithm for on-chip networks," in Proc *IEEE International Symposium on Circuits and Systems(ISCAS'08)*, 2008, pp. 2653-2656.

[13] D. Fick, A. DeOrio, G. Chen, V. Bertacco, and D. Blaauw, "A highly resilient routing algorithm for fault-tolerant NoCs," in Proc. *Design, Automation & Test in Europe (DATE'09)*, Apr. 2009, pp. 21-26.

[14] I. Loi, S. Mitra, T. H. Lee, S. Fujita, and L. Benini, "A low-overhead fault tolerance scheme for TSV-based 3D network on chip links," in Proc. *ICCAD'08*, Nov. 2008, pp. 598-602.

[15] A. Mejia, J. Flich, J. Duato, S. Reinemo, and T. Skeie, "Segment-based routing: an efficient fault-tolerant routing algorithm for meshes and tori," in Proc. *IPDPS'06*, Apr. 2006, pp. 10.

[16] A. Shahabi, N. Honarmand, H. Sohofi and Z. Navabi, "Degradable mesh-based on-chip networks using programmable routing tables," *IEICE Electron. Express*, vol. 4, no. 10, pp. 332-339.

Self-Calibration Using Functional BIST for Transient-Fault-Tolerant Sequential Circuits in Severe Electromagnetic Environment

Masayuki Arai Aromhack Saysanasongkham Kenta Imai Yoshifumi Koyama Satoshi Fukumoto
Graduate School of System Design, Tokyo Metropolitan University
6-6, Asahigaoka, Hino, Tokyo 191-0065, Japan
Contact Author: Masayuki Arai m-arai@tmu.ac.jp Tel/Fax: +81-42-585-8453

Abstract

We propose a fault-tolerant sequential circuit, which can tolerate periodic transient faults due to electromagnetic radiation caused by DC-AC inverters. The proposed circuit repeatedly applies functional BIST to measure the duration of a transient fault, and then appropriately negates clock signals during normal operation.

keywords: transient error tolerant circuit, self-calibration, on-line functional BIST, electromagnetic radiation, DC-AC inverter

1. Introduction

Recently, with integration of power electronics and information technology, DC-AC inverter circuits tend to exist just beside a logic circuit, such as seen in smart grid, solar cell, and electronic vehicle. Development of DC-AC inverter has been continuously targeting on higher switching frequency, higher voltage, higher power density, integration with power source, control logic, and/or actuator in the final products, and lower manufacturing costs [1]. These trends could lead to problems on the operation of the logic circuits beside due to severe electromagnetic environment [2], which would not be able to be handled as conventional soft-error model.

In this study we discuss a fault-tolerant sequential circuit which can correctly operate even under severe electromagnetic environment due to high-frequency and high-power DC-AC inverters. First, we set up a fault model considering electromagnetic radiation of a DC-AC inverter. On the bases of the fault model, we introduce a transient-fault-tolerant sequential circuit with clock gating, where the period of the clock gating is calibrated by a functional Built-in Self-Test (BIST) circuit.

2. Fault model for electromagnetic radiation by DC-AC inverter

Figure 1 shows an example of DC-AC inverter, which is a single-phase half-bridge pulse-width modulation (PWM) inverter. Two insulated gate bipolar transistor (IGBT) inverters (Q1, Q2) are turned on and off by control signals (G1, G2) in order to convert source DC voltage (V_{dc}) into two-valued AC voltage (v_{out}). The output AC might be further transformed into pure sine curve when used as utility power, or just input directly into actuators. Basic frequency of output PWM wave, which also corresponds to basic switching frequency of inverters, is typically in the range of 1 KHz ~ 1 MHz [1,2].

When the inverters turn on and off, internal noise current is generated. The characteristics of the internal noise current, such as S/N ratio, duration, and noise frequency, are affected by electric characteristics of individual elements in the DC-AC inverter circuit as well as load circuit, and thus each DC-AC

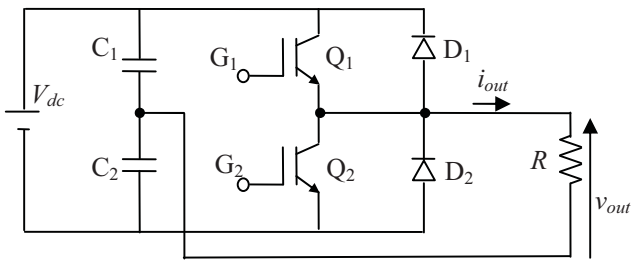
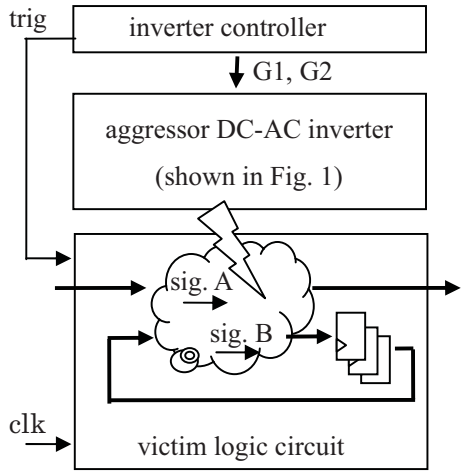


Fig. 1 Example of DC-AC inverter: single-phase half-bridge pulse-width-modulation inverter.

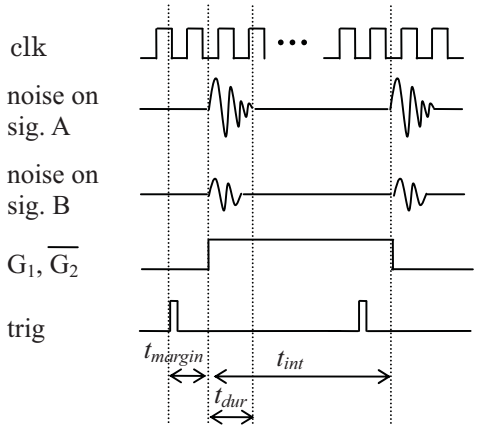
inverter might have different noise characteristics. The characteristic of the internal noise current, however, can be considered as stable on a given DC-AC inverter, as long as it is utilized in a stable environment during a short period of time.

High-speed and high-power DC-AC inverter might cause severe electromagnetic interference (EMI), and thus near-field noise (radiative noise) might be generated in the logic circuit which is located close to the DC-AC inverter [2]. Once the logic circuit is affected by near-field noise, it might be periodically suffer from the noise, according to switching activity of the inverters. In this study we model the effect of near-field noise as transient faults, as shown in Fig. 2. That is:

- Multiple wires on the logic circuit is inverted simultaneously (multi-error transient: MET).



(a) circuit and signal model



(b) timing chart

Fig. 2 Model of periodic transient fault caused by electromagnetic radiation from DC-AC converter

- Flip-flops are never inverted (non single/multi-error upset), but might capture the effect of MET.
- Transient faults occur periodically, where intervals between two transient faults (t_{int}) is larger enough (say, > x10) than clock cycle of the logic circuit, and it is not constant because of PWM.
- The logic circuit can be noticed the trigger of a transient fault by monitoring controlling signals of the DC-AC inverter. The inverter controller further might be able to indicate the occurrence of the noise with a sufficient timing margin (t_{margin})
- The maximum duration of a transient fault (t_{dur}) might be longer or shorter than operational clock cycle of the logic circuit. It is determined by the characteristics of individual logic circuit and surrounding environments including DC-AC inverter, and is not changed as long as the logic circuit is powered-on.

3. Transient-fault tolerant sequential circuit with self-calibration of clock gating

According to the periodic transient fault model mentioned in the previous section, we propose an architecture of transient-fault tolerant circuit. As a first step of the study, we take a simple approach based on clock gating where the clock signal is negated with appropriate duration corresponding to occurrence of a transient fault. The duration of a transient fault, t_{dur}, depends on the characteristic and environment of the individual logic circuit, and thus it has to be measured at the final system, not at production test. Therefore, we apply on-line functional BIST to calibrate clock gating control circuit.

Figure 3 shows a conceptual block diagram of the proposed circuit architecture. Once a noise trigger is input, the clock control unit negates the clock signal sent to the target logic circuit for a certain period, which is specified by t_{dur} register. The value of t_{dur} register is calibrated by the functional BIST unit when the circuit is powered-on. First, setting small value into the register, a functional BIST session is executed. Then, the BIST session is repeatedly executed with increased t_{dur} value until the test is passed. Finally, some margin is added to t_{dur}, and the circuit transits to the normal-operation mode.

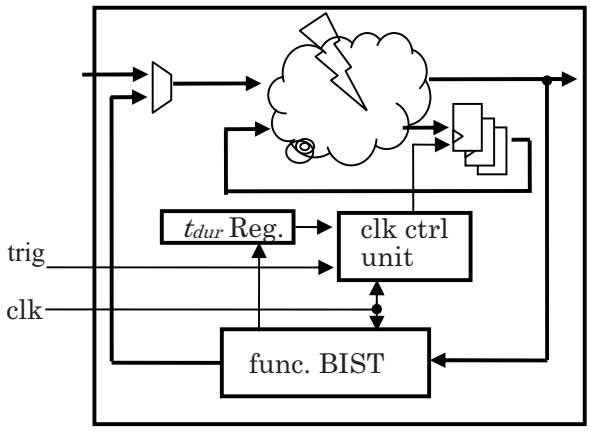


Fig. 3 Diagram of proposed transient-fault-tolerant sequential circuit with self-calibration of clock gating

4. Experimental Results

We implemented the proposed fault-tolerant architecture, and evaluated its functionality and hardware overhead.

Figure 4 shows the architecture of evaluation circuit implemented. As a target logic circuit, we applied simple 8-bit counter with a register (Reg) consisting of 8 FFs. To perform power-on BIST application, we added two 1-bit registers, test_mode and finish, into BIST circuit. When powered-on, the BIST circuit transits to the test mode, and inputs the test patterns for a given clock cycles. The test clock cycle is counted up as Check_reg in the BIST circuit. Because the target circuit is a simple counter, the test pattern only consists a pulse of resetFF singal, and the expected value for the output of the target circuit when the test is terminated is simply corresponding to the value of Check_reg.

Figure 5 shows the state transition diagram for the BIST and clock control circuit. When the power is turned on, reset signal should be activated in order to start power-on BIST sequence. The BIST circuit waits for the pulse of trig, and then transits to test mode. At the test mode, as mentioned in the previous section, test patterns are repeatedly applied to determine the value of t_{dur}. Once the value of t_{dur} is determined, finish flag becomes high, and the circuit transits to normal state, where only clock control circuit is activated in order to gate the internal clock.

We described the evaluation circuit by verilog HDL, and simulated by Icarus Verilog simulator [3]. Figure 6 shows the simulation results. The setting of the simulation is as follows.

- clock cycle: 10 ns
- trig (inverter) cycle: 100 ns
- timing margin between trig and noise: 10 ns
- noise duration: 10 ns
- initial value of t_{dur} : 10 ns (1 clock cycle)
- length of test pattern: 10 clock cycle

In Fig. 6(a), we can see that all registers are first initialized and nothing is done until first pulse of trig is input. When the trig is input, the internal clock signal int_clk is negated during 10 ns (1 clock cycle) which is specified as t_{dur}

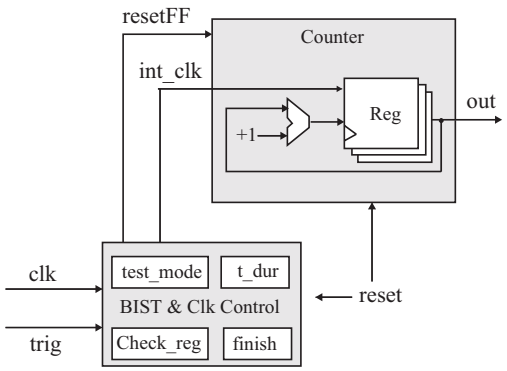


Fig. 4 Evaluation circuit of the proposed scheme: target circuit is arranged as simple 8-bit counter

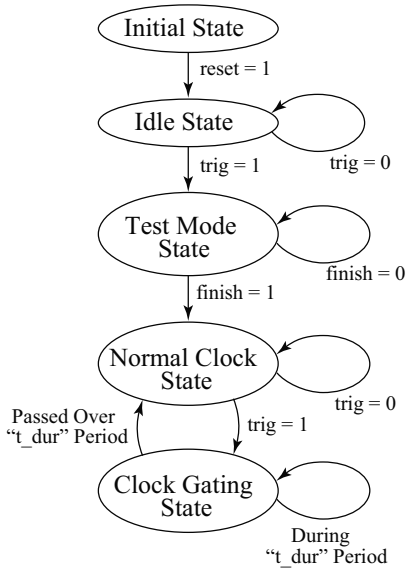
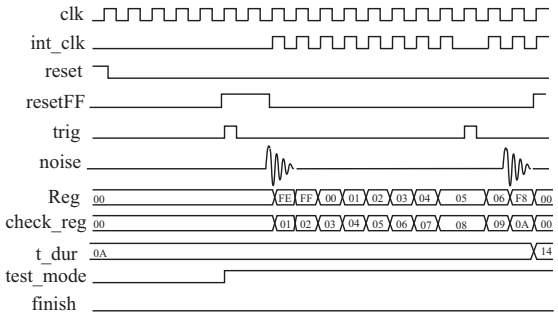


Fig. 5 State transition diagram for BIST and clock control unit

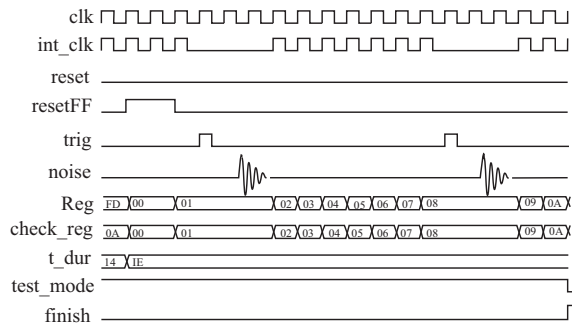
register. Also, resetFF becomes high, the registers in the target circuit (Reg) are zero-cleared, and test_mode is activated. The figure shows that 10 ns of clock gating cycle from trig is too short to avoid to capture the erroneous value into Reg. At the rightmost part of Fig. 6(a), the BIST circuit detects that the test is failed by comparing the output of the target circuit to the expected value stored in check_reg, and then it re-runs the test with the increased value of t_{dur}.

Because the edges of trig and clk are not guaranteed to be synchronized to each other, BIST circuit might sometimes set larger value of t_{dur} than the sum of timing margin between trig and noise and noise duration. Figure 6(b) shows the continued test mode where BIST circuit is noticed that 2 clock cycles of interval is not sufficient, and BIST is re-executed with 3 clock cycles of t_{dur} value. In this case, clock gating successfully avoids to capture the effects of the noise, and thus the circuit would transit to normal mode.

We then logic-synthesized the described evaluation circuit by Synopsys Design Compiler with the standard cell library class.lib, and calculated the hardware overhead of the proposed architecture. The original target circuit of 8-bit counter is arranged by 90 logic gates. By applying our proposed fault-tolerant technique, clock control unit of 20 gates and BIST circuit of 65 gates are inserted, and the target circuit is widened into 123 gates because of input selector. In this experiments the hardware overhead was large value of about 212%. The BIST and clock control unit, however, is expected to be arranged by still the small number of gates even for large target circuit. The hardware overhead of the proposed scheme can become very small, when it is applied to practical-scale target circuits.



(a) power-on and test fail



(b) test pass

Fig. 6 Simulation results of BIST test mode

5. Conclusions

We proposed a fault-tolerant sequential circuit, targeting on periodic transient faults. The proposed circuit repeatedly applies functional BIST to measure the duration of a transient fault, and then appropriately negates clock signals during normal operation. Applying to simple counter as a target circuit, we evaluated the functionality and hardware overhead of the proposed architecture.

In this study we assumed that faults only occur on signals in the target logic circuits, and that BIST circuit, clock control circuit and clock signal are well-protected by electromagnetic compatibility (EMC) techniques. Tolerance for more severe faults, such as faults on clock or BIST circuit, is still one of the open issues.

Acknowledgements

Authors would like to thank Prof. Keiji Wada, Prof. Toshihisa Shimizu, and Mr. Zen-nosuke Ariga, of Tokyo Metropolitan University, for their valuable suggestions.

References

[1] J. W. Kolar, U. Drofenik, J. Biela, M. Heldwein, H. Ertl, T. Friedli and S. Round, "PWM Converter Power Density Barriers," IEEJ/IAS Trans. D, Vol. 128, No. 4, pp.468-480, 2008.

[2] Z. Ariga and K. Wada, "Analysis and Evaluation of Near Field Noise Voltage on Power Electronics Circuits," Int'l Conf on Power Electronics and Drive Systems, pp. 1014-1019, 2009.

[3] Icarus Verilog, http://iverilog.icarus.com

On the Functional Test of Branch Prediction Units

E. Sanchez, M. Sonza Reorda

Dipartimento di Automatica e Informatica
Politecnico di Torino
Torino, Italy
{ernesto.sanchez, matteo.sonzareorda}@polito.it

1. Introduction

Embedded system applications characterized by high performance requirements often rely on RISC or superscalar processors. In order to increase their performance, it is common practice to equip them with highly efficient and sophisticated Branch Prediction Units (BPUs), which can significantly decrease the negative impact of branches [1].

However, the complexity of these architectures, combined with the increased sensitivity to faults of new technologies, ask for suitable techniques able to effectively detect possible faults affecting them, at the end of the manufacturing process, for incoming inspection, and during the operational life (on-line test) [2].

Testing BPUs can obviously be performed resorting to traditional test solutions, such as scan test (for testing the chunks of logic), or BIST (to test the memory blocks). However, these solutions are often inadequate. First of all, because scan solutions can hardly be exploited during the operational life, even for non-concurrent on-line testing; secondly, because companies involved in processor design and manufacturing tend not to disclose details about internal Design for Testability (DfT) architectures, in order to better achieve IP protection; this means that both for incoming inspection, and for end-of-production test of System-on-Chip (SoC) devices, these DfT structures can hardly be adopted; thirdly, because scan test is generally inadequate for testing delay faults, that usually require at-speed stimuli application and response observation (not to mention the overtesting scan test tends to produce). For all these reasons, a functional test approach based on developing suitable test programs to be executed by each core and on observing the produced results is often the only suitable solution, provided that effective techniques are available for generating the required test programs. This approach is also known as Software-Based Self-Test (SBST) [3].

Branch Prediction Units are among the most critical components within high-performance embedded systems, not only because they are deeply embedded into the processor architecture, but also because their behavior can significantly affect the performance of the whole system. More specifically, faults affecting BPUs often do not cause the generation of erroneous results, but rather slow down the system, increasing the number of mispredictions and possibly causing the system not to match the expected target in terms of performance. This means that suitable techniques are required for observing the processor behavior when the test programs are executed, not only in terms of produced results, but also in terms of performance [9] (e.g., by monitoring the bus, or by measuring the time to execute a given piece of code).

BPU testing has been the subject of a few previous papers. In [4] the authors propose a hardware-based method, which requires the insertion of proper circuitry in the processor for BPU test. The technique proposed in [5] follows the SBST approach, and mainly focuses on BPUs based on the Branch Target Buffer (BTB) architecture. This paper reports a very convincing analysis of faults affecting BPUs, and proposes the usage of performance counters to detect them. However, the proposed method does not achieve full coverage of stuck-at faults, and requires very long test times. In [6], the authors use faults in BPUs as the typical example of the so-called Performance Degrading Faults, and analyze their impact on the performance of a processor, showing that their proper identification can significantly help improving the yield. The authors of [7] propose a method to make Branch Prediction Units resilient to faults: however, the method is based on first detecting possible faults affecting each BPU, and then reconfiguring it, which raises even further the issue of how to test BPUs. Finally, in [11] we proposed a technique for generating functional test programs able to test faults affecting not only the memory part of a BPU, but also the logic associated to it, when the BPU is based on a Branch History Table (BHT).

The purpose of this paper is first of all to extend the method proposed in [11] to the case of BPUs adopting the BTB architecture: in that case the BPU module is normally dominated by the memory blocks, and the paper describes a method to write a test program implementing a March algorithm [8] suitable to test them. Experimental results gathered on the SimpleScalar platform [10] allows the reader to practically evaluate the size and duration of the resulting test programs, showing that they are also suitable for on-line testing.

The second goal of the paper is to show that in the case of BPUs it is possible to generate effective test programs starting from rather limited information about the internal implementation of the module, and only working on the function it implements. Since the approach does not require the knowledge of any implementation detail, it is well suited to be adopted by system companies for both Incoming Inspection, and on-line test, as well as by semiconductor companies producing SoCs, when they decide to follow the functional approach (e.g., because they don't have access to structural information about the processor core).

The test approach proposed in this paper belongs to the SBST family; therefore, it can be applied at-speed, does not require any change in the processor or BPU hardware, and is particularly suitable to test delay faults. Moreover, being based on test programs to be executed by the processor, it can be activated at any time, even when the system is already in its operational phase. The relatively short duration of the test program makes it easily applicable even during concurrent on-line testing.

2. Proposed approach

Testing a BTB mainly requires testing the memory storing the address and target fields, including the logic for accessing memory cells and reading/writing to it. Testing the BTB memory can be achieved by resorting again to a March algorithm, properly implemented by a suitable sequence of instructions executing the corresponding sequence of accesses to the BTB memory.

Following the same approach adopted in [11], we identified first the instructions implementing the read and write operations on a BTB word, and developed solutions for allowing these instructions to be executed in the order mandated by the generic March algorithm.

Let assume that the processor code memory is composed of $M=2^m$ words, and the BTB is composed of $N=2^n$ words. Reminding that each BTB entry includes an address and a target field, let denote by a_i and t_i the content of its i-th line at a given time.

Accessing the x-th line in the BTB is obtained by forcing the processor to execute an instruction whose n least significant address bits (neglecting the k least significant bits, being 2^k the byte size of each instruction) hold the value x. Obviously, the least significant k bits of the instruction address always hold the value 0 for sake of alignment.

Denoting by $<a_x,t_x>$ the content of the x-th line of the BTB, reading from it may correspond to executing a branch instruction stored in a memory location whose address holds the value a_x:x:$zero^k$ (corresponding to the concatenation of three fields, whose length in bits is m-n-k, n, and k, respectively, and $zero^k$ is a sequence of k 0s) and performing a branch to the address t_x. The line correctly stores the expected value iff the BTB correctly predicts the branch target.

On the other side, writing into the x-th line of the BTB may correspond to executing a branch instruction stored in a memory location whose address holds the value b_x:x:$zero^k$, where b_x is different than the value a_x currently stored in the BTB line; hence, the instruction outcome can not be correctly predicted when the BTB is accessed and causes an update in the BTB content. Let denote by r_x the target address of the instruction: the considered instruction causes the value $<b_x, r_x>$ to be written in the x-th line of the BTB. In particular, the value b_x is written in the address field, while the value r_x is written in the target one.

In most cases the conventional 0 and 1 values used by the generic March algorithm when testing a m-bit memory can be freely chosen, provided that 0 = s (s being any value on m bits), and 1=not(s).

Based on the above observations, and assuming that the conventional 0 value corresponds to the concatenation of two values a_0 and t_0, the sequence of instructions implementing a March Element such as ↓w0 is a sequence of branch instructions stored at consecutive addresses in memory (starting from the address a_0:$zero^n$:$zero^k$ where $zero^n$ is a sequence of n 0s, and $zero^k$ is a sequence of k 0s), and all jumping to the same address t_0. The value of a_0 depends on where the memory area to store the sequence is located. Clearly, implementing a March Element such as ↑w0 requires executing the same sequence of branch instructions in the reverse order.

These requirements can be fulfilled by

- Using as branch instruction the procedure call instruction (denoted as JAL in the following)
- Filling the procedure body with the instruction returning to the calling program, only; in most processors, this instruction corresponds to a register branch instruction (i.e., a branch instruction whose target address is stored in the specified register), which does not rely nor affect the BTB; specifically, the return instruction corresponds to a register branch instruction whose parameter register is the same register where the call instruction saves the return address; if the March Element requires the JAL instructions to be executed in their order, this is sufficient; if it requires them to be executed in the reverse order, some additional instructions are required in the procedure body, that suitably adjust the value of the return register, so that the execution flow goes to the previous JAL instruction, instead of the next.

Fig. 1 reports the code implementing the ↓w0 March Element on a N-element BTB. For sake of clarity, and without any loss of generality, we adopted the assembly language used by the SimpleScalar simulation environment [10], which is similar to the MIPS one. As it happens in several modern processors, we assume that the procedure call instruction stores in a given register (in this case \$31) the return address. Hence, the body of the procedure simply contains the return instruction, corresponding to J \$31. In such a way, at the end of the i-th execution of the procedure, the execution flow returns to the (i+1)-th JAL instruction.

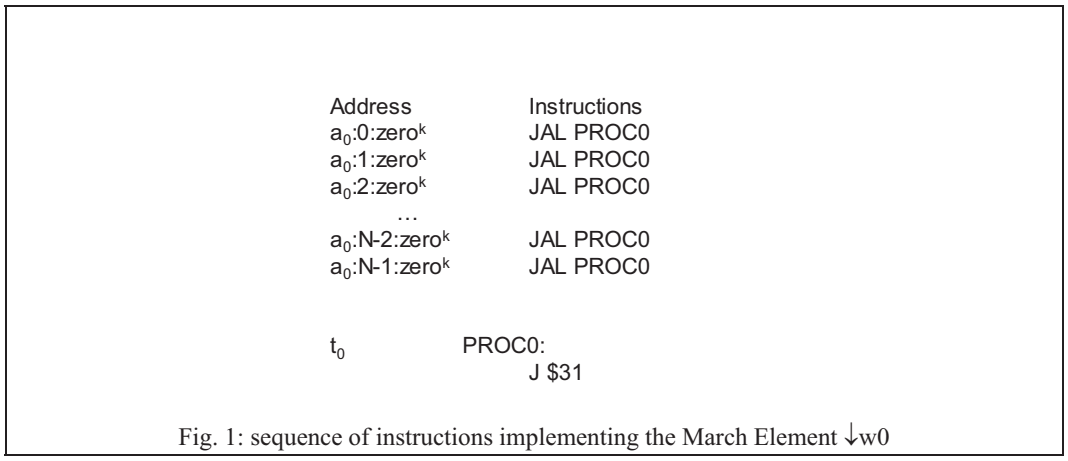


Fig. 1: sequence of instructions implementing the March Element ↓w0

Fig. 2 shows the code implementing the March Element ↑w0 on the same N-elements BTB. This case is more complex, since the sequence of JAL instructions must be executed in the reverse order. Hence, the body of the procedure must also contain some instructions aimed at adjusting the value of \$31 so that at the end of the procedure PROC0 the control is passed to the next JAL instruction according to the desired order (i.e., to the previous one in the sequence). An effective solution for doing so lies in using a vector VET storing the set of offset values to be added to the return address stored in \$31, one for every call to the procedure: in such a way, at the i-th execution of the procedure \$31 is incremented by the negative value stored in the i-th element of the vector VET, thus obtaining the address of the JAL instruction to be executed at the (i+1)-th step. In the considered case, the values stored in VET are $-(2\times2^k)$, $-(2\times2^k)$, …, (2×2^k), $(2\times2^k\times(N-1))$, being 2 the size in bytes of each instruction. The last value in VET allows jumping to the first instruction after the sequence of JAL instructions corresponding to the March Element. Clearly, in this case the first JAL instruction to be executed is the last (i.e., the one at the address a_0:N-1:zero^k). By suitably adjusting the values stored in VET, and jumping to the first or last instruction in the sequence, one can thus implement either the ↓w0 or the ↑w0 March Element.

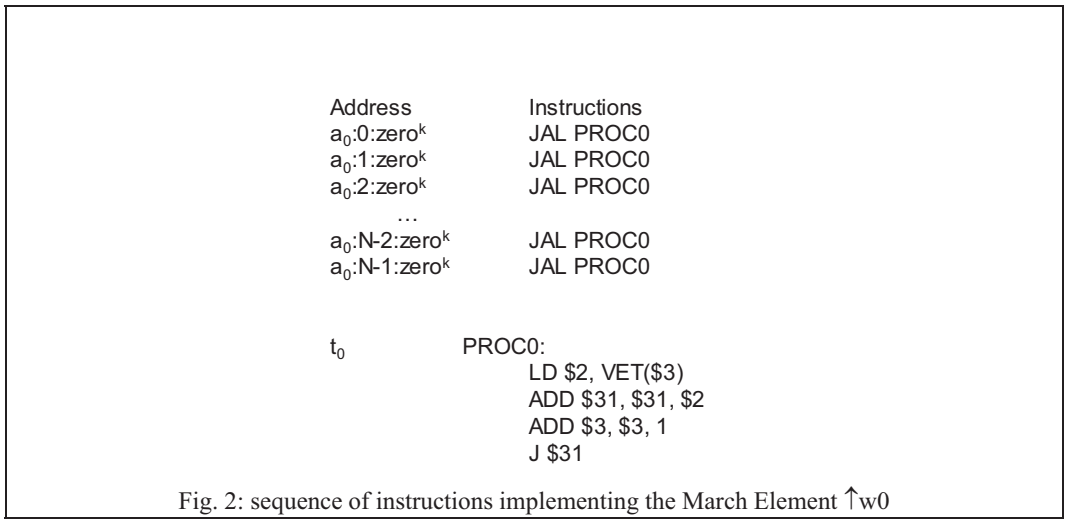


Fig. 2: sequence of instructions implementing the March Element ↑w0

Finally, March Elements corresponding to reading or writing the conventional 1 value (corresponding to the value not(s), being s the value corresponding to the conventional 0) can be implemented by resorting to

- a sequence of JAL instructions located at the consecutive addresses starting from a_1, being a_1=not(a_0)
- a procedure PROC1 located at the address t_1=not(t_0) and composed of the same instructions of PROC0.

Using the above techniques one can easily transform every March Element into the corresponding test program. The only limitation lies in the fact that it is not possible to perform a write operation on the BTB without performing immediately before a read operation on the same line: in fact, a write operation corresponds to a branch instruction updating the BTB (typically during the Execute stage), and its execution necessarily accesses the BTB for getting the prediction (hence, performing a read operation) already during the Fetch stage.

3. Experimental results

In order to validate the proposed approach we resort to SimpleScalar [10], an open-source system software infrastructure widely used for computer architecture research and teaching. Validation of the correctness of the proposed technique was achieved by implementing the code for the MATS+ March algorithm, checking that the corresponding instruction cause the desired read and write operations on the BTB memory.

Moreover, experimental results that are currently being gathered allow to evaluate the characteristics of the method in terms of code memory and execution time requirements.

4. Conclusions

We described a method for testing the circuitry implementing the Branch Prediction Unit of a processor, assuming that it is based on the Branch Target Buffer architecture,

and without resorting to specific information about its implementation. The resulting test program can be used in different steps: at the end of the manufacturing step, exploiting its ability to perform an at-speed test; during incoming inspection, thanks to the fact that it does not rely on any information about the implementation of the circuitry; during on-line test, due to the fact that the method is relatively fast and does not require any special hardware to be applied. Experimental results that are currently being gathered show the characteristics of the approach in terms of code memory and execution time requirements.

References

[1] J.L. Hennessy, D.A. Patterson, "Computer Architecture: A Quantitative Approach", Morgan Kaufmann Publishers, 2007

[2] M.L. Bushnell, V.D. Agrawal, "Essential of Electronic Testing", Kluwer Academic Publishers, 2000

[3] M. Psarakis, D. Gizopoulos, E. Sanchez, M. Sonza Reorda, "Microprocessor Software-Based Self-Testing", *IEEE Design & Test of Computers*, vol.27, no.3, pp.4-19, May-June 2010

[4] S. Almukhaizim, P. Petrov, A. Orailoglu, "Low-Cost, Software-Based Self-Test Methodologies for Performance Faults in Processor Control Subsystems", Proc. IEEE 2001 Custom Integrated Circuits Conference, pp. 263-266

[5] M. Hatzimihail, M. Psarakis, D. Gizopoulos, A. Paschalis, "A Methodology for Detecting Performance Faults in Microprocessors via Performance Monitoring Hardware", Proc. IEEE International Test Conference, 2007, paper 29.3

[6] T.-Y. Hsieh, M.A. Breuer, M. Annavaram, S.K. Gupta, K.-J. Lee, "Tolerance of Performance Degrading Faults for Effective Yield Improvement", Proc. IEEE International Test Conference, 2009, Lecture 3.1

[7] S. Almukhaizim, O. Sinanoglu, "Error-Resilient Design of Branch Predictors for Effective Yield Improvement", IEEE Latin-American Test Workshop, 2011

[8] A. J. Van de Goor, "Testing Semiconductor Memories, Theory and Practice", John Wiley & Sons, 1991

[9] W. J. Perez, J. Velasco-Medina, D. Ravotto, E. Sanchez, M. Sonza Reorda, "A Hybrid Approach to the Test of Cache Memory Controllers Embedded in SoCs", Proc. IEEE International On-Line Testing Symposium, 2008, pp. 143-148

[10] http://www.simplescalar.com/

[11] E. Sanchez, M. Sonza Reorda, A. Tonda "On the Functional Test of Branch Prediction Units based on Branch History Table", 19th IFIP/IEEE International Conference on Very Large Scale Integration, 2011, pp. 278-283

Hierarchical Instruction Level Self Testing of Embedded Processor Cores

Parisa Sha'afi Kabiri, Zainalabedin Navabi
Electrical and Computer Engineering Department
School of Engineering Colleges, Campus 2
University of Tehran, Iran
{pani, navabi}@cad.ut.ac.ir

Abstract— **Embedded processors are often used in systems that are both safety-critical and long-living. Therefore testing of these devices is critical not only after production, but also in the field of application. Due to the limited accessibility of embedded processors, testing of SoCs is a major challenge in terms of ultra deep submicron issues. Scan based testing methods cannot be applied to embedded processor cores which cannot be modified to meet the design requirements for scan insertion. Generating test vectors for these high gate count devices is a major task. This paper presents a Hierarchical Instruction Level Self Testing (HILST) method which achieves stuck-at-fault coverage for an embedded processor core without the need for scan insertion. The test cost includes both the amount of test data and the duration of a test process. The proposed method (HILST) is well suited for meeting the challenges of testing SoCs which contain embedded processor cores. The HILST is high level in the sense that it is based on the knowledge of the Instruction Set Architecture (ISA) and Register Transfer Level (RTL) description of the processor and also our methodology is superior in terms of test development effort because it basically uses random test generation and improves random test set in such a way that significantly increases test generation quality.**

Keywords: **Instruction Level Test Generation, CPU Testing, Processor Testing, Hierarchical RTL Testing, Self Testing of Processors, Test Generation (TG).**

I. INTRODUCTION

Embedded processor cores are popular components used in design of many System-on-Chips (SoC). Every complex SoC generally includes a number of embedded processor cores which may be either general purpose or special purpose processors. Core based design has several advantages including design reuse and portability to different SoCs. This makes possible use embedded processors in a variety of applications in a cost effective manner. Testing of a system including embedded cores is a major challenge since access to these embedded processor cores from the pins of the SoC is impossible [1].

Scan based methods which deliver test patterns to the systems are quite effective, but because of limited accessibility of embedded cores for scan insertion, scan based methods cannot always be applicable to systems containing embedded processors, because it is required to isolate the cores from the remainder of system. From the other sight, the processor architecture itself may prohibit the use of scan test methods since processor designs may contain a mixture of latches and flip-flops, asynchronous logic, internal tri-states, and gated clocks that are becoming popular due to the need for low power design [2, 3]. These structures generally create many problems for scan based testing.

Additionally, the problem of at-speed testing arose because of the increasing gap between the operating frequencies of Automatic Test Equipment (ATE) and the operating frequencies of SoC, which leads to the escape of failures that can be detected only when testing is performed in the actual speed of the IC. The transfer of SoC test task from an external ATE to an internal Built-In-Self-Test (BIST) mechanism has significant advantages. Using self-test methodologies for processor testing results in yield loss reduction and drives down the overall test cost of the SoC [1] and also it allows actual at-speed testing. Self-test methodologies for embedded processor cores utilize the processor functionality and instruction set and do not add hardware or performance overheads in an optimized processor design. The HILST is a self-test methodology, which allows at-speed testing in contrast to testing through ATEs.

There is a long history on functional testing of microprocessors [5, 6, 7 and 8]. The obtained fault coverage is low in functional testing, because it does not consider the RTL structure of processor. More recently, some research has been initiated towards functional testing using some RTL information. However, these techniques are not exactly suited for embedded processor cores.

In [9, 10] functional self-testing approaches for processor cores are presented that rely on the use of random instruction sequences while [11, 12] rely on pseudorandom operations and operands. The problem in these methods is that, due to the use of a pseudorandom strategy both test size and test time are extremely large. The pseudorandom nature of that methodology leads in a large test generation code, large memory requirements and excessive test application time.

In [13] an Instruction Randomization Self Test is presented. The test methodology exploits the functional behavior of a microprocessor (e.g. the ability to execute instructions) to achieve stuck-at-fault coverage. In this

method, random stream of instructions are executed and the results are compressed by internal test hardware. A special software programs, written in the native assembler code of the processor, exploits the randomization behavior to achieve the observability and controllability needed for good fault coverage.

In [14] a structural testing methodology for processor cores is presented. In this methodology, pseudorandom pattern sequences are developed for each processor with regard to the instruction set constraints, based on the knowledge of the gate-level processor. At the test execution, pseudorandom test patterns are expanded by LFSR and then the pseudorandom test patterns are applied to the processors. This methodology is restricted by the need of gate-level details of the processor structure, since this information may not always be available, and also extracting the instruction set constraints is a very time consuming task, using this method may not always lead to an acceptable fault coverage.

In [15] a high level structural self-test methodology for embedded processor cores has been introduced. In this method, for each component operation a compact self-test routine is developed, based on the operations that the component performs, and then each operation is mapped to a processor instruction.

In this paper, we introduce a high level self-test methodology for embedded processor cores which is based on the knowledge of the processor ISA and its RTL description. Showing the connections among the functional components of the processor (ALUs, adders, multipliers, etc), the storage components (registers, register files, flags) and the steering logic components (multiplexers, bus elements) is a piece of information which is usually available in the RTL description and is much more easily manageable than a detailed gate-level Netlist. In this paper we present the application of the high level self-test methodology for embedded processor cores that uses as a case study the SAYEH processor core introduced in [16]. Then, providing detailed experimental results, we demonstrate that our methodology has significantly small self-test code size and memory requirements. Besides, the number of processor clock cycles required for execution of our test code is small. Reduced test code size and reduced test code execution time lead to a large total reduction in the overall test application time.

The rest of this paper is organized as follows; Section II introduces the HILST methodology. Section III presents the SAYEH processor as a case study in this paper. Experimental results are given in Sections IV. Conclusions are drawn in the last section.

II. HILST METHODOLOGY

HILST is based on the ISA and the RTL description of the processor core, is first developed in four steps:

A. *Step 1: Information Extraction*

From the processor instruction set architecture and RTL description, we extract the effects of each instruction execution on every component. Thus, for every instruction *I* and for every component *C* participating in the instruction execution the following information is extracted:

- ✓ The specific operation *O*, that the component performs along with the relevant control signals which are enabled by the control unit for the execution of this operation.

The extracted information from this step is used for mapping the component operations and related control signals to processor instructions.

This step of test methodology is done manually and required information by extracting information from ADL (Architecture Description Language) of the processor.

B. *Step 2: Test Generation*

In this step, four consecutive tasks must be done for each component of the processor core. The goal of this step is to generate a valid test set for each component to evaluate the functionality of the component.

It should be noted that the processor component can be as small as a Program Counter register or as large as the whole processor.

This step has four phases, which are described below. Each phase is done automatically.

1) Random Test Generation (RTG)

In this phase, a test set is generated for each component of the processor core to evaluate the functionality of each component. The test set is generated based on random test generation. Random test generation is the easiest form of test generation but it is not fully efficient way of test generating. In the experimental results, we will show that the HILST which is based on the RTG can reach to high fault coverage for processor components. The number of random test patterns applied to each component of processor is completely proportionate to the target fault coverage considered for that component of processor.

This phase is done automatically by using available commercial tools [18, 19].

2) Compress Random Generated Test Set

Memory size is one of the major concerns in designing embedded processor cores because it has an influence on power consumption and area, so as a contribution for the proposed method, all test set are compressed, this is more appropriate for sequential circuits.

As the HILST uses random test generation, so there are many test vectors that have not any effect in the evaluation of the circuits, and eliminating these test vectors from the test set has not any negative effect on the obtained fault coverage of the components, so after generating random test set, the set should be compressed.

In combinational circuits, the output is a pure function of the inputs, so when a test vector detects no faults from the circuit, it can be removed from the test set, but this rule is not applicable for sequential circuits. Because in contrast to combinational circuits, the sequential circuits are type of logic circuits whose

output depends not only on the present inputs but also on the history of inputs, so in case of sequential circuits, the test compression techniques should be applied to the circuits.

This phase is done automatically by using test compression method introduced in [20].

3) Remove Invalid Test Vectors

As we are going to test a processor by valid instructions, it is not avoidable to have invalid test vectors in the random generated test set, so these test vectors must be removed from the test set, and generally, this may lead to fault coverage reduction.

Removing invalid test vectors are done automatically according to the information extracted from step 1. With regard to the component operations and control signals issued by instruction execution, invalid test vectors can be recognized and removed from the test set.

4) Sequence Extraction

After obtaining the valid test set from previous phase, for each component the sequence of operations that the component performs during test is extracted. This sequence can be used in the next step, when we want to convert invalid test vectors to valid ones.

C. Step 3: Improve Test Generation

As mentioned before, a processor component can be a simple module such as Instruction Register or can be considered as a top level module which created by cascading and grouping the simple modules, such as DataPath, in higher level of abstraction, these modules hierarchically contain several simple modules. By considering this, processor components are classified in two groups, *simple modules* and *hierarchical modules*. Based on this classification, two different methods for improving the random generated test set proposed.

After removing all invalid test vectors from the test set, the fault coverage would be reduced. Depending on the component functionality and its design, this reduction may be excessive. For example, consider an embedded processor core that ALU control codes are one-hot, and ALU performs five different functions. A 5-bit control input of this unit selects its various operations. As this control signal is one-hot, so for performing an operation by ALU, in each test vector, just one bit of 5-bit control signal must be 1. With a 5-bit input, there are 32 different possible states, but just 5 of them are valid for the ALU unit. And because test generation is based on random test generation, so nearly 85% of test set is invalid.

1) Convert Invalid Test Vector to Valid One

To improve fault coverage of *simple modules* after removing invalid test vectors, invalid test vectors should be converted into valid test vectors.

This conversion should be done in a way that results in a better fault coverage. The problem is that as generally each component performs several operations, there are several options for replacing each invalid test vector with a valid one, and selecting among these options is an important issue. Randomly selecting one valid operation is a simplest form of replacing method, but it may not be a best solution. Checking all possible options is practically impossible, because applying a test vector to the processor will change processor internal states and it is imposible to return the processor state to the time before applying the test vector.

In the HILST proposed method, for replacing an invalid operation with a valid one, the valid operation will be selected with regard to two different information:

- ✓ *Type 1:* After RTG (first phase of step 2), for every component *C*, and every operation *O* performed by *C*, we extracted two types of information, percentage of using operation *O* for testing of component *C*, and also fault coverage obtained by executing operation *O* in component *C*. The second information is completely accurate for combinational circuits.
- ✓ *Type 2:* Use executed sequences of test vectors which are extracted from 4th phase of step 2. Considering this information is more suitable for sequential circuits.

By extracting the executed test vectors sequence in valid test set (Type 2) and also obtained fault coverage for the test sequence (Type 1), the best valid operation for replacing is selected and an invalid test vector will be replaced by a valid one. This process is done manually, but can be easily automated.

This method can also be used for enhancing the fault coverage of hierarchical modules, depending on the structure and design of these modules. But for hierarchical modules, in some cases, it is a time consuming and non efficient task to convert invalid test vectors to valid ones.

For example, consider *DataPath* of an embedded procesor as a component under test, during test of *DataPath*, because of its hierarchical structure that consists of several sub-components, the test vector for the *DataPath* consists of several multiple-bit control signals for sub-components, which determined that each sub-component should perform which operation during execution of a valid instruction in the processor. In this case, because of random nature of test generation, it is more predictable that the random generated control signals for sub-components of *DataPath*, have not been derived from a valid instruction execution in the processor, and also it is not efficient to convert invalid test vectors to valid ones, because of the number of sub-components in the *DataPath*, so another technique should be used for hierarchical modules with more sub-components or complicated structure.

2) Hierarchical Test Vectors Cascading

This method should be used for hierarchical modules with more sub-components or complicated structure.

According to our methodology, test generation are done for each component in the processor and the test set are extracted for each of them. For test of every hierarchical module, the procedure is to test sub-components by starting from the most hard-to-test and large sub-components of the hierarchical module. In this way for testing of hierarchical modules, sub-components with a high potential to activate multiple operations in other components during the application of instructions that test them must be considered first. It is a fact that during software testing of the hard to test sub-components, several other processor components are tested as well, thus eliminate the need for considering every individual sub-component in the test preparation process of hierarchical modules testing. This process is done automatically.

As mentioned in [15] these hard to test sub-components are generally the functional units. So the test set of these units should be converted to the corresponding hierachical module test set. This conversion should be done in a way that applying the converted test set to the hierarchical module, causes applying the corresponding test set to the sub-component.

D. *Step 4: Test Set Compression*

It should be noted that this step is completely similar to the test compression phase applied to the random generated test set in step 2, but in this step, the final improved test set for each component will be compressed.

This phase is done automatically by using test compression method introduced in [20].

III. CASE STUDY

In this section, we demonstrate the application of our methodology considering the SAYEH processor as presented in [16].

The CPU has a 16-bit data bus and a 16-bit address bus. The processor has 8 and 16-bit instructions. Short instructions contain shadow instructions, which can effectively pack two 8-bit instructions into a 16-bit word. The hex code *0F* is used as filler for the right most 8-bits of a 16-bit word that only contains an 8-bit instruction in its 8 left-most bits. The 16-bit instructions have the immediate field and the 8-bit instructions do not. Figure 1, shows SAYEH interface signals.

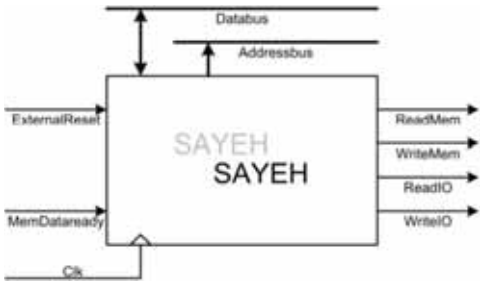


Figure 1 SAYEH interface [16]

The datapath of SAYEH is shown in Figure 2. The processor has a datapath and a controller. Datapath components are *Addressing Unit*, *IR* (Instruction Register), *WP* (window Pointer), *Register File* (RF), *ALU* (Arithmetic Logic Unit), and the *Status Register* (SR). The Addressing Unit is further partitioned into the PC and Address Logic. The Addressing Logic is a combinational circuit that is capable of adding its inputs to generate a 16-bit output that forms the address for the processor memory. Program Counter and Instruction Register are 16-bit registers. Register File is a two-port memory and a file of 64 16-bit registers. The Window Pointer is a 6-bit register that is used as the base of the Register File. Specific registers for read and write (R0, R1, R2 or R3) in the Register File are selected by its 4-bit input bus coming from the Instruction Register. Two bits are used to select a source register and other two bits select the destination register. When the Window Pointer is enabled, it adds its 6-bit input to its current data. The status register is a 2-bit register that saves the flag outputs of the Arithmetic Unit. The Arithmetic Unit is a 16-bit arithmetic and logic unit that has logical, shift, add and compare operations. *ALU* performs ten functions.

As shown in Figure 2, these components are either hardwired or connected through three-state busses. Component inputs with multiple sources use three-state busses. Three-state busses in this structure are Dastabus and OpndBus.

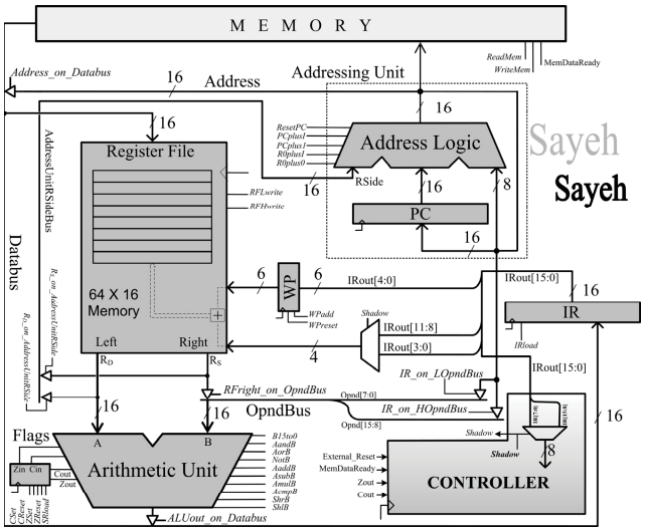


Figure 2 SAYEH DataPath [16]

Controller of SAYEH has eleven states for various reset, fetch, decode, execute, and halt operations. Signals generated by the controller control logic unit operations and register clocking in the datapath.

SAYEH instruction set includes most common instructions like load and store, arithmetic, logical operations; branch and jump instructions. It also supports direct and indirect addressing modes. SAYEH has a total of 29 different instructions.

Figure 3 shows the hierarchical structure of SAYEH components.

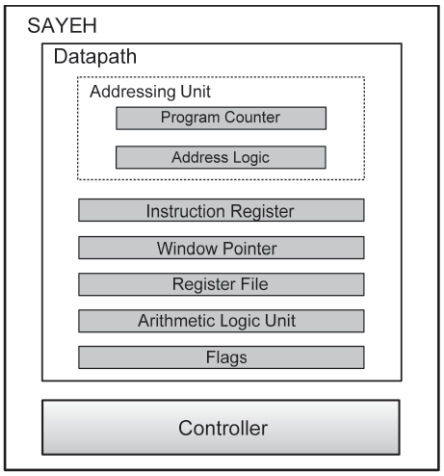


Figure 3 SAYEH Hierarchical Structure [16]

IV. EXPERIMENTAL RESULTS

In order to generate Netlist from the RTL Verilog-HDL code of SAYEH processor, *NetlistGen.exe* from [17] is used. For calculating the fault coverage for each component processor, we perform fault simulations with HOPE fault simulator [18]. HOPE is an efficient parallel fault simulator for synchronous sequential circuits.

Single stuck-at fault model is used for fault simulation and test generation. Results of Netlist generation for SAYEH processor is shown in Table I.

TABLE I. COMPONET'S SYNTHESIS RESULTS

Component name	Number of Pis	Number of Pos	Number of Gates	Number of DFFs	Number of Collapsed Faults
Address Logic	43	16	392	0	764
ALU	37	18	1383	0	3347
IR	17	16	85	16	198
PC	17	16	85	16	198
RF	28	32	6164	1024	18472
SR	7	2	35	2	60
WP	8	6	77	6	194
Addressing Unit	30	16	524	16	1037
DataPath	50	50	8974	1064	24247
Controller	19	38	1005	5	1697
SAYEH	17	36	9977	1069	25840

It should be noted in the Table I, PI and PO represent Primary Input and Primary Output, respectively. And the reported values show the number of required bits for PIs and POs of each component.

All four steps of HILST methodology which are introduced in section II are applied to the SAYEH processor. The following four sub-sections show the results of test methodology when applied to the SAYEH processor.

A. *Information Extraction*

According to Steps 1 of the proposed methodology we construct Table II, which shows the list of processor components and the operations that the components perform. List of processor components can be easily extracted by RTL description of processor, as shown in Figure 3. The list of operations that each component performs is easily extracted from its hardware description. After the identification of the set of operations O_C that component C performs we identify the set of processor instructions $I_{C,O}$ that execute operation O. Identification of the set of $I_{C,O}$ that performs operation O of component C can be extracted by ISA and RTL description of the processor.

TABLE II. COMPONENT OPERATION TO INSTRUCTION MAPPING

Component C	Operation O_C	Processor Instruction $I_{C,O}$
Address Logic	Reset	External Reset (it is not an instruction)
	PCPlusI	Brc, brz, jpr, spc
	PCPlus1	All instruction except {brc, brz, jpr, spc}
	RplusI	Jpa
	Rplus0	sta, lda, oup, inp
ALU	Pass B	sta, mvr, mih, mil, oup
	And	And
	Or	Orr
	Not B	Not
	Shift left B	shlB
	Shift right B	shrB
	Add	Add
	Sub	Sub
	Multiply	Mul
	Compare	Cmp
IR	Load IR	All instruction
PC	Load PC	All instruction
SR	Load carry and zero flag registers	cmp, mul, sub, add, shrB, shlB, notB, orr, and, inp, mvr, mih, mil
	Set Carry Flag	Scf
	Reset Carry Flag	Ccf
	Set Zero Flag	Szf
	Reset Zero Flag	Czf
WP	Reset	Cwp
	Add	Awp

B. *Test Generation*

In the second step, the test generation procedure should be performed for all components of processor, from Program Counter to whole of SAYEH as a processor. The results of random test generation step are shown in Table III.

It should be noted for sequential components, HOPE fault simulator [18] is used for test generation and for combinational components (components which have no Flip-Flops as mentioned in Table I), and the Atalanta [19] fault simulator is used. Atalanta is an automatic test pattern generator and fault simulator for stuck-at faults in combinational circuits that are developed by Virginia Polytechnic & State University. For both combinational and sequential circuits, test generation process is done based on random test generation.

It should be noted that the amount of test patterns applied to a component is measured when the fault coverage reaches a steady state i.e. applying more test causes no or negligible improvement in the fault coverage of the corresponding component.

TABLE III. COMPONENTS FAULT COVERAGE

Component	Fault Coverage	Number of test patterns applied
Address Logic	98.560 %	40
ALU	99.492 %	212
IR	100.000 %	32
PC	100.000 %	28
RF	100.000 %	2423
SR	98.333 %	53
WP	98.969 %	290
Addressing Unit	92.960 %	8643
DataPath	90.452 %	100000
Controller	87.153 %	1500
SAYEH *	**27.016 %**	500

As shown in Table III, the SAYEH processor that consists of all processor components has low fault detection capability, and also by increasing the number of applied test patterns by factor of 100 or 1000, the fault coverage will not be increased. This happens due to the existence of RPR (Random Pattern Resistant) faults in the SAYEH processor. For SAYEH processor, RPR faults are related to *RF*. *RF* is a memory-based component and consists of 64 16-bit register. Because of this, a test set can efficiently test the *RF*, when it considers all 64 registers in the *RF*. It is almost impossible for a test set, to use 64 different registers. Checking the undetected faults in the SAYEH processor indicates that nearly most undetected faults belong to the *RF*'s faults.

As we will show in the Table IV, the number of test patterns would be reduced by using compression techniques. The test compression method introduced in [20] is used as a test set compressor in the proposed HILST method.

TABLE IV. COMPONENTS FAULT COVERAGE AFTER TEST COMPRESSION

Component	Fault Coverage	Number of test patterns applied
Address Logic	-	-
ALU	-	-
IR	100.000 %	16
PC	99.495 %	14
RF	99.497 %	521
SR	98.333 %	12
WP	98.454 %	21
Addressing Unit	91.032 %	39
DataPath	88.390 %	8560
Controller	85.444 %	213
SAYEH	26.548 %	210

The compression method presented in [20] uses a selective random test generation technique implemented in an HDL environment. The method uses a statistical expectation graph and states of the sequential circuit for selecting the appropriate test vectors to achieve better fault coverage and a more compact test set. To further reduce the size of generated test set, the presented method [20] uses a static compaction method after the test generation process. This method just could be applied to the sequential circuits.

As shown in the Table IV, after test set compression, the size of test sets for each component is substantially decreased which may be led to a negligible reduction in fault coverage. As shown, number of test patterns reduced excessively and this achieved, just by 0.699% fault coverage reduction in average for each component.

The aim of HILST methodology is to test the processor via sequences of valid instructions, so in the next step, invalid test patterns should be removed from the test set. The fault coverage of each component after removal of invalid test vectors is shown in Table V.

TABLE V. COMPONENT'S FAULT COVERAGE AFTER INVALID TEST VECTORS REMOVAL

Component	Fault Coverage	Number of test patterns applied
Address Logic	86.780 %	29
ALU	98.656 %	205
IR	96.970 %	11
PC	74.242 %	7
RF	99.497 %	521
SR	70.000 %	7
WP	98.454 %	21
Addressing Unit	72.710 %	17
DataPath	59.805 %	2690
Controller	70.890 %	187
SAYEH	24.056 %	182

As shown in the Table V, after removal of invalid test vectors from the test set, the fault coverage would generally be decreased, this reduction is nearly 12.116% in average per each processor component.

C. *Improve Test Generation*

As mentioned previously in section II, all processor components are classified into two groups of simple modules and hierarchical modules. The definition of these modules is presented in step 3 of HILST methodology.

SAYEH simple modules (components) are *AddressLogic*, *ALU*, *IR*, *PC*, *SR*, *WP*, *AddressingUnit* and *Controller*. For these components, the method for improving the fault coverage converts the invalid test vectors to the valid ones.

As mentioned in step 3 of section II, two types of information are extracted to help in selecting a valid operation to substitute in the test set against an invalid operation for each component. By using the information extracted from the random test set with valid test vectors and also by getting inspiration from the test sequence extracted in step 2 (4^{th} phase) of the proposed test methodology, for each component, we are able to select a valid operation and replace the invalid test vector by this valid test vector. In some cases, for a multi-operation component, for replacing an invalid test vector, there are several options to select among them (for example *ALU* which performs several operations), the simulation results demonstrated that there is not any severe difference in fault coverage in selecting an operation among all the available operations, but in this case, we generally select the operation that has larger segment in the Netlist of the component, for example as the results

show, multiply instructions has better fault coverage than add instruction, because multiply has a larger circuit than add instruction in the Netlist of ALU component, so we prefer the multiply instruction.

DataPath and SAYEH are hierarchical modules in the processor, which their test set is produced by merging the sub-modules (simple modules) test set. At first functional units should be considered because these components are really hard to test components [15]. During test program execution of hard to test sub-components, other sub-components of hierarchical module are tested as well. This process will be continued until target fault coverage is achieved and thus it eliminates the need for considering every sub-component in the test preparation process.

The test set conversion of sub-component to the corresponding test set of hierarchical module should be done in a way that applying the converted test set to the hierarchical module, causes applying the corresponding test set to the sub-component.

TABLE VI. COMPONENT'S FAULT COVERAGE AFTER TEST GENERATION IMPROVEMENT

Component	Fault Coverage	Number of test patterns applied
Address Logic	97.513 %	28
ALU	99.671 %	206
IR	100.000 %	15
PC	74.242 %	7
RF	99.497 %	521
SR	90.000 %	13
WP	98.969 %	22
Addressing Unit	95.468 %	37
DataPath *	95.109%	2318
Controller	83.677 %	213
SAYEH *	93.205%	2531

As shown in the Table VI, the fault coverage for each component is substantially increased after test generation improvement step in the proposed methodology. Fault coverage enhancement is nearly 15.968% in average. And also it should be noted that for hierarchical modules (DataPath and SAYEH) the average fault coverage improvement is 52.406%, which demonstrates that the hierarchical test generation improvement that is based on merging the sub-module test sets, has a reasonable impact on fault coverage.

The main reason that causes a tremendous improvement in the fault coverage of DataPath and SAYEH components is that these components are two hierarchical modules, and as mentioned before the *RF*'s faults are mostly RPR faults. So the test set that fully test the *RF* has been incorporated in the enhanced test set of these modules, and because of this, detecting these faults increases the fault coverage considerably but also increases the number of test patterns.

Comparing the results of Table V with Table VI confirms that always all the undetected faults cannot be detected via valid instructions, this happens because there are some sections in the code of every processor which are *default cases* that related to the case when the processor faces with unexpected instructions, so the faults related to these segments of code cannot be detected by valid instructions.

D. Test compression

As the last step of our proposed method, again the compression method presented in [20] is applied to each component test set. But the main problem is that the introduced test compressor method [20] works based on random test generation and compress random generated tests, so we modified the test compressor to be able to compress deterministic test set. Table VII shows the result of fault coverage for each component of SAYEH processor after test compression.

As shown in the Table VII, after test set compression, the reduction in size of test sets for each component is negligible because all test vectors have an important effect in fault coverage. Number of test patterns is nearly reduced by 4.364% in average per every processor component; this was achieved just by 0.0439% fault coverage reduction in average.

TABLE VII. COMPONENT'S FAULT COVERAGE AFTER APPLYING COMPRESSION TO THE IMPROVED TEST SET

Component	Fault Coverage	Number of test patterns applied
Address Logic	97.513 %	28
ALU	99.671 %	206
IR	100.000 %	15
PC	74.242 %	6
RF	99.497 %	521
SR	90.000 %	11
WP	98.969 %	21
Addressing Unit	94.985 %	25
DataPath	95.109%	2318
Controller	83.677 %	197
SAYEH	93.205%	2515

It should be noted that the size of test set for SAYEH processor is 5.219 KByte that with regard to the processor fault coverage, it is a reasonable test size, and also the test application time for evaluating the functionality of SAYEH processor is 51.483 seconds that computed by HOPE fault simulator [18].

V. CONCLUSION

This paper introduces various concepts associated with the HILST and illustrates some of its advantages. HILST is a self test method which can be applied to microprocessor cores. The example shown demonstrates that a fairly high level of fault coverage can be achieved with no test hardware overhead on this particular design. There are several advantages of HILST that offers as a test methodology especially when dealing with processor cores.

Flexibility is a key advantage that HILST offers for testing processor cores. With on-chip memory, HILST provides many advantages associated with other BIST methods including at-speed fault validation. Many processor cores today are designed with deep submicron technology in order to enable high speed operation. With such designs, at-speed testing has become extremely important since some speed-sensitive faults cannot be detected with low frequency test vectors [21, 22]. HILST offers a true at-speed test since

valid instruction paths and combinations are exercised during the test execution.

A unique property of HILST is that the test logic is totally non-intrusive to the processor core. For this reason there is no need to redesign the core for insertion of scan chains, as is needed with scan methodologies. The HILST self test is naturally isolated from the processor core and other functional blocks which might reside on a silicon system. There is no hardware overhead for HILST compared to other BIST methods. By using HILST, the core suffers no loss in performance or increase in size due to the scan insertion.

While HILST is enabled, the processor will execute instructions on every clock cycle in a functional mode just as if it were executing real software programs. This differs from scan test methods since several clocks are needed to shift in serial patterns before a functional clock is provided. It is believed that this feature along with the fast execution of HILST might be a key advantage of HILST which results in potentially higher fault detection efficiency compared to scan based methods.

VI. REFERENCES

[1] Murray and J Hayes, "Testing ICs, getting to the core of the problem", *IEEE Design and Test of Computers*.Vol.29, no.11.

[2] J.V. Woods, P. Day, S.B. Furber, J.D. Garside, N.C. Paver, and S. Temple, "Amulet 1: An Asynchronous ARM Microprocessor", *Transactions of IEEE*. Vol. 46, no.4. 1997.

[3] B. Tuck, "Power Pushed to Prominence by System on Chip and Portable Products", *Computer Design*, January 1998.

[4] International Technology Roadmap for Semiconductors, 2001 Edition.

[5] D. Brahme and J. Abraham. "Functional Testing of Microprocessors", *IEEE Trans. on Computers*, C-33, 6, 1984.

[6] A. van de Goor et al. "Functional Testing of Current Microprocessors", *International Test Conference*, Sept. 1992.

[7] J. Lee and J. Patel. "An Instruction Sequence Assembling Methodology for Testing Microprocessors", *International Test Conference*, Sept. 1992.

[8] S. Thatte and J. Abraham. "Test Generation for Microprocessors", *IEEE Trans. on Computers* Vol. C-29, 1980.

[9] J.Shen, J.Abraham, "Native mode functional test generation for processors with applications to self-test and design validation", *in Proceedings of the International Test Conference* 1998, pp. 990-999.

[10] K.Batcher, C.Papachristou, "Instruction randomization self test for processor cores", *in Proceedings of the VLSI Test Symposium* 1999, pp. 34 – 40.

[11] F.Corno, M.Sonza Reorda, G.Squillero, M.Violante, "On the Test of Microprocessor IP Cores", *Design Automation & Test in Europe* 2001, Munich, Germany, March 2001.

[12] J.Rajski, J.Tyszer, "Arithmetic Built-In Self-Test for Embedded Systems", *Prentice Hall*, 1997.

[13] K. Batcher and C. Papachristou, "Instruction randomization self test for processor cores," *in Proc. IEEE VLSI Test Symp.*, 1999, pp. 34–40.

[14] Li Chen, S.Dey, "Software-Based Self-Testing Methodology for Processor Cores", *IEEE Transactions on CAD of Integrated Circuits and Systems*, vo.20, no.3, pp. 369-380, March 2001.

[15] N.Kranitis, A.Paschalis, D.Gizopoulos, Y.Zorian, "Effective Software Self-Test Methodology for Processor Cores", *Design Automation & Test in Europe* 2002, Paris, France, March 2002.

[16] Z. Navabi, *VHDL: Modular Design and Synthesis of Cores and Systems*, McGraw-Hill, Boston, 2007.

[17] Z. Navabi, *Digital System Test and Testable Design: Using HDL Models and Architectures*, Springer, 2010.

[18] H. K. Lee, D. S. Ha, "HOPE: An Efficient Parallel Fault Simulator for Synchronous Sequential Circuits", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 15, No. 9, September 1996.

[19] http://service.felk.cvut.cz/vlsi/prj/Atalanta-M/man.html, available on, Aug 2011.

[20] M. H. Haghbayan, S. Karamati, F. Javaheri, Z. Navabi "Test Pattern Selection and Compaction for Sequential Circuits in an HDL Environment", *IEEE Asian Test Symposium*, pp. 53-56, Dec. 2010.

[21] B. Nadeau, D. Burek, et al. "Scan-BIST, a Multifrequency Scan-Based BIST Method", *IEEE Design and Test of Computers.* Vol. 11, Number 1. Spring 1994.

[22] M. Hamad. "A Statistical Methodology for Modeling and Analysis of Path Delay Faults in VLSI Circuits". *Computers and Electrical Engineering*. Vol. 23, Number 5. September 1997.

An Approach to Hardware SAT Solvers for Test Generation Based on Instance Similarity

Tsuyoshi Iwagaki Fumiyuki Hafuri Kenji Ueda Toshiya Mukai Hideyuki Ichihara Tomoo Inoue
Graduate School of Information Sciences, Hiroshima City University
3-4-1 Ozuka-Higashi, Asaminami, Hiroshima 731-3194, Japan
E-mail: {iwagaki, ichihara, tomoo}@hiroshima-cu.ac.jp, {hafuri,ueda,mukai}@cd.info.hiroshima-cu.ac.jp

***Abstract*—This paper presents a test pattern generation (TPG) method based on Boolean satisfiability (SAT) which is intended to be implemented on hardware. In the proposed method, a solution to a SAT instance that expresses the TPG problem of a fault is reused to obtain a solution to that of another fault. This reuse can help to reduce unnecessary backtracking during search for the succeeding fault if the detection conditions of those two faults are similar. To maximize this benefit of reuse, a given fault list is arranged according to the similarity in terms of SAT clauses. Since the proposed method only manages the previous solution to process a current SAT instance, it is suitable for hardware implementation in a situation where memory resources are limited. Experimental results for some benchmark circuits show that the proposed method can reduce the overall test generation time.**

***Index Terms*—Test generation, Boolean satisfiability, hardware solver, solution reuse, instance similarity**

I. Introduction

Test pattern generation (TPG) is one of the challenging issues for today's VLSI-CAD. Among many TPG methods previously proposed, TPG based on Boolean satisfiability (*SAT-based TPG*) is a promising approach. It has been studied for the past twenty years and has still been studied [1].

In efficient SAT solvers, many techniques (e.g., Boolean constraint propagation (BCP) [2], non-chronological backtracking, conflict-based learning [3], etc.) are used to accelerate the decision process of solving a single SAT instance. On the other hand, for solving the sequence of SAT instances efficiently, the concept of *incremental SAT* has been introduced [4]. Given that a set of SAT clauses S is satisfiable, the satisfiability of $S \cup C$ is checked for a given set of new clauses C in the incremental SAT problem. This concept has been widely applied to specific problems of TPG and verification [5], [6], [7], [8], [9], [10].

To solve SAT instances more quickly, *hardware implementation of SAT solvers* have also been studied [11], [12], [13]. Hardware solvers can efficiently solve SAT instances due to its high-speed performance and high parallelism, even though it can treat only smaller instances than software solvers. To the best of our knowledge, existing hardware SAT solvers are for general-purpose use and not especially for TPG.

In this paper, we discuss a hardware-friendly technique for SAT-based TPG based on incremental SAT. We focus on the following characteristic of the SAT problem for TPG: SAT-based TPG for large-scale circuits must target numerous SAT instances but there is a kind of similarity among the SAT instances. As mentioned above, such characteristics can be successfully handled with incremental SAT. The technique that we discuss in this paper reuses a solution to a SAT instance that expresses the TPG problem of a fault in order to obtain a solution to that of another fault. This reuse can help to reduce unnecessary backtracking during search for the succeeding fault if the detection conditions of those two faults are similar. To maximize the benefit of reuse, we attempt to arrange a given fault list according to the similarity in terms of SAT clauses. Since the proposed method only manages the previous solution to process a current SAT instance, it is suitable for hardware implementation in a situation where memory resources are limited.

II. Basics of SAT-based TPG

Boolean satisfiability (SAT) is the problem of determining if the variables of a given Boolean formula can be assigned in such a way as to make the formula evaluate to true. In SAT-based TPG, the problem of finding a test pattern for a particular fault (e.g., stuck-at fault) is translated into a SAT instance. Such a SAT instance is usually given in conjunctive normal form (CNF).

To illustrate how SAT-based TPG is performed, we give the following example. For stuck-at fault f_1 shown in Figure 1, Figure 2 shows its miter circuit. Test vectors for f_1 are identical to input vectors of the miter circuit that set the output to one. This miter circuit can be transformed into the SAT instance shown in Figure 3 by means of a basic simple gate-to-clause transformation [14]. For example, the part surrounded with a dotted line in Figure 3 corresponds to the AND gate in the circuit of Figure 1. Figure 3 also shows a solution (truth assignment) of instance $I(f_1)$; this SAT instance is satisfied with this solution. The test vector for f_1 is $(A,B,C,E,H)=(0,1,0,0,0)$ which can be obtained by running a branch-and-bound algorithm on the search tree of the target SAT instance.

$(A1+\overline{A})\cdot(\overline{A1}+A)\cdot(A2+\overline{A})\cdot(\overline{A2}+A)$
$\cdot(D+\overline{B})\cdot(D+\overline{C})\cdot(\overline{D}+B+C)$
$\cdot(F+E)\cdot(F+A1)\cdot(\overline{F}+\overline{E}+\overline{A1})$
$\cdot(\overline{G}+\overline{A2})\cdot(\overline{G}+\overline{D})\cdot(G+A2+D)$
$\cdot(I+H)\cdot(\overline{I}+\overline{H})$
$\cdot(J+G)\cdot(J+I)\cdot(J+\overline{G}+\overline{I})$
$\cdot(B)\cdot(\overline{Bf})$
$\cdot(Df+\overline{Bf})\cdot(Df+\overline{C})\cdot(\overline{Df}+Bf+C)$
$\cdot(\overline{Gf}+\overline{A2})\cdot(\overline{Gf}+\overline{Df})\cdot(Gf+A2+Df)$
$\cdot(\overline{Jf}+Gf)\cdot(\overline{Jf}+I)\cdot(Jf+\overline{Gf}+\overline{I})$
$\cdot(XJ+J+\overline{Jf})\cdot(XJ+\overline{J}+Jf)$
$\cdot(\overline{XJ}+J+Jf)\cdot(\overline{XJ}+\overline{J}+\overline{Jf})$
$\cdot(XJ)$

$(A,A1,A2,B,C,D,E,F,$
$G,H,I,J,Bf,Df,Gf,Jf,XJ)$
$= (0,0,0,1,0,1,0,1,$
$0,0,1,0,0,0,1,1,1)$

Figure 3. Instance $I(f_1)$ for f_1 and a solution

$(A1+\overline{A})\cdot(\overline{A1}+A)\cdot(A2+\overline{A})\cdot(\overline{A2}+A)$
$\cdot(D+\overline{B})\cdot(D+\overline{C})\cdot(\overline{D}+B+C)$
$\cdot(F+E)\cdot(F+A1)\cdot(\overline{F}+\overline{E}+\overline{A1})$
$\cdot(\overline{G}+\overline{A2})\cdot(\overline{G}+\overline{D})\cdot(G+A2+D)$
$\cdot(I+H)\cdot(\overline{I}+\overline{H})$
$\cdot(J+G)\cdot(J+I)\cdot(J+\overline{G}+\overline{I})$
$\cdot(D)\cdot(\overline{Df})$
$\cdot(\overline{Gf}+\overline{A2})\cdot(\overline{Gf}+\overline{Df})\cdot(Gf+A2+Df)$
$\cdot(\overline{Jf}+Gf)\cdot(\overline{Jf}+I)\cdot(Jf+\overline{Gf}+\overline{I})$
$\cdot(XJ+J+\overline{Jf})\cdot(XJ+\overline{J}+Jf)$
$\cdot(\overline{XJ}+J+Jf)\cdot(\overline{XJ}+\overline{J}+\overline{Jf})$
$\cdot(XJ)$

$(A,A1,A2,B,C,D,E,F,$
$G,H,I,J,Df,Gf,Jf,XJ)$
$= (0,0,0,1,1,1,0,1,$
$0,0,1,0,0,1,1,1)$

Figure 4. Instance $I(f_2)$ for f_2 and a solution

$(A1+\overline{A})\cdot(\overline{A1}+A)\cdot(A2+\overline{A})\cdot(\overline{A2}+A)$
$\cdot(D+\overline{B})\cdot(D+\overline{C})\cdot(\overline{D}+B+C)$
$\cdot(F+E)\cdot(F+A1)\cdot(\overline{F}+\overline{E}+\overline{A1})$
$\cdot(\overline{G}+\overline{A2})\cdot(\overline{G}+\overline{D})\cdot(G+A2+D)$
$\cdot(I+H)\cdot(\overline{I}+\overline{H})$
$\cdot(J+G)\cdot(J+I)\cdot(J+\overline{G}+\overline{I})$
$\cdot(H)\cdot(Hf)$
$\cdot(If+Hf)\cdot(\overline{If}+\overline{Hf})$
$\cdot(\overline{Jf}+G)\cdot(\overline{Jf}+If)\cdot(Jf+\overline{G}+\overline{If})$
$\cdot(XJ+J+\overline{Jf})\cdot(XJ+\overline{J}+Jf)$
$\cdot(\overline{XJ}+J+Jf)\cdot(\overline{XJ}+\overline{J}+\overline{Jf})$
$\cdot(XJ)$

$(A,A1,A2,B,C,D,E,F,$
$G,H,I,J,Hf,If,Jf,XJ)$
$= (0,0,0,0,0,0,0,1,$
$0,0,1,1,1,0,0,1)$

Figure 5. Instance $I(f_3)$ for fault f_3 and a solution

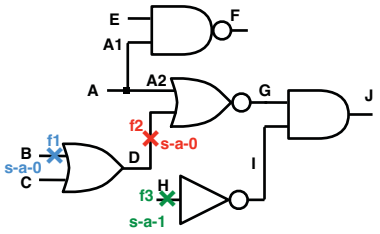


Figure 1. Circuit and three faults

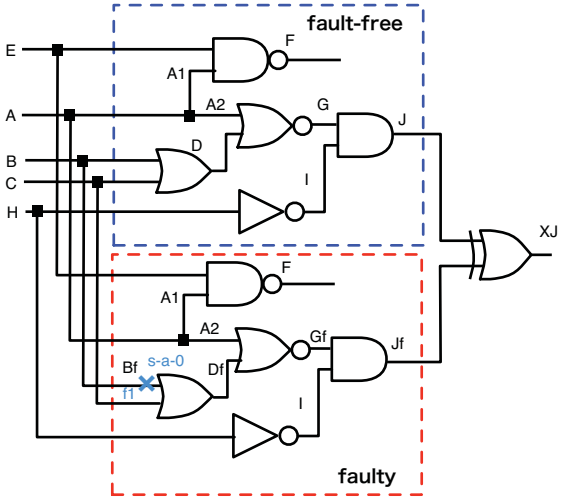


Figure 2. Miter circuit for fault f_1

Note that one fault produces one SAT instance; if a given circuit has numerous faults, basically we must solve numerous instances.

III. Hardware SAT solver

A hardware SAT solver is a system including hardware (FPGAs in many cases) implementing algorithms to solve SAT instances [11], [12], [13]. An FPGA-based approach [12] can accelerate the SAT solution process with the goal of speedily solving large SAT instances in a scalable fashion. The FPGA system consists of an FPGA board implementing a SAT engine to solve a given SAT instance and a memory storage to keep up not-yet-solved SAT instances. To accelerate solving a SAT instance, this system partitions a given SAT instance into small sub-instances properly, and it attempts to satisfy each sub-instance quickly by taking advantage of hardware. The sub-solutions for sub-instances are shared with a global memory. Moreover, to reduce the influence of the backtracks among sub-instances, the authors proposed a technique to arrange the order of sub-instances.

The SAT engine in the above system solves SAT instances one-by-one: it picks one instance from the memory storage and solves it, and then, it fetches the next SAT instance. When the system solves every instance, it halts.

IV. Proposed SAT-based TPG

In this work, we present a method to accelerate performing SAT-based TPG on the hardware SAT solver [12]. We aim at improving the effectiveness of the hardware SAT solver by exploiting the similarity of the SAT instances for TPG.

A. TPG reusing previous solutions

In general, it frequently occurs that a test vector for a fault is similar with that for another fault. In particular, this tendency may increase if the two faults are close to each other. For example, in the circuit of Figure 1, faults f_1 and f_2 are on one of the inputs and output of the OR gate, respectively. For these faults, a test vector of fault f_1 is (A,B,C,E,H) = (0,1,0,0,0) and f_2 is (0,1,1,0,0); they are very similar. This means that the SAT instances corresponding to these faults, $I(f_1)$ and $I(f_2)$, have some similar solutions, as shown in the bottom of Figures 3 and 4.

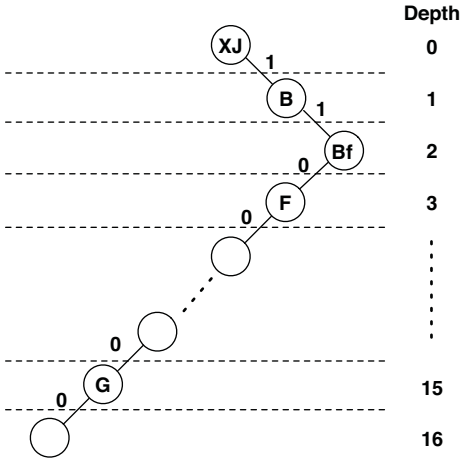


Figure 6. Search tree for the solution of Figure 3

From the above observation, we propose to reuse a solution of a SAT instance as the initial assignment of the successive SAT instance. Given a SAT instance, the procedure of reusing a solution works as follows.

1) Assign the appropriate value for each unit clause containing just one literal and perform the Boolean constraint propagation in the given SAT instance.
2) Perform Step 3) until an inconsistency occurs.
3) Assign the given solution to each variable of the current SAT instance in ascending order of its depth in the search tree and perform the Boolean constraint propagation.
4) Perform the normal branch-and-bound search.

Note that if the value of a variable for reuse is specified in Step 1), this variable is discarded in Step 3).

Here, let us suppose that two SAT instances $I(f_1)$ and $I(f_2)$ are solved in this order. Assume that the solution shown in Figure 3 of instance $I(f_1)$ and the search tree shown in Figure 6 are generated. If we solve instance $I(f_2)$, reusing of $XJ = 1$, $B = 1$, $F = 0, \ldots, G = 0$ occur in this order. Note that $B_f = 0$ is not reused because $I(f_2)$ does not include B_f. In this case, no backtracking occurs during search for $I(f_2)$. In contrast, if $I(f_2)$ is solved without solution reuse, backtracking can occur five times. In this way, the technique of the solution reuse can reduce the number of backtracks during search if two SAT instances have similar solutions.

We consider that the above reuse approach is suitable for the hardware SAT solver [12]. This is because the hardware attempts to solve instances one-by-one, and it has a global memory to share sub-solutions by sub-instances, so that the global memory can be also used for sharing solutions by SAT instances.

B. Ordering of SAT instances

In general, there is a possibility to increase the number of backtracks if the previous solution is not beneficial for the SAT instance under consideration. To estimate whether the previous solution is useful, we define the similarity $C(i,j)$ of two SAT instances I_i, I_j as follows:

$$C(i,j) = \frac{|S_i \cap S_j| \times 2}{|S_i| + |S_j|}$$

where S_i (resp. S_j) is the set of clauses in instance I_i (resp. I_j). Note that we assume that if the similarity between two SAT instances is high, their solutions can be very similar. Here, consider the similarity $C(f_1, f_2)$ of the instance $I(f_1)$ in Figure 3 and the instance $I(f_2)$ in Figure 4. In $I(f_1)$, there are 34 clauses, and $I(f_2)$ has 31 clauses. Since the number of common clauses is 29, the similarity $C(i,j)$ of these two instances is calculated as

$$C(f_1, f_2) = \frac{29 \times 2}{34 + 31} \approx 0.89.$$

We consider the following problem to maximize the benefit of solution reuse: Given a set of SAT instances $\mathcal{S}$, determine a permutation π of $\{1, 2, \ldots, |\mathcal{S}|\}$ in such a way that the following objective function is maximized:

$$\sum_{i=1}^{|\mathcal{S}|-1} C(\pi(i), \pi(i+1)).$$

This problem can be viewed as a kind of the traveling salesperson problem [15] on a weighed complete graph where each vertex v_i represents a SAT instance I_i and the weight associated with an edge (v_i, v_j) is the similarity between instances I_i, I_j. A permutation corresponds to the order of solving SAT instances. In this work, we tackle the above problem by using a modified version of the nearest neighbor heuristics [15].

Figure 7 shows an example of a weighed complete graph for four SAT instances to determine the order of those instances. First, the edge with the largest weight (similarity) is identified and the two vertices of the edge are selected (visited) as the first vertex and the second one in our ordering algorithm. In Figure 7, vertices B and C are selected in this order because 0.95 is the largest similarity. Note that the order between the two vertices selected first is randomly determined, i.e., the order of $C \to B$ is possible. Next, the second vertex is marked as the current one and the largest edge connecting the current vertex (C) to an unvisited vertex is found out. As a result, vertex A is selected and the current vertex is changed to A. This process is repeated until all the vertices are selected. Finally, the order of $B \to C \to A \to D$ is obtained.

In the case of the three instances $I(f_1), I(f_2), I(f_3)$ shown in Figures 3–5 where $C(f_1, f_2) \approx 0.89$, $C(f_1, f_3) \approx 0.72$ and

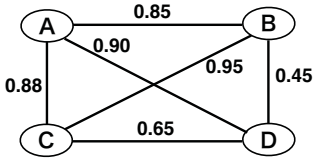


Figure 7. Weighted graph for ordering

$C(f_2, f_3) \approx 0.75$, the order of $(f_1) \rightarrow I(f_2) \rightarrow I(f_3)$ is created by our ordering algorithm. If we perform TPG in that order, the number of backtracks can be 11. On the other hand, if the order of $(f_1) \rightarrow I(f_3) \rightarrow I(f_2)$ is used for TPG, the number of backtracks can be increased to 15. From this fact, we believe that instance similarity is a good estimator to make TPG more efficient.

V. Experiments

In the following experiments, we evaluate the effectiveness of the proposed method for the combinational parts of small ISCAS '89 benchmark circuits (s27, s208.1 and s298). The SAT solver used in the experiments was implemented in C language that has the capability of Boolean constraint propagation [2] and non-chronological backtracking [3], and it ran on a Macintosh computer (CPU: Intel Core 2 Duo 2.26 GHz, Memory: 4 GB). All the stuck-at faults in each benchmark circuit were targeted for test generation and fault collapsing and fault simulation were not performed in the experiments.

To demonstrate how the proposed method is effective, three options were considered for evaluation as shown in Table I. The first option (Reuse, Ordering) = (OFF, OFF) means that each SAT instance is solved without the reuse of solutions and the ordering based on the instance similarity. In the second option (Reuse, Ordering) = (ON, OFF), the reuse of solutions is considered but the fault list is not arranged based on the instance similarity. The third option (Reuse, Ordering) = (ON, ON) corresponds to the proposed approach.

Table I shows that the third option (our approach) outperformed the other options in terms of processing time for all the three circuits. Note that all the instances were satisfied in the experiments, i.e., there are no redundant faults in the circuits. This performance gain was mainly due to the reduction of backtracking during search. By comparing the proposed approach with the second option, we can see that the ordering of SAT instances is important and effective to reduce the processing time if solutions are reused.

As shown the above, our approach can reduce the overall test generation time. We analyze the experimental results to find out a possibility of further improvement of our approach in term of instance similarity here. This analysis was done for specific instances $f_{a1}, f_{b1}, f_{c1}, f_{d1}$ in s208.1 and s298 as shown in Table II. In the case of f_{a1}, similarity scores, which are very high, are the same for f_{a0} and f'_{a0} and solutions for f_{a1} are generated fast by reusing their solutions. In contrast, although f_{b1} has a larger similarity score for f'_{b0} compared to that for f_{b0}, reusing the solution of f'_{b0} increases the processing time of f_{b1}. This implies that the metrics of instance similarity is not useful for this specific case. Furthermore, f_{d1} has a relatively low similarity score for f_{d0}. However, the solution of f_{d0} makes the processing time for solving f_{d1} faster. From the above analysis, we can see that more sophisticated metrics for solution reuse should be discussed to improve our approach in the future.

VI. Conclusions and Future Work

In this paper, we proposed a SAT-based test generation technique which is intended to be implemented on a hardware solver like [12]. The solution reuse technique can work well for the fault list arranged by our ordering algorithm based on instance similarity although there is room for improving its metrics. Our future plans are to conduct extensive experiments for large benchmark circuits and to investigate a search technique suitable for solution reuse.

Acknowledgments

This work was supported in part by the Grant-in-Aid for Young Scientists (B) (22700049) of the Japan Society for the Promotion of Science (JSPS) and Hiroshima City University (HCU) under the HCU Grant for Special Academic Research.

References

[1] R. Drechsler, S. Eggersglüß, G. Fey and D. Tille, *Test pattern generation using Boolean proof engines*, Springer, 2009.

[2] M. Moskewicz, C. Madigan,Y.Zhao, L. Zhang and S. Malik, "Chaff: Engineering an efficient SAT solver," *Proc. Design Automation Conf.*, pp. 530–535, 2001.

[3] J. Marques-Silva and K. Sakallah, "GRASP: A search algorithm for propositional satisfiability," *IEEE Trans. on Comp.*, Vol. 48, No. 5, pp. 506–521, 1999.

[4] J. N. Hooker, "Solving the incremental satisfiability problem," *Journal of Logic Programming*, Vol. 15, pp. 177–186, 1993.

[5] J. Kim, J. Whittemore, J. P. Marques-Silva, and K. Sakallah, "On applying incremental satisfiability to delay fault testing," *Proc. Design, Automation and Test in Europe*, pp. 380–384, 2000.

[6] O. Shtrichman, "Pruning techniques for the SAT-based bounded model checking problem," *Proc. Correct Hardware Design Verification Methods*, LNCS 2144, pp. 58–70, 2001.

[7] J. Whittemore, J. Kim and K. Sakallah, "SATIRE: A new incremental satisfiability engine," *Proc. Design Automation Conference*, pp. 542–545, 2001.

[8] G. Fey, T. Warode and R. Drechsler, "Reusing learned information in SAT-based ATPG," *Proc. International Conf. on VLSI Design*, pp. 69–74, 2007.

[9] S. Disch and C. Scholl, "Combinational equivalence checking using incremental SAT solving, output ordering, and resets," *Proc. Asia and South Pacific Design Automation Conference*, pp. 938–943, 2007.

[10] D. Tille, S. Eggersglus and R. Drechsler, "Incremental solving techniques for SAT-based ATPG," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 29, No. 7, pp. 1125–1130, 2010.

TABLE I
TEST GENERATION RESULTS

Circuit	# of faults	(Reuse, Ordering)	# of backtracks	Processing time [s]
s27	52	(OFF, OFF)	8	0.001
		(ON, OFF)	3	0.001
		(ON, ON)	2	**0.000**
s208.1	436	(OFF, OFF)	75,927	33.611
		(ON, OFF)	50,578	28.989
		(ON, ON)	13,935	**8.959**
s298	596	(OFF, OFF)	359,759	370.731
		(ON, OFF)	589,950	587.651
		(ON, ON)	310,224	**318.457**

TABLE II
THE RELATIONSHIP BETWEEN SIMILARITY AND PROCESSING TIME FOR SPECIFIC INSTANCES

Circuit	(Reuse, Ordering)	Similarity	# of backtracks	Processing time [s]
s208.1	(OFF, OFF)	N/A	0	0.659
	(ON, OFF)	$C(f_{a0}, f_{a1}) = 0.994$	0	0.002
	(ON, ON)	$C(f'_{a0}, f_{a1}) = 0.994$	0	0.002
	(OFF, OFF)	N/A	4,225	1.553
	(ON, OFF)	$C(f_{b0}, f_{b1}) = 0.945$	0	0.002
	(ON, ON)	$C(f'_{b0}, f_{b1}) = 0.986$	7,681	5.320
s298	(OFF, OFF)	N/A	6,097	9.547
	(ON, OFF)	$C(f_{c0}, f_{c1}) = 0.995$	0	0.004
	(ON, ON)	$C(f'_{c0}, f_{c1}) = 0.997$	0	0.004
	(OFF, OFF)	N/A	287	0.304
	(ON, OFF)	$C(f_{d0}, f_{d1}) = 0.673$	0	0.005
	(ON, ON)	$C(f'_{d0}, f_{d1}) = 0.996$	0	0.004

[11] I. Skliarova, "Reconfigurable hardware SAT solvers: A survey of systems," *IEEE Trans. on Comp.*, Vol. 53, No. 11, 2004.

[12] K. Gulati, S. Paul, S. P. Khatri, S. Patil and A. Jas, "FPGA-based hardware acceleration for Boolean satisfiability," *ACM Trans. on Design Automation of Electronic Systems*, Vol. 14, No. 2, 2009.

[13] M. Safar, M. Shalan, M. El-Kharashi and A. Salem, "A novel conflict directed jumping algorithm for hardware-based SAT solvers," *Proc. International Design and Test Workshop*, pp. 103–108, 2010.

[14] T. Larrabee, "Test Pattern Generation Using Boolean Satisfiability," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 11, No. 1, pp.4–15, 1992.

[15] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan and D. B. Shmoys, *The traveling salesman problem: A guided tour of combinatorial optimization*, Wiley, 1985.

Reconfigurable Array-Based Area-Efficient Test Structure for Standard Cell Characterization

Bishnu Prasad Das[1] and Hidetoshi Onodera[1,2]
[1] Graduate School of Informatics, Kyoto University, Japan
[2] JST, CREST
bishnu.iisc@gmail.com and onodera@vlsi.kuee.kyoto-u.ac.jp

***Abstract*—Any RTL/digital design highly depends on the quality of the standard cell library. In this paper, all-digital array based test structure is presented to test the quality (i.e. functionality and performance) of all types of logic gates in the standard cell library using the reconfigurable array of gate delay measurement cell (GDMC). The reconfigurable array based test structure can easily isolate a faulty standard cell from non-faulty standard cell. The test structure is area efficient with a saving of 2× area compared to the normal ring oscillator based delay measurement. The gate delay is estimated using the least square method with measured ring oscillator period/frequency. As the least square method averages out the random noise in the measured ring oscillator period, measured gate delay is estimated accurately. Fifty standard cells were tested using this reconfigurable array structure. A test chip has been fabricated in an industrial 180nm technology node to study the feasibility of the approach.**

***Index Terms*—On-chip standard cell characterization, Reconfigurable ring oscillator, Delay measurement, Least square method.**

I. Introduction

Standard cells are the basic building blocks of the Register Transfer Language(RTL) or digital design flow. The performance of the digital circuit design completely depends on the accuracy of the designed standard cells. Hence, standard cells are tested on silicon before it would be used by the circuit designers. Hence, there is a need of an area-efficient circuit technique for the on-chip testing of standard cells as the number of standard cells are huge.

Traditionally, the propagation delay of standard cell is measured with ring oscillator (RO) because of its easy on-chip implementation and high sensitivity [1], [2], [3], [4] to process variation. Normally, the delay of a cell is quite small; delay can be increased by cascading a number of stages and the delay of the individual cell can be found out by averaging. When the number of standard cells are large, then there is a requirement for an area-efficient test structure. Each ring oscillator requires an enable circuit to control its functionality. The enable circuit is a two input NAND gate with one input in the path of ring oscillator and other as the control input. In one hand, ring oscillator with small number stages is area-efficient. However, the enable circuit used to control the ring oscillator, impacts heavily on the accurate performance characterization in case of ring oscillator with small number of stages. Therefore, the ring oscillator with large number of stages are needed to mitigate the impact of the enable circuit which in turn increases the area of the test structure.

Picosecond imaging circuit analysis (PICA) [5] is used to obtain quantitative delay information by counting the number of infra-red photons from the back side of a thinned package chip. However, the technique requires infra-red measurement which makes it costly and tedious for large scale standard cell characterization.

Jitter measurement using vernier ring based time-to-digital converter was presented by [6]. Delay measurement using delay lock loop (DLL) was proposed by [7]. Even this solution looks promising in measuring to sub-nanosecond resolution of the cells under test. However, the complexity of the on-chip measurement circuitry limits its large scale implementation which will be needed for large scale standard cell characterization.

The standard cell characterization using random sampling technique is proposed in [8]. In this technique, two samplers are used to sample the input and output of the cell. Due to local process variation [9], the two samplers may suffer from variation and hence the measurement would be inaccurate. The standard cell delay characterization using on-chip sampling oscilloscope is presented in [10]. However, the test structure requires analog comparator for delay measurement. Hence, the accuracy of the gate delay would be limited by the process variation [9] in comparator.

The individual gate delay measurement using reconfigurable ring oscillator is presented in [11], [12]. The present work is based on [12] to test all types of logic gate in standard cell library. We propose an area efficient reconfigurable array structure to characterize standard cells in silicon. In [12], the gate delay is estimated by solving a set of linear equations. However, it leads to inaccuracy in gate delay estimation if there is random noise in the ring oscillator period/frequency measurement due to power supply fluctuation. In this work, the system of linear equations are solved using least square method which averages out the random noise in the

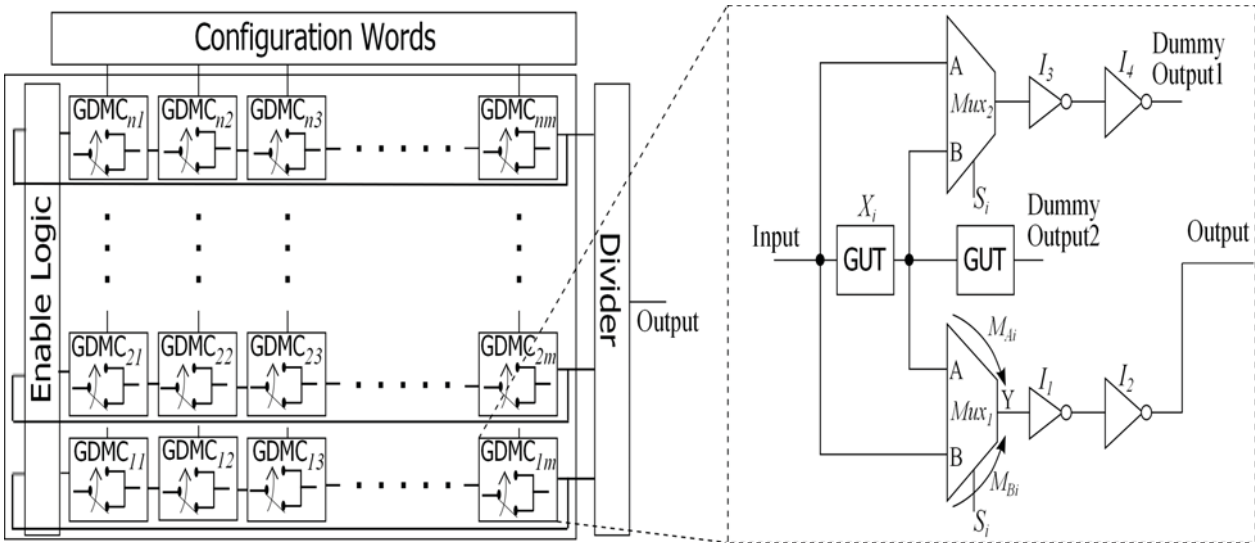


Fig. 1. Reconfigurable array structure for standard cell characterization with gate delay measurement cell (GDMC)

measurement. The major contributions of this work is as follows:

- Reconfigurable array based test structure for standard cell testing.
- Quantification of the area efficiency of the test structure for all types of standard cells characterization.
- Usage of least square method for calculation of gate delay to avoid random noise in measurement.
- Validation of the proposed concept in silicon.

II. Reconfigurable Array Test Structure

The reconfigurable array structure is an array of gate delay measurement cells (GDMC) as shown in Fig. 1. It is area-efficient compared to normal ring oscillator based structure [1], [2], [3], [4]. Each GDMC is the unit cell of the array which contains the gate of the standard cell library. The reconfigurable array based test structure can easily isolate a faulty standard cell from non-faulty standard cell. The test structure consists of 24 reconfigurable ring oscillators [12] with fifty different standard cells. The gates of the standard cells are arrange in a matrix format as shown in Fig. 1. The enable logic controls the functionality of the ring oscillators. The configuration word determines the number of stages of each ring oscillator. With the help of the enable logic and the configuration words, each of the GDMCs in ring oscillator can be accessed and finally characterized. The output of of the ring oscillator is divided to slow down the frequency of oscillation.

A. Gate Delay Measurement Cell

The gate delay measurement cell (GDMC) used in this work is taken from [12] and its schematic is shown in Fig. 1. This cell consists of inverters and two multiplexers along with gate under test (GUT). The GUT X_i is the gate whose delay is to be measured. Another dummy GUT and the inverters I_1, I_2, I_3 and I_4 are used for load matching and buffering respectively. The multiplexer Mux_1 allows bypassing of the logic gate, thus enabling the calculation of the gate's delay by the difference of two period measurements of the ring oscillator. Thus, the calculated delay is the sum of the delay of the logic gate and the difference in path delays of the multiplexer Mux_1 (i.e between input A to output Y and input B to output Y) due to unequal slew input to two inputs of Mux_1 for the two different mux settings. Symmetric multiplexers of large size which have balanced delays between input A to output Y and input B to output Y are used to reduce any systematic mismatches. If any standard cell is faulty due to the limitation of the fabrication process, then multiplexer in the GDMC helps in isolating the faulty cell from a non-faulty cell.

III. Delay Measurement using Least Square Method

For simplicity of description, we have considered a 4-stage ring oscillator with enable circuit as shown in Fig. 2. However, the technique is applicable for any number of stages of the ring oscillator. With this circuit setup, it is shown that delay of four GUTs can be measured.

A. Gate delay measurement procedure

Gate delay measurement procedure is presented with the help of a 4-stage ring oscillator as shown Fig. 2. We have used the least square method on measured ring oscillator period to estimate the gate delay.

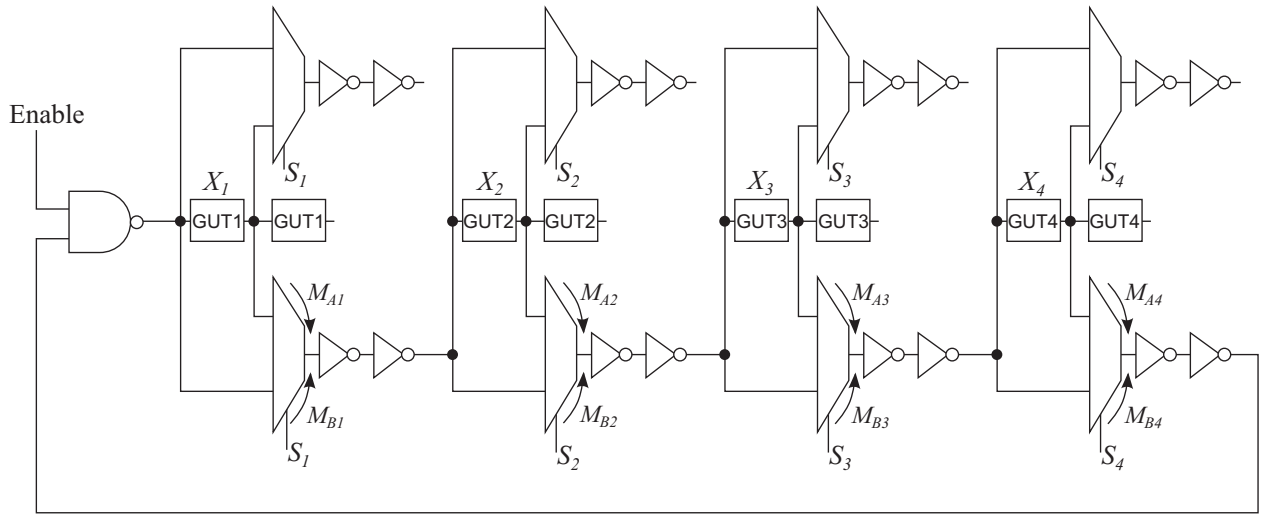


Fig. 2. Gate delay measurement using reconfigurable ring oscillator with four gates under test

Here X_i is the average of rise and fall delay of the i^{th} GUT, and M_{Ai} and M_{Bi} are the delays between inputs A and B to output Y of the multiplexer Mux_1 respectively. K is the delay of the rest of the elements of the ring oscillator such as the 2-input NAND gate as enable logic, the buffers consisting of inverters I_1, I_2 of all the cells, the wires etc. Let S be the configuration vector or status vector consisting of 4 bits. If S_i = 1, then input A to output Y of the multiplexer Mux_1 of the i^{th} stage GDMC is selected. If S_i = 0, then input B to output Y of the multiplexer Mux_1 of the i^{th} stage GDMC is selected. T_1, T_2, T_3, T_4, T_5, T_6, T_7 and T_8 are the period of RO for eight configuration words S := {1111, 0000, 0011, 0101, 1001, 0110, 1010, 1100} respectively. Since, the least square method is used for delay estimation, we have taken all possible combinations of configuration vectors or status vectors.

When select status vector S = 1111, then the input A to output Y of stages 1st, 2nd, 3rd and 4th are connected. Then,

$$\sum_{i=1}^{4}(X_i + M_{Ai}) + K = T_1/2 \tag{1}$$

When select status vector S = 0000, then the input B to output Y of stages 1st, 2nd, 3rd and 4th are connected.

$$\sum_{i=1}^{4} M_{Bi} + K = T_2/2 \tag{2}$$

When select status vector S = 0011, then the input A to output Y of stages 3rd and 4th are connected and the input B to output Y of stages 1st and 2nd are connected.

$$\sum_{i=3}^{4}(X_i + M_{Ai}) + \sum_{i=1}^{2} M_{Bi} + K = T_3/2 \tag{3}$$

When select status vector S = 1100, then the input A to output Y of stages 1st and 2nd are connected and the input B to output Y of stages 3rd and 4th are connected.

$$\sum_{i=1}^{2}(X_i + M_{Ai}) + \sum_{i=3}^{4} M_{Bi} + K = T_8/2 \tag{4}$$

Similarly, the linear equations are formed for other status vectors. Eight linear equations can be written using eight status vectors. These linear equation can be expressed in matrix form as follows:

$$AZ = B. \tag{5}$$

where A is a 8×9 matrix, B is a 8×1 column vector and Z is a 9×1 column vector whose delay will be estimated. These are defined as follows:

$$A = \begin{vmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{vmatrix},$$

$$Z = \begin{vmatrix} X_1 + M_{A1} \\ X_2 + M_{A2} \\ X_3 + M_{A3} \\ X_4 + M_{A4} \\ M_{B1} \\ M_{B2} \\ M_{B3} \\ M_{B4} \\ K \end{vmatrix} \quad and \quad B = \begin{vmatrix} T_1/2 \\ T_2/2 \\ T_3/2 \\ T_4/2 \\ T_5/2 \\ T_6/2 \\ T_7/2 \\ T_8/2 \end{vmatrix} \tag{6}$$

Equation 5 is solved using the linear least square method to find out 9 unknowns in Z column vector. The sum of the

delay of the gate under test and the input A to output Y of the multiplexer (i.e. $(X_i + M_{Ai})$ for i^{th} stage) is estimated by this method. It also calculates the delay of input B to output Y of the multiplexer (i.e. M_{Bi} for i^{th} stage). During design of the circuit, the delay of input A to output Y of the multiplexer M_{Ai} are made equal to the delay of input B to output Y of the multiplexer M_{Bi}. Hence, the delay of the gate under test X_i can be evaluated by subtracting $(X_i + M_{Ai})$ from M_{Bi}. The measured gate delay is given by

$$Measured\ i^{th}\ stage\ gate\ delay\ =\ X_i + (M_{Ai} - M_{Bi}) \quad (7)$$

Symmetrical multiplexer is used to make the input A to output Y of the multiplexer M_{Ai} same as the input B to output Y of the multiplexer M_{Bi}. From eqn. 7, delay of the i^{th} stage gate is measured.

As least square method is employed for delay estimation, the effect of random noise in measurement of the ring oscillator period is minimized. The K is constant for all equations which includes the delay of the rest of the elements in the ring oscillator such as the 2-input NAND gate as enable logic, the buffers consisting of inverters I_1, I_2 of all the cells, the wires etc. Hence, there is no impact of enable circuit on gate delay measurement.

For simplicity of explanation of least square, we have used 4-stage reconfigurable ring oscillator. In this case, the number of variables are 9 and number of equations are 8 which would be under-determined. However, if 6-stage reconfigurable ring oscillator is considered, then the number of variables are 13 and number of equations are 32 (i.e. C(6,0) + C(6,2) + C(6,4) + C(6,6)= 32). Here, the number of equations are greater than number of variables which is over-determined. Hence, least square will produce better results with more equations.

IV. Measured Results

A test chip has been fabricated in 180nm CMOS process to validate the proposed approach. We have implemented around fifty types of standard cells (i.e. both inverting and non-inverting cells) using this reconfigurable array test structure. The gate delay of the standard cells are estimated using the least square method as explained in Section III. The test chip layout showing the reconfigurable array of standard cell is presented in Fig. 3.

The delay dependance on the different drive strengths of inverter is shown in Fig. 4. The 0.5X inverter is the slowest inverter in the library and 4.0X inverter is the fastest inverter in the library. 0.5X and 1.0X are single finger gate while the former has the half the length compared to the later. Hence, the delay of 1.0X inverter is very fast compared to the 0.5X inverter. The 4.0X inverter has four fingers, it can conduct more current compared to the 1.0X inverter with one finger. So, the delay of the 4.0X inverter is fast compared to the 1.0X inverter.

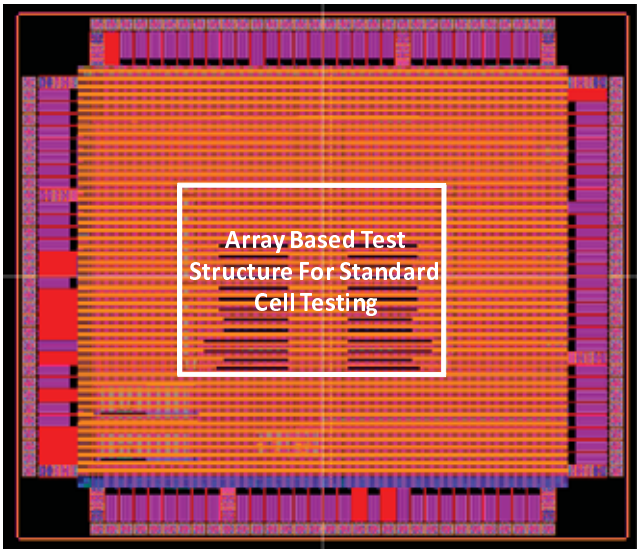


Fig. 3. Layout of test chip showing the reconfigurable array of standard cells

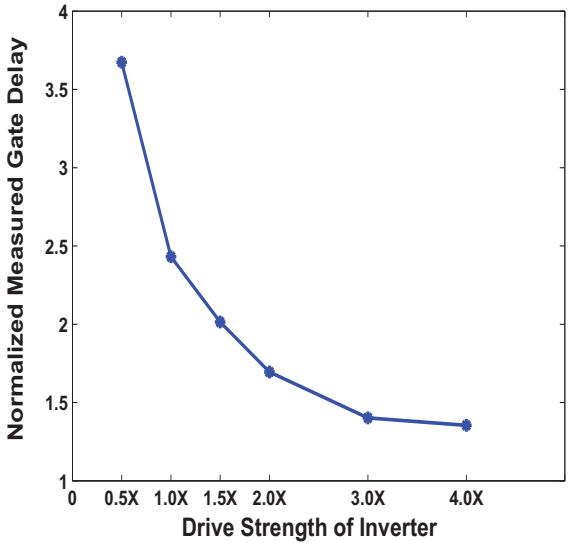


Fig. 4. Measured gate delay across different drive strength of inverter

The delay variation with supply voltage for different strength of buffer cells are shown in Fig. 5. In the both supply voltages 1.8v and 2.5v, the buffer delay decreases from 0.5X strength to 5.0X due to large current in the 5X gate having five fingers. The delay variation at supply voltage 1.8v is more compared the supply voltage 2.5v. This is because of the reduced gate overdrive.

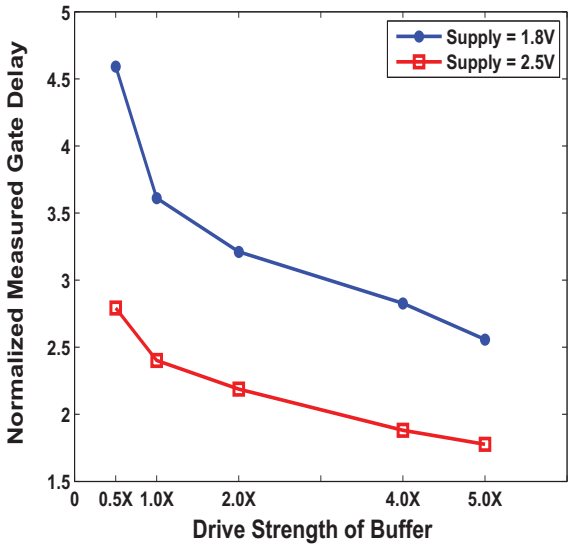


Fig. 5. Measured delay of different drive strength of buffer at two different supply Voltage 1.8v and 2.5v

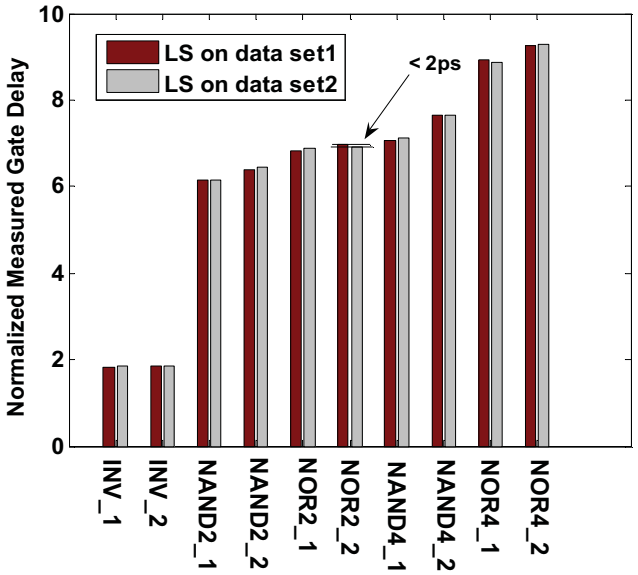


Fig. 6. Measured accuracy of gate delay measurement using least square method

A. Accuracy of the Least Square

The least square method is employed to estimate the delay of individual gate in standard cell library. We need to verify the accuracy of the approach. So, the same gate delay is estimated using two different set of status vectors. Then, the least square is applied on the measured period to find out the corresponding gate delay separately. The results of the experiment are shown in Fig. 6. It shows the error in measurement is within 2ps among 10 different types of gates. This validates our approach of using least square method for gate delay estimation. The least square method is helpful in minimizing error in period/frequency measurement due to random noise.

B. Area Comparison

We have compared our approach with normal ring oscillator based standard cell characterization [1], [2], [3], [4] in terms of area. We have assumed that the 13-stage ring oscillator is required for one gate under test measurement. The area of the 13-stage ring oscillator is the sum of the area of a 2-input NAND gate and 13 times the gate under test. For example, for 10 different types of GUTs, the area would be 10 times the 13-stage ring oscillator. However, in our approach, for testing 10 GUTs, one reconfigurable ring oscillator with 10 GDMCs are needed. In our case, the area is the sum of the area of the 10 GDMCs with 10 different types of GUTs and the area of a 2-input NAND gate. We have tested around fifty types of standard cells using this reconfigurable array test structure. All the analysis are performed using double-back layout style. Using this modeling concept, the area needed for two approaches are compared. The normal ring oscillator requires 72K μm^2 whereas the reconfigurable array based structure requires 24K μm^2(i.e. where 1K = 1000). So, the reconfigurable array structure has a saving of $2\times$ in area compared to the normal ring oscillator [1], [2], [3], [4] based standard cell characterization. The bar graph showing the area requirement using the proposed reconfigurable array versus normal ring oscillator is shown in Fig. 7. It clearly shows that reconfigurable array based test structure is area-efficient compared to normal ring oscillator approach for all types of gate in standard cell library.

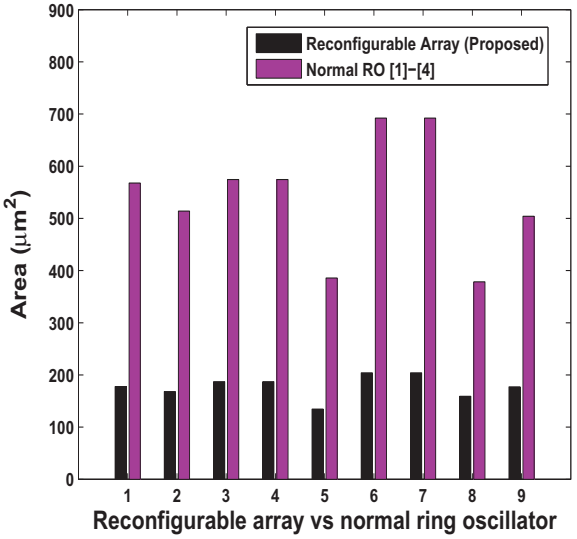


Fig. 7. Area comparison between re-configurable and normal ring oscillator for standard cell characterization

V. Conclusion

We have presented an all-digital reconfigurable array based test structure for standard cell characterization which is area-efficient. Each gates in the array can be accessed and measured using the configuration words and enable logic. During testing of standard cell, one can isolate a faulty standard cell from a non-faulty standard cell. The gate delay is estimated using least square method on the measured period/frequency. The least square method can average out the random noise in ring oscillator period/frequency measurement. We showed an area saving of $2\times$ using this reconfigurable array based test structure compared to normal ring oscillator based measurement. The accuracy of the technique is verified on measured data from a test chip in 180nm CMOS technology node.

Acknowledgment

The VLSI chip in this study has been fabricated in the chip fabrication program of VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Rohm Corporation and Toppan Printing Corporation.

References

[1] J. Panganiban, "A ring oscillator based variation test chip," M.Engg. Thesis, MIT, Dept. of EECS, May 2002.

[2] M. Bhushan and et al, "Ring oscillators for CMOS process tuning and variability control," *IEEE Trans. on Semi. Manufacturing*, vol. 19, no. 1, pp. 10–18, Feb 2006.

[3] H. Masuda, S. Ohkawa, A. Kurokawa, and M. Aoki, "Challenge: Variability characterization and modeling for 65- to 90-nm processes," in *Proc. IEEE Custom Integrated Circuits Conference*, 2005, pp. 908–913.

[4] L. T. Pang, and B. Nikolic, "Measurements and analysis of process variability in 90 nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 5, pp. 1655–1663, May 2009.

[5] P. Sanda, D. Knebel, J. Kash, H. Casal, J. Tsang, E. Seewann, and M. Papermaster, "Picosecond imaging circuit analysis of the power3 clock distribution," in *Proc. IEEE International Solid-State Circuits Conference*, 1999, pp. 372–373.

[6] J. Yu and F. F. Dai, "On-chip Jitter Measurement Using Vernier Ring Time-to-digital Converter," in *Proc. 19th IEEE Asian Test Symposium*, 2010, pp. 167–170.

[7] N. Abaskharoun and et al, "Circuits for on-chip subnanosecond signal capture and characterization," in *Proc. IEEE Custom Integrated Circuits Conference*, 2001, pp. 251–254.

[8] S. Maggioni, A. Veggetti, A. Bogliolo, and L.Croce, "Random sampling for on-chip characterization of standard cell propagation delay," in *Proc. IEEE International Symposium on Quality Electronic Design*, 2003.

[9] K. Okada, K.Yamaoka, and H.Onodera, "A statistical gate-delay model considering intra-gate variability," in *Proc. International Conference on Computer-Aided Design*, 2003, pp. 908–913.

[10] X. Zhang, K. Ishida, M. Takamiya, and T. Sakurai, "An on-chip characterizing system for within-die delay variation measurement of individual standard cells in 65-nm CMOS," in *Proc. Asia and South Pacific Design Automation Conference*, 2011, pp. 109–110.

[11] B. P. Das, B. Amrutur, H. S. Jamadagni, N. V. Arvind, and V. Visvanathan, "Within-die gate delay variability measurement using reconfigurable ring oscillator," in *Proc. IEEE Custom Integrated Circuits Conference*, 2008, to pp. 133–136.

[12] B. P. Das, B. Amrutur, H. S. Jamadagni, N. V. Arvind, and V. Visvanathan, "Within-die gate delay variability measurement using reconfigurable ring oscillator," *IEEE Trans. on Semi. Manufacturing*, vol. 22, no. 2, pp. 256–267, May 2009.

ESDQL:A Metric for Evaluating Small Delay Defect Coverage

Xuefeng Zhu[1,2], Huawei Li[1*], Xiaowei Li[1]
[1]State Key Laboratory of Computer Architecture
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, P.R. China
[2]Graduate University of Chinese Academy of Sciences, Beijing, P.R. China
{zhuxuefeng, lihuawei, lxw}@ict.ac.cn

***Abstract*—As the technology node continues to shrink, it is becoming more and more necessary to test small delay defects (SDDs). Owing to including gross delay defect coverage, prior metrics cannot properly reflect the real SDD coverage. We propose a new metric, Extended Statistical Delay Quality Level (ESDQL), for differentiating SDDs and gross delay defects (GDDs) and suitably measuring SDD coverage. Experimental results show that ESDQL is more conservative and effective in evaluating the quality of test sets in detecting SDDs and outperforms other metrics in guiding test generation.**

Keywords-small delay defect; test generation; ESDQL

I. INTRODUCTION

With the development of the technology, more and more factors (such as process variation, power supply noise, crosstalk and resistive opens/bridges) introduce small delay defects (SDDs) in the circuits. Tests based on traditional transition fault model cannot screen these SDDs. In order to achieve higher product quality, it is necessary to test SDDs of the product.

The test quality of detecting SDDs can be targeted or evaluated during transition fault testing or path delay fault testing. Path delay fault testing has the nature of capturing delay failures caused by accumulating distributed small delay variations. But it faces the problem of path selection [1]. The number of the paths in the circuit increases exponentially with the scale of the circuit. It is increasingly challenging to identify a critical path set that can represent the timing of the chip as process variation becomes more and more serious [2].

Therefore, great efforts have been made to detect SDDs in transition fault (TF) testing, which can be enhanced by reducing the slack of the sensitized paths for TFs. Either timing-aware test generation or faster-than-at-speed testing can enhance the detection of SDDs. Timing-aware test generation tries to activate the fault and propagate the fault effect through the longest path [3][4][6]. Faster-than-at-speed testing, on the other hand, reduces the slack of the sensitized paths for each TF test by increasing the frequency of test clock [5].

As can be seen from the above techniques, both sensitized path length [7][8] and test frequency should be considered in SDD coverage evaluation. In order to evaluate the effectiveness of the test set in detecting SDDs, several metrics have been proposed [6][8][10]. These metrics have three applications. To begin with, they can be used to compare the quality of different test sets. What's more, we can use them to compare the SDD coverage of different designs. Last but not least, they can be utilized to instruct test generation to get better test set.

After the GDF coverage is high enough, SDFs need to be targeted to improve the quality level. However, the existing metrics which reflect the delay test quality, such as DTC [6], SDQL [10] and SDDC [8], cannot distinguish the space of GDDs from that of SDDs. In other words, delay defects with a large delay size, namely GDDs, are not excluded in the SDD coverage evaluation. If these metrics were proposed to measure the coverage of SDDs, why do they bother to calculate the detection of GDDs in vain? What's more, the inclusion of detection on GDDs in these metrics disturbs the effectiveness of targeting SDDs, and may bring too much optimism during evaluation of SDDs. Motivated by this observation, this paper presents a new metric, extended statistical delay quality level (ESDQL) which excludes GDDs, to reflect the real SDD coverage. Our experiments demonstrate that because of not excluding GDDs from the targeting space of SDDs, existing metrics will distract from SDD detection and give misguided results in comparison with ESDQL.

The rest of this paper is organized as follows. In section II, we review three prior metrics. In section III, we propose the metric ESDQL. Section IV gives experimental results on several ISCAS'89 and ITC'99 benchmark circuits. Section V concludes the paper.

II. PRIOR WORK

In this section we review three prior metrics on delay testing quality and discuss their respective advantages and disadvantages.

Let's first introduce the following notations that are used in this paper. Just as existing metrics [6][8][9][10], for simplicity, process variation will not be considered in the calculation of the following parameters. However, one can still use conservative parameters to deal with process variation to some extent.

T_{sys}: the system clock period;
T_{test}: the test clock period;
T_{mgn}: the slack of the longest testable path passing through the fault site under the system clock;
T_{det}: the slack of the actual test path passing through the fault site under the test clock;
$T_{threshold}$: the slack of the shortest testable path passing through the fault site under the test clock;

*To whom correspondence should be addressed.

Max_pdf: the delay of the longest testable path passing through the fault site;

Pdf: the delay of the actual test path passing through the fault site;

Min_pdf: the delay of the shortest testable path passing through the fault site.

S_{defect}: the delay induced by delay defect with a certain size.

From the above notations, we have:

$$T_{mgn}+Max\_pdf=T_{sys} \quad (1)$$

$$T_{det}+Pdf=T_{test} \quad (2)$$

$$T_{threshold}+Min\_pdf=T_{test} \quad (3)$$

A. Delay Test Coverage (DTC)

The DTC metric [6] measures the SDD coverage by calculating the efficiency of a test set to test transition faults along the longest testable path passing through each fault site. The DTC of the design is calculated by taking the average weighted sum of all transition faults, as shown below:

$$w_f = \frac{Pdf}{Max\_pdf} \quad (4)$$

$$DTC = \frac{\sum_{f \in F} w_f}{N} \quad (5)$$

Where, F is the set of the TFs and N is the number of faults in F. w_f reflects the weighted value of fault f being detected by a test, which is the ratio of the delay of actual test path to the delay of the longest testable path.

If fault f is tested through the longest path by a test, w_f is 1. Note that for a test set, the largest w_f given by the tests in the test set is used for each fault in calculating DTC of the test set. So the upper bound of the DTC is equal to the TF test coverage when all the TFs are tested along their longest testable paths.

The above equation implies that the DTC metric doesn't depend on the test clock frequency and the delay defect distribution. Hence, the same design tested under two different test clocks will get the same DTC value. However, it is not appropriate to ignore the test clock frequency in test quality evaluation. Only the delay defects whose sizes are bigger than the slack of the test path can be detected, while the slack is related to both the test clock period and the delay of the testing path. Next, it can be induced that in DTC delay defects of different sizes are assumed to occur with the same probability which is unreasonable. It has been reported [11] that delay defects of different sizes have different probabilities. In general, smaller delay defects have bigger probability to occur.

B. Statistical Delay Quality Level (SDQL)

To evaluate the quality of a test set in detecting SDDs, SDQL [10] tries in the reverse direction with DTC. SDQL reflects the space of the escaped delay defect. The smaller the escaped delay defect space is, the better the quality of the test set is. As shown in Figure 1, The SDQL value of a detected transition fault f is the area of undetected defects between T_{mgn} and T_{det} under the delay defect distribution curve. The SDQL value of the whole design can be calculated using (6). SDQL assumes that the delay defect distribution function $H(s)$ has been got from the fabrication process, where s is the delay defect size and F is the number of TFs.

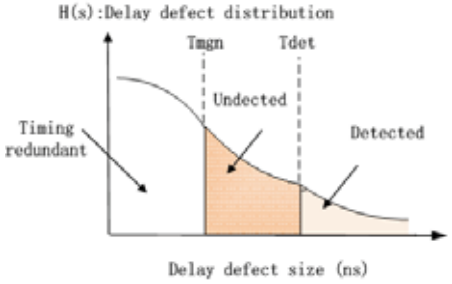


Figure 1. SDQL.

$$SDQL = \sum_{i=1}^{F} \int_{T_{mgn}}^{T_{det}} H(s)ds \quad (6)$$

From (1) and (2), we have,

$$T_{mgn}= T_{sys}\text{-}Max\_pdf; \quad (7)$$

$$T_{det}= T_{test}\text{-}Pdf; \quad (8)$$

The delay defect whose size is smaller than T_{mgn} is timing redundant because it cannot be propagated to the observation point (i.e. captured by a flip-flop) along any path in normal function. The delay defect whose size is bigger than *Tdet* is considered to be detected by the test set. The delay defect whose size is between T_{mgn} and T_{det} is considered to be likely to affect the normal function of the circuit but it cannot assure to be detected by the test set.

From the shape of $H(s)$ we can see that delay defects of different sizes have different occurring probability and the smaller the size is, the bigger the probability is. What's more, SDQL also takes into consideration of the test clock which is related with T_{det}. So compared with DTC, it overcomes the two shortcomings of DTC. However, it brings a disadvantage that DTC doesn't have. As [8] has pointed out, SDQL cannot explicitly reflect what degree that SDDs have been detected and be directly used to compare the SDD coverage of different designs. The reason is that SDQL value is related with the number of TFs. A large design may have a bigger SDQL value because of its great fault set. There is no uniform SDQL objective that can be used to guide delay test generation targeting satisfactory test quality.

C. Small Delay Defect Coverage (SDDC)

In light of the pros and cons of DTC and SDQL, the SDDC metric is proposed in [8] by learning from the advantage of DTC and offsetting the weakness of SDQL. Thus, we only compare the evaluation results with SDDC in our experiments. The SDDC of each fault is defined as the ratio of the detected delay defect space to the irredundant delay defect space which is likely to affect the normal function of the circuit.

$$w_i = \frac{\int_{T_{det}}^{\infty} H(s)ds}{\int_{T_{mgn}}^{\infty} H(s)ds} \quad (9)$$

$$SDDC = \frac{\sum_{i=1}^{F} w_i}{F} \quad (10)$$

It seems that the SDDC metric is accurate enough to represent delay test quality. However, the including of detected delay defects with large size in SDDC will weaken the guidance of the metric in evaluating the quality of the test set in detecting SDDs. Latter we can see that owing to the interference of GDDs, SDDC metric cannot concentrate to reflect the SDD coverage of the test sets and will impact the comparative result of SDD coverage among different designs.

III. ESDQL

A delay defect whose size is large enough will be detected by any transition test of the corresponding transition fault. Such defects do not need to be targeted in deliberate testing for SDDs. In view of this, we propose the extended statistical delay quality level (ESDQL), as shown in Figure 2, where $T_{threshold}$ is introduced to differentiate SDDs from GDDs that can be detected by TF tests. As the delay inequality between the long paths and short paths is becoming increasingly smaller in modern VLSI design, parameter $T_{threshold}$ is also getting smaller. Therefore, more and more smaller-size delay defects can be classified into GDDs.

We have:

$$T_{threshold} = T_{test} - Min\_pdf; \quad (11)$$

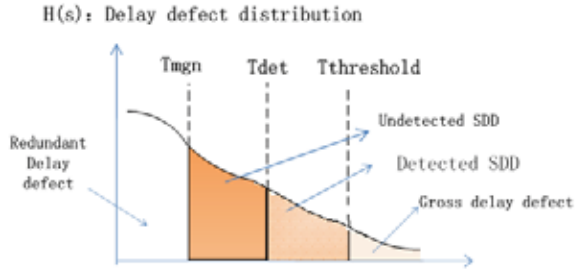


Figure 2. ESDQL.

From Figure 2 we can see that the delay defects can be divided into three categories:

- Gross delay defect: Its size is big enough to be detected along any testable path passing through the fault site. Thus GDD at a site can be tested by any corresponding transition fault test pattern. We have: $S_{GDD} > T_{threshold}$.
- Small delay defect: Its size may or may not cause the circuit fail to work. When the delay defect is tested through a short testable path, fault effect may not be observed. When tested through a long path, fault effect may be observed. We have: $T_{mgn} < S_{SDD} < T_{threshold}$.
- Redundant delay defect (RDD): It cannot affect the circuit function in any case. Even if the delay defect is tested through the longest testable path it will not make the circuit fail to work. We have: $S_{RDD} < T_{mgn}$.

Given T_{det} decided by a test pattern, SDDs are further divided into two parts, detected SDDs and undetected SDDs. Detected SDDs refer to the delay defects whose sizes are between T_{det} and $T_{threshold}$. Undetected SDDs refer to the ones whose sizes are between T_{mgn} and T_{det}. For a fault, the small delay defect coverage of a test can be defined as the ratio of the area under the delay defect distribution between T_{det} and $T_{threshold}$ to that between T_{mgn} and $T_{threshold}$. The ESDQL value of a test set is then defined in the following equations.

$$C_i = \frac{\int_{T_{det}}^{T_{threshold}} H(s)ds}{\int_{T_{mgn}}^{T_{threshold}} H(s)ds} \quad (12)$$

$$ESDQL = \frac{\sum_{i=0}^{N} C_i}{N} \quad (13)$$

The character of the ESDQL metric is that the coverage of GDDs is removed from the evaluation of SDD detection. One can still use the transition fault coverage as the coverage of GDDs. Next, two case studies will be given to compare the difference between ESDQL and SDDC.

A. Case Study 1: SDD Coverage of Different Designs

Suppose we have two designs: A, B. their system clock period and test clock period are both 10 ns. The delay of the longest testable path passing through the fault site in each design is 10 ns. The delay of every gate is 1 ns. The delay defect distribution is the same as in [6], as shown in Figure 3. Now we take a fault in each design as an example, the fault site and corresponding fault propagating paths are demonstrated in Figure 4 and Figure 5.

For design A, the delay of actual test path is 4 ns, and the delay of the shortest testable path is 2 ns. So, T_{det}=10–4=6 ns, T_{mgn}=0 ns, $T_{threshold}$=10–2=8 ns. The delay defects whose size is not smaller than 8 ns have been detected by transition fault tests.

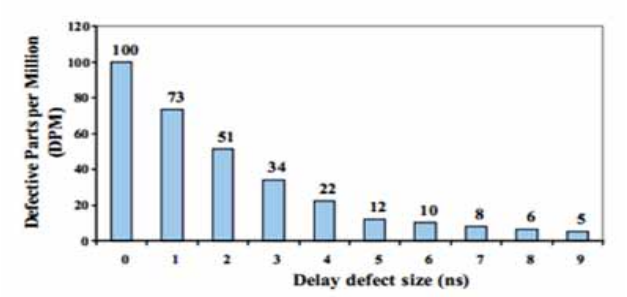


Figure 3. Delay defect distribution.

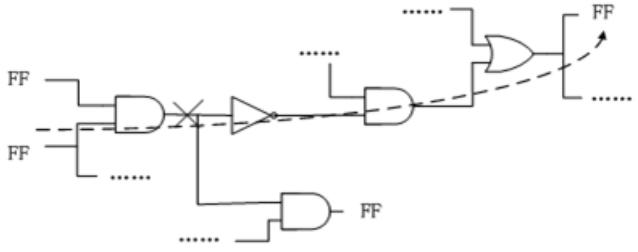


Figure 4. Design A.

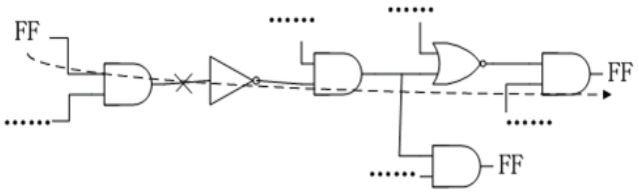


Figure 5. Design B.

For design B, the delay of actual test path is 5 ns, and the delay of the shortest testable path passing through the fault site is 4 ns. So, T_{det}=10–5=5 ns, T_{mgn}=0 ns, $T_{threshold}$=10–4=6 ns. Note that delay defects whose size is not smaller than 6 ns have been detected by transition fault tests.

Because SDDC metric doesn't distinguish GDDs and SDDs, we have:

$SDDC(A)$=(10+8+6+5)/321=29/321 (14)

$SDDC(B)$=(12+10+8+6+5)/321=41/321 (15)

$SDDC(A)<SDDC(B)$.

Considering ESDQL, the size of the delay defect covered in design A is between 6 ns and 8 ns (not containing 8 ns). The size of the SDDs which may affect the work of the circuit is between 0 ns and 8 ns (not containing 8 ns). So for design A, $ESDQL(A)$ = (10+8)/310≈0.058. For design B, the size of the delay defects detected is between 5 ns and 6 ns (not containing 6 ns). While the size of the delay defects that may affect the function of the circuit is between 0 ns and 6 ns (not containing 6 ns). So for design B: $ESDQL(B)$=12/292≈0.041. we have: $ESDQL(A)> ESDQL(B)$, which is opposite to the results of SDDC evaluation. Obviously, by removing large defects that can be detected by transition fault tests, ESDQL is superior to SDDC for SDD coverage evaluation.

B. Case Study 2 : the Quality of Different Test Sets

Suppose we have two faults in a design: f_1 and f_2. Let's evaluate the small delay defect coverage of two test sets: T_A and T_B, using SDDC and ESDQL respectively. Table I illustrates a possible test result. Both faults are detected by both test sets. The TF coverage, or GDD coverage, for both test sets, is 100%.

For test set T_A, its SDDC value is (0.7+0.8)/2=0.75. For test set T_B, its SDDC value is (0.75+0.76)/2=0.755. So, $SDDC(T_A)<SDDC(T_B)$. For test set T_A, its ESDQL value is (0.5+0.8)/2=0.65. For test set T_B, its ESDQL value is (0.7+0.5)/2=0.6. So $ESDQL(T_A)>ESDQL(T_B)$.we can conclude that due to the interference of GDD, SDDC is not as effective as ESDQL when evaluating the quality of different test sets.

TABLE I. A POSSIBLE CASE FOR TEST QUALITY EVALUATION

Fault	Metric	T_A	T_B
f_1	SDDC	0.7	0.75
	ESDQL	0.5	0.7
f_2	SDDC	0.8	0.76
	ESDQL	0.8	0.5

IV. EXPERIMENTAL RESULTS

Experiments were conducted on 6 benchmark circuits from ISCAS'89 and ITC'99. All the circuits are full-scanned and tested at-speed. The delay defect distribution is assumed the same as in [6] and shown as following:

$H(s)=1.58\times10^{-3}\times e^{-2.1s}+4.94\times10^{-6}(dpm)$

A. Evaluating SDD Detection for Different Designs

In this experiment, transition fault test sets were generated using a commercial ATPG tool. The SDDC and ESDQL values for different circuits are given in Table II. The last column of Table II shows the test clock period. From the results we can see that the SDDC values for these designs are close to 1 while the ESDQL values vary from 0.66 to 0.90. This shows ESDQL is more conservative than SDDC. The reason is obvious. Since SDDC doesn't exclude the coverage of GDDs, it is much more optimistic in evaluation. The optimism of SDDC may also cause misleading of comparison. Although the SDDC of B22 is smaller than the SDDC of B21, the ESDQL of B22 is bigger than the ESDQL of B21. Even though the SDDC of both B22 and B21 are bigger than the SDDC value of B17, both of their ESDQL values are smaller than B17. Through this experiment, we confirm our analysis that by excluding the interference of GDD, the SDD test quality reflected by ESDQL will be different from or more pessimistic than that of SDDC, and thus giving more deliberate guidance toward enhancing the SDD detection.

TABLE II. SDD DETECTION COMPARISON AMONG DIFFERENT CIRCUITS

Design	SDDC	ESDQL	Period(ns)
s38584	0.994198	0.900669	260
s38417	0.992715	0.843422	125
s35932	0.991762	0.682033	75
B22	0.930695	0.658261	360
B21	0.934716	0.655177	315
B17	0.907181	0.660612	470

Therefore, in delay quality evaluation one can still use SDDC to reflect the coverage of the detected delay defects (including GDD and SDD) as a comprehensive evaluation. But it can't give much valuable information about SDD coverage. What's more, we also recommend combining transition fault coverage and ESDQL for targeting GDD and SDD respectively. Thus, more detailed test quality information can be acquired. Testing GDDs with high coverage takes priority over detecting SDDs. If the desired quality level can't be obtained after GDD testing, S-SDFC can be used to evaluate the test quality of SDDs. The test set which can achieve high ESDQL value as well as large enough TF coverage will be preferable.

B. Evaluating SDD Detection for Different Test Sets

In this experiment, we generated two different transition fault test sets for each circuit. In order to reflect the fact that SDDC and ESDQL will give different results，two test sets with slight difference are specially chosen for each circuits. Therefore, the SDDC or ESDQL values of two sets are similar.

TABLE III. COMPARISON OF SDD DETECTION FOR DIFFERENT TEST SETS

-		s38584	s38417	s35932	B22	B21	B17
Test set 1	SDDC	0.994198	0.992720	0.991759	0.930695	0.934716	0.907181
	ESDQL	0.900669	0.843587	0.683768	0.658261	0.655177	0.660612
	Pat	272	148	35	491	555	846
	Cov	95.636%	99.46%	88.40%	98.90%	98.57%	95.93%
Test set 2	SDDC	0.994299	0.992779	0.991841	0.931747	0.932070	0.908948
	ESDQL	0.901241	0.843275	0.682168	0.662225	0.640134	0.666495
	Pat	291	146	40	481	552	853
	Cov	95.665%	99.27%	87.65%	99.98%	98.63%	95.96%

For each test set, the SDDC, ESDQL, test sets and TF coverage are given in Table III.

From the experimental results, we can see that for s35932 and s38417, although the SDDC value of test set 1 is smaller than that of test set 2, the ESDQL value of test set 1 is bigger than the ESDQL of test set 2. For other circuits, SDDC and ESDQL show the same trends. This further validates that though SDDC reflects the coverage of SDD, there is a possibility of misleading in test set comparison because of the integration of GDD into the metric.

C. Test Generation Guided by different metrics for SDD

Traditional transition fault ATPG is prone to test the transition faults through short paths, leaving many SDDs escaped. Timing-aware ATPG makes use of the delay information of the circuit to test the fault through paths as long as possible. Thus the quality of the test set generated by timing-aware ATPG is better than the traditional transition fault test set. However, as in [3], the authors pointed out that a test set generated by timing-aware ATPG can be up to eight times of the traditional transition test set, which cannot be acceptable in test application. Based on the observation that transition fault tests sometimes can also activate and propagate the faults through long paths, the authors in [5] recommended selecting a small set of transition faults which is used to do timing-aware ATPG and let the remaining faults be targeted by traditional TF ATPG.

This experiment was conducted for 4 benchmark circuits following the principle of [5], while either SDDC or ESDQL was used to guide test generation (namely the SDDC-based process or the ESDQL-based process). The process, including three phases, is shown in Figure 6, where ATPG1 means traditional TF ATPG, ATPG2 means timing-aware TF ATPG (with the same effort for all circuits). In phase (1), total fault set is targeted for traditional TF ATPG, generating a test set which is used in fault simulation to evaluate the SDDC and ESDQL value of every fault and the circuit (shown in the last row of Table IV). In phase (2), according to the descending order of the SDDC or ESDQL value of the each TF fault, the last 2000 TF faults constitute fault set 2 which is targeted for timing-aware TF ATPG and the remaining make up fault set 1 which is targeted for traditional TF ATPG. The two generated test sets are combined together as the final test set. In phase (3), fault simulation is conducted again on the final test set to evaluate the SDDC and ESDQL value of the circuit. The guidance effect of ESDQL and SDDC are illustrated using the results shown in the 3rd and 4th row of Table IV respectively.

TABLE IV compares the test generation results of traditional TF ATPG, timing-aware TF ATPG (with the same effort for all circuits), and SDDC/ESDQL guided ATPG using the process in Figure 6. Using the SDDC value obtained in Timing-aware TF ATPG as the upper bound, $SDDC_{max}$, and the SDDC value in Traditional TF ATPG as the lower bound, $SDDC_{min}$, the row "Imp(SDDC)" shows how much the corresponding SDDC value is improved by calculating: $(SDDC-SDDC_{min})/(SDDC_{max}-SDDC_{min})$. The row "Imp (ESDQL)" shows how much the corresponding ESDQL is improved in a similar way. The Ave column refers to the average effect of four circuits. We can see that for B21, B22 and s38417 the results of the ESDQL-based process are better than those of the SDDC-based process for both improvement of SDDC and ESDQL. In average, compared with the Traditional TF ATPG process, the ESDQL-based process improves the SDDC and ESDQL by 58.48% and 59.1%, respectively. However, the SDDC-based process only improves by 50.35% and 46.32%. We can conclude that compared with the SDDC-based process, the generated test set of the ESDQL-based process can cover more SDDs while ensuring high GDD coverage. The test set which is generated specially for enhancing the detection of SDDs can cover the space of all GDDs. Therefore, during the ATPG process, we only need to focus on achieving high SDD coverage with the guidance of the metric. The admixture of SDDs and GDDs weakens the ability of SDDC in deliberately generating high quality test set in detecting SDDs. On the contrary, ESDQL can instruct ATPG tools to concentrate on the generation of a better test set because of the exclusion of GDDs.

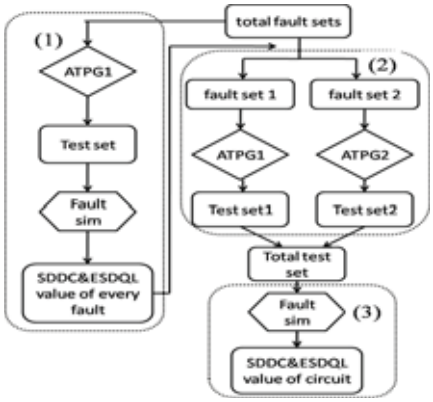


Figure 6. The process of TF ATPG guided by SDD evaluation

TABLE IV. COMPARISON OF TEST GENERATION RESULTS USING DIFFERENT METHODS

Method	metric	B21	B22	s38417	s38584	Ave
Timing-aware TF ATPG	SDDC	0.947255	0.949279	0.996060	0.997122	-
	ESDQL	0.713647	0.747040	0.918408	0.954126	
	Pat.num	6170	9293	8666	9850	
	CPU	56.87	95.38	26.65	28.21	
ESDQL-based	SDDC	0.938849	0.937932	0.995947	0.996110	-
	Imp (SDDC)	32.96%	38.94%	96.62%	65.39%	58.48%
	ESDQL	0.671573	0.693980	0.910474	0.942744	-
	Imp (ESDQL)	28.40%	40.23%	89.42%	78.71%	59.1%
	Pat.num	2869	3072	3271	2535	-
	CPU	41.86	40.65	14.41	10.13	
	TF Cov	98.56%	98.86%	99.53%	95.65%	
SDDC-based	SDDC	0.937970	0.934067	0.995601	0.996275	-
	Imp (SDDC)	25.95%	18.14%	86.28%	71.03%	50.35%
	ESDQL	0.664966	0.670703	0.903556	0.940411	-
	Imp (ESDQL)	16.74%	14.01%	80.19%	74.34%	46.32%
	Pat.num	2866	3085	3381	2668	-
	CPU	49.80	51.07	19.25	11.89	
	TF Cov	98.49%	98.81%	99.43%	95.69%	
Traditional TF ATPG	SDDC	0.934716	0.930695	0.992715	0.994198	-
	ESDQL	0.655177	0.658261	0.843422	0.900669	

V. CONCLUSION

This paper discusses the delay size of SDD and proposes the ESDQL metric for SDD coverage evaluation. In ESDQL, the coverage of GDD is removed from the evaluation of SDD detection. The experimental results show that ESDQL distinguishes itself in both evaluating the quality of different test sets and comparing the SDD coverage of different designs. Because of not excluding GDDs from the targeting space of SDDs, other existing metrics will distract from SDD detection and give misguided results in comparison with ESDQL. Our results also show ESDQL can give a better guidance during test generation toward enhancing the SDD coverage.

REFERENCES

[1] Zijian He, Tao Lv, Huawei Li, Xiaowei Li, "Fast path selection for testing of small delay defects considering path correlations" Proc. of IEEE 28th VLSI Test Symposium (VTS'10), Santa Cruz, USA, May 2010,pp.3-8..

[2] Xiang Fu, Huawei Li, Xiaowei Li, "Testable Critical Path Selection Considering Process Variation" IEICE Transactions on Information and Systems, Vol.E93-D, No.1, 2010, pp.59-67.

[3] M. Yilmaz, T. Olsen, "A Case Study of Timing-Aware ATPG", In Proc. Mentor U2U Conference, 2008.

[4] R. Mattiuzzo et al. "Small Delay Defect Testing", In Proc. SNUG San Jose 2008 Conference, 2008.

[5] S. K. Goel, N. Devta-Prasanna and R. Turakhia, "Effective and Efficient Test Pattern Generation for Small Delay Defects", In Proc. VLSI Test Symposium, pp. 111-116, 2009.

[6] Xijiang Lin, Kun-Han Tsai, et al. "Timing-Aware ATPG for High Quality At-Speed Testing of Small Delay Defects", In Proc. Asian Test Symposium, pp. 139-146, 2006.

[7] E. S. Park, M. R. Mercer, and T. W. Williams, "Statistical delay fault coverage and defect level for delay faults", In Proc. International Test Conference, pp.160-167, 1988.

[8] N. Devta-Prasanna, S.K. Goel, et al. "Accurate Measurement of Small Delay Defect Coverage of Test Patterns", In Proc. International Test Conference, paper 4.2, 2009.

[9] A. Uzzaman, M. Tegethoff, et al. "Not all Delay Tests Are the Same – SDQL Model Shows True-Time", In Proc. Asian Test Symposium, pp. 147-152, 2006.

[10] Y. Sato, S. Hamada, T. Maeda, et al. "Invisible delay quality: SDQM lights up what could not be seen", In Proc. International Test Conference,pp.47.1,2005.

[11] P. Nigh and A. Gattiker, "Test Method Evaluation Experiments and Data", In Proc. International Test Conference, pp. 454-463, 2000

Selection of the Flip-Flops for Partial Enhanced Scan Techniques

A. Matrosova, A. Melnikov, R. Mukhamedov, S. Ostanin
Tomsk State University
Tomsk, Russia
mau11@yandex.ru

V. Singh
Indian Institute of Science
Bangalore, India
viren@serc.iisc.ernet.in

***Abstract*—Structural scan based delay testing is used to detect delay faults. Because of the architectural limitations not each test pair v_1, v_2 can be applied by scan delay testing. That declines test coverage. Partial enhanced scan approach based on selection of flip-flops was suggested to permit using arbitrary test pairs v_1, v_2. The problem of selection of flip-flops may be solved with applying estimations of controllability and observability of the state variables corresponding to the flip-flops. Calculation of controllability and observability estimations is based on 2-length combinational equivalent analyses and PDF testing.**

Path delay fault (PDF); robust PDF; equivalent normal form (ENF)

I. INTRODUCTION

Delay testing has become very important problem in nanometer technologies. Structural scan based delay testing is used for detecting the circuit delays. Because of the architectural limitations not each pair v_1, v_2 can be applied by a scan delay test. Enhanced scan techniques were developed to remove the restrictions on vector pairs. Unfortunately these techniques have rarely been used in practice because of the near doubling of the flip-flop area. Most promising are partial enhanced scan approaches that are based on selection of the proper flip-flops for including them in enhanced scan chains [1]. It is desirable to use in such chains flip-flops with low controllability or/and observability oriented to delay testing. In this paper we suggest to appreciate flip-flop controllability as sum of 01 transition probability and 10 transition probability for the corresponding state variable. The flip-flop observability is appreciated as a probability of robust PDF manifestation for the paths of the same state variable. The algorithms of controllability and observability calculation are developed.

In Section 1 the problem of calculation 01(10) transition probability for the state variable is discussed. In Section 2 the method of calculation of robust PDF manifestation probability for the paths corresponding to a state variable is suggested.

1. CALCULATION OF A PROBABILITY OF 01(10) TRANSITIONS OF STATE VARIABLES

Let we have a synchronous circuit (Fig. 1) in which $x_1,...,x_n$ are input variables, $y_1,...,y_p$ are state variables, $z_1,...,z_m$ are output variables, and $d_1,...,d_p$ are flip flops.

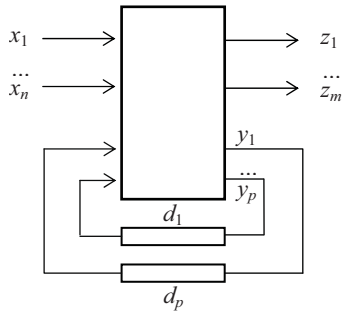


Figure 1. Synchronous circuit

Random input sequence of a sequential circuit is described with the probability distribution $\rho(x_1),...,\rho(x_n)$. Let assume that the probability distribution $\rho(y_1),...,\rho(y_p)$ of state variables is also known. Here $\rho(x_i)$, i=1…n, is the probability of 1 value of input variable x_i.

We want to calculate the probability 01(10) transition for a state variable y_i. For that consider the 2-length combinational equivalent of Fig. 2.

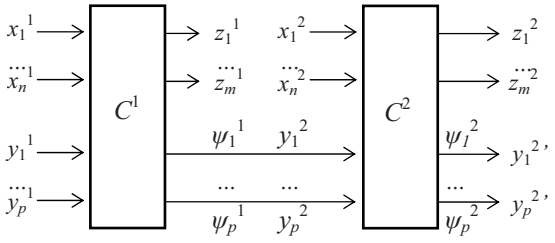


Figure 2. 2-length combinational equivalent

Having the structural description of a combinational equivalent copy we may represent the copy functions corresponding to the combinational circuit C^2 by the shared BDD which roots correspond to the state variables $y_1^{2\prime},...,y_p^{2\prime}$. These functions depend on the input variables $x_1^2,...,x_n^2$ and the state variables $y_1^2,...,y_p^2$ (Fig. 2).

Select root correlating with the variable $y_i^{2\prime}$. All paths connecting this root with the 1-terminal node of the shared BDD represent the corresponding function ψ_i^2 ($x_1^2,...,x_n^2$, $y_1^2,...,y_p^2$) as a disjoint sum of products (DSoP) D_i that is as a

sum in which products are orthogonal each other. All paths connecting this root with 0-terminal node represent an inversion of this function as DSoP $\overline{D}_i$ in the same way.

Two products are orthogonal if some variable appears in one product without inversion and in another – with inversion.

A. 01 transition probability

Extract a product K from the D_i that either contains literal $\bar{y}_i^2$ or variable y_i^2 is absent in K. In the last case we add literal $\bar{y}_i^2$ to product K. This product encloses in general case both the input and the state variables.

Notice that the state variables correspond to the transition functions $\psi_1^1,...,\psi_p^1$ that depend on the input variables $x_1^1,...,x_n^1$ and the state variables $y_1^1,...,y_p^1$ and all these variables are statistically independent on the input variables $x_1^2,...,x_n^2$ [2]. It means that we may substitute instead of the variables $x_1^2,...,x_n^2$ of the product K their probabilities and do that with each product from the D_i, selected above mentioned way. We have got the following sum:

$$\sum b_j k_j$$

Here b_j is result of multiplications of the probabilities $\rho(x_1),...,\rho(x_n)$ of the proper input variables from a set $\{x_1^2,...,x_n^2\}$ and k_j consists of some state variables (possibly with inversions) from a set $\{y_1^2,...,y_p^2\}$. The different functions corresponding to the state variables of the product k_j (they are represented with the C^1 copy) in general case depend on the same variables and these functions are statistically dependent.

Represent corresponding to the product k_j conjunction of the state variable functions with the BDD-graph.

We may calculate a probability of this conjunction using the given probability distribution for the input and the state variables.

Multiply the result by b_j. Execute this with each product K of the considered kind. Summing results we calculate the 01 transition probabilities.

B. 10 transition probability

To calculate 10 transition probabilities we need obtain DSoP $\overline{D}_i$ from shared BDD selecting root that correlates with the variable $y_i^{2\prime}$. Then we choose the products K from $\overline{D}_i$ with the literal y_i and the products that does not contain the variable y_i adding to the last products the literal y_i.

C. Example of calculating transition probability

Consider an example to illustrate the above mentioned results. The transition functions of a copy of the combination equivalent are represented with the following shared BDD (Fig. 3). For simplicity we don't use upper indexes of variables.

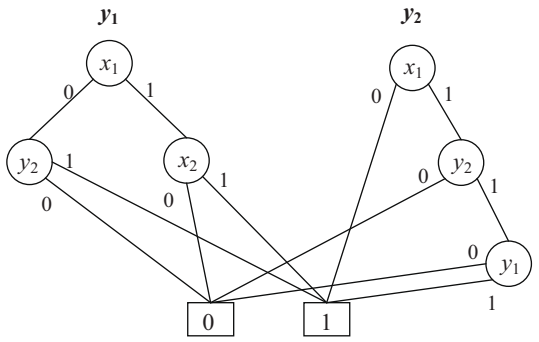


Figure 3. Shared BDD

Let probabilities of 1 value of the internal and the state variables be the same and equal to 1/2:

$$\rho(x_1) = \rho(x_2) = \rho(y_1) = \rho(y_2) = 1/2.$$

First calculate the 01 transition probability. Select the state variable $y_i^{2\prime}$ (Fig. 2). Extract the corresponding DSoP from the shared BDD (Fig. 3). Here we take into consideration upper indexes:

$$y_1^{2\prime} = x_1^2 x_2^2 \text{ v } \bar{x}_1^2 y_2^2.$$

All products do not contain y_1^2. Consequently we need add the literal $\bar{y}_1^2$ to each product. As a result we have got DSoP D^*:

$$x_1^2 x_2^2 \bar{y}_1^2 \text{ v } \bar{x}_1^2 \bar{y}_1^2 y_2^2.$$

Substitute probabilities instead of the variables x_1^2, x_2^2 and obtain the following result:

$$1/4\ \bar{y}_1^2 \text{ v } 1/2\ \bar{y}_1^2 y_2^2. \qquad (*)$$

Extract the functions $\bar{\psi}_1^1,\psi_2^1$, corresponding to the $\bar{y}_1^2$, y_2^2 from the shared BDD (Fig. 2 and 3):

$$\bar{y}_1^2 = \bar{x}_1^1 \bar{y}_2^1 \text{ v } x_1^1 \bar{x}_2^1;$$

$$\rho(\bar{y}_1^2) = 1/4+1/4 = 1/2;$$

$$y_2^2 = \bar{x}_1^1 \text{ v } x_1^1 y_1^1 y_2^1.$$

The conjunction of these functions is represented with the expression: $(\bar{x}_1^1 \bar{y}_2^1 \text{ v } x_1^1 \bar{x}_2^1)$ & $(\bar{x}_1^1 \text{ v } x_1^1 y_1^1 y_2^1)$.

The BDD of this expression is represented on Fig. 4.

Extract DSoP, representing the conjunction $\bar{y}_1^2 y_2^2$:

$$\bar{x}_1^1 \bar{y}_2^1 \text{ v } x_1^1 \bar{x}_2^1 y_1^1 y_2^1 = 1/4+1/16 = 5/16.$$

We could calculate $\rho(\bar{y}_1^2 y_2^2)$ directly from the BDD of Fig. 4.

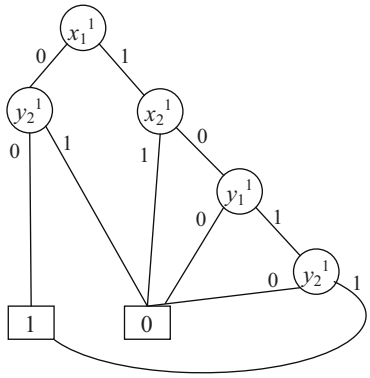


Figure 4. BDD representation of the conjunction $\bar{y}_1^2 y_2^2$

Substitute into the expression marked with (*) the obtained probabilities and calculate the probability of 01 transition:

$$\rho\,(01) = 1/4*1/2 + 1/2*5/16 = 9/32.$$

II. A PROBABILITY CALCULATION OF MANIFESTATION OF ROBUST PDFS FOR STATE VARIABLE

Consider the problem of probability calculation of manifestation of robust PDFs for state variable. First we consider robust PDF test pair properties [3]. For that we will examine the equivalent normal form (ENF) of a combinational equivalent copy.

A. Robust PDF test pair properties

An equivalent normal form represents the function implementing with the circuit and all the circuit paths. Each ENF literal is supplied with the index sequence enumerating the gates of the path. It should be noted that a literal with the same index sequence may appear in different ENF products. The ENF of the circuit (Fig. 5) is as follows (1).

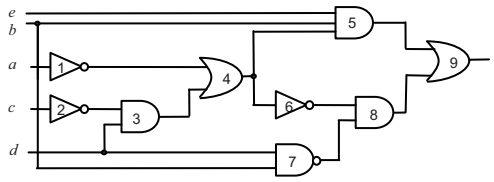


Figure 5. The combinational circuit

$$\bar{a}_{1459}\, b_{59}\, e_{59} \vee b_{59}\, \bar{c}_{23459}\, d_{3459} e_{59} \vee$$

$$a_{14689}\, \bar{b}_{789}\, c_{234689} \vee a_{14689}\, c_{234689}\, \bar{d}_{789} \vee \qquad (1)$$

$$a_{14689}\, \bar{b}_{789}\, \bar{d}_{34689} \vee a_{14689}\, \bar{d}_{34689}\, \bar{d}_{789}.$$

If literals have the same index sequences but their variables are opposite in sign then we call them as opposite literals. The same variables $x_i, \bar{x}_i$ being opposite in sign we call opposite variables. Opposite literals are absent in an ENF but opposite variables may present. We will spread all operations on products of a SoP (Sum of Products) on ENF products. We call an ENF product empty if it contains opposite variables. Notice that an ENF empty product consists of different literals.

Examine non-empty products of ENF. Any such product turns into SoP product after elimination of index sequences from literals followed with elimination of repeated variables. Sum of obtained products is SoP representing the function *f*. Non empty ENF product call implicant of the function *f*.

We consider single robust PDF of the path α in the circuit for the appropriate transitions along the path [3] as the temporary a_p (b_p)-fault of the ENF literal $x_{i\alpha}$. This fault lasts during time ω, $\omega > \tau$. Here τ is the time passing between adjacent synchronizing signals of the circuit.

The literal $x_{i\alpha}$ is changed for the constant 1 (b_p-fault) in ENF products when the corresponding PDF activates the 1 value instead of the expected 0 value on the relevant circuit output and a vector v_2.This PDF corresponds to falling transition.

The literal $x_{i\alpha}$ is changed for the constant 0 (a_p-fault) in ENF products when the corresponding PDF activates the 0 value instead of the expected 1 value on the relevant circuit output and a vector v_2. This PDF corresponds to rising transition.

A test pattern v_2 in the PDF test pair v_1, v_2 is a test pattern for b_p(a_p)-fault of ENF.

Let K be ENF product (in partly K may be empty ENF product). K is expansible with the literal $x_{i\alpha}$ if elimination of this literal gives rises to product K^* that is implicant of the function *f*. Otherwise K is not expansible with the $x_{i\alpha}$.

Elimination of the literal from a product modifies the given ENF (K changes for K^*). If K^* is non-empty product and K^* is non implicant of the function *f*, then *f* changes along with ENF. It means that the 1 value area of the function *f* increases.

K^* is the result of glue of products K and $\overline{K}$. Here $\overline{K}$ is obtained from K with changing the literal $x_{i\alpha}$ for opposite literal. We will call $\overline{K}$ as addition of K.

We suppose that the variable x_i in the literal $x_{i\alpha}$ doesn't have an inversion. First consider b_p-fault of the literal $x_{i\alpha}$.

Let $\boldsymbol{K}$ be a set of ENF products so that each product does not contain $x_{i\alpha}$.

Divide the rest ENF products into two sets: one of them $\boldsymbol{K_{rxi}}$ consists of products so that each of them has repeating variable x_i (the same variables with the same sign of inversion and the different index sequences). Products of $\boldsymbol{K_{rxi}}$ don't change the function *f* when b_p-fault takes place and consequently not generate test pattern v_b for b_p-fault.

Another set $\boldsymbol{K_{sxi}}$ consists of the products (empty and non-empty) without repeating variable x_i.

Let K be non-empty product from $\boldsymbol{K_{sxi}}$. If K is non-expansible product by literal $x_{i\alpha}$ then changing that literal for the constant 1 in K alters the function *f*. That may be detected with a test pattern v_b which turns into 1 the product $\overline{K}$ and turns into 0 free fault ENF. This test pattern is at the same time test pattern v_2 from a test pair v_1, v_2 for the corresponding PDF.

Notice that v_b turns into one the product K^* possibly together with other products derived from $\boldsymbol{K_{sxi}}$ by changing the literal $x_{i\alpha}$ for the constant 1.

If K is expansible product by literal $x_{i\alpha}$ then changing that literal for the constant 1 does not alter the function f. It means that there is no test pattern v_b for b_p-fault originated by the product K.

1) b_p-fault

Consider b_p-fault of the literal $x_{i\alpha}$ and test pattern v_b that satisfies above mentioned conditions. Variable $\bar{x}_i$ take the 1 value on v_2.

Denote as u the minimal cube covering v_1, v_2 and as $k(u)$ – the product representing u.

Theorem 1. To derive robust PDF test pair v_1, v_2 corresponding to b_p-fault we need the following.

1. v_2 is a test pattern for b_p-fault;
2. Variable x_i in v_1, v_2 takes the opposite values;
3. $k(u)$ is orthogonal to each product from $\boldsymbol{K}$;
4. Test pattern v_1 turns into 1 product K from $\boldsymbol{K_{sxi}}$ that generates test pattern v_b .

Corollary 1. The function f takes the 0 value on v_2 and takes the 1 value on v_1.

C***orollary 2.*** Empty product K from $\boldsymbol{K_{sxi}}$ doesn't originate robust PDF test pair.

Corollary 3. v_1 is a test pattern for a_p-fault.

As $k(u)$ is orthogonal to all products of $\boldsymbol{K}$ then v_1 is orthogonal to all products of $\boldsymbol{K}$ but v_1 turns into 1the function f. It means that v_1 is a test pattern for a_p-fault.

2) a_p-fault

Now consider a_p-fault of the literal $x_{i\alpha}$. All products from $\boldsymbol{K}$ remain in the fault ENF.

The rest products of the fault free ENF are divided into two sets: one of them $\boldsymbol{K_{exi}}$ consists of the empty products and another $\boldsymbol{K_{nexi}}$ consists of the non-empty products. The products of $\boldsymbol{K_{exi}}$ do not change the function f when a_p-fault takes place and consequently not generate test pattern v_a .

Consider the set $\boldsymbol{K_{nexi}}$. All its products disappear when a_p-fault takes place. If it changes the function f then there exists a test pattern v_a that detects a_p-fault. The test pattern turns into 1 some products from $\boldsymbol{K_{nexi}}$ and turns into 0 the rest products of the fault free ENF. The test pattern v_a also turns into 1 the variable x_i.

Theorem 2. To derive robust PDF test pair v_1, v_2 corresponding to a_p-fault we need the following.

1. v_2 is a test pattern for a_p-fault;
2. Variable x_i in v_1, v_2 takes the opposite values;
3. $k(u)$ is orthogonal to each product from $\boldsymbol{K}$;
4. There exists product K from $\boldsymbol{K_{nexi}}$ that does not contain repeated variable x_i so that values of the variables of the cube representing this product and values of the variables of test patterns v_1, v_2 (except variable x_i) coincide.

Corollary 1. The function f takes the 1 value on v_2 and takes the 0 value on v_1.

Corollary 2. v_1 is a test pattern for b_p-fault.

As $k(u)$ is orthogonal to all products of $\boldsymbol{K}$ then v_1 is orthogonal to all products of $\boldsymbol{K}$ and by the construction v_1 turns into 1addition of the product K. It means that v_1 is a test pattern for b_p-fault.

Theorem 3. A test pair v_1, v_2 of robust PDF corresponding to b_p-fault (falling transition) becomes the test pair for robust PDF corresponding to a_p-fault (rising transition) with rearranging these test patterns.

Notice that test pair may be represented with a product that doesn't contain variable x_i.

B. A probability calculation of robust PDF manifestation for the given path

Represent as the DSoP all robust test pairs for the given path originated by one product of ENF. For that we have to find all roots of the corresponding equation $D = 0$ representing them either as ROBDD or Free BDD. Each path from root till 1-terminal node arises to the product, corresponding to 2^{n-1-r} robust test pairs. Here r is rank of the product and n is the number of ENF variables. (Notice that the variable x_i is absent in the product). Having got the BDD for different ENF products that correspond to the same path α and do not contain repeated variable x_i we may execute disjunction operation on these BDDs and obtain BDD representing all robust test pairs. Using the last BDD we may calculate a probability of the robust PDF manifestation for the path α substituting instead of the variables their probabilities from the probability distribution.

Illustrate that procedure by an example. Let the variable c be a state variable. First consider the literal c_{234689}. Take into consideration that for the product $a\bar{b}c$, $k(u) = a\bar{b}$, and the corresponding D, $D = \bar{d}\ v\ \bar{d}\bar{d}$. The BDD represents the only product d.

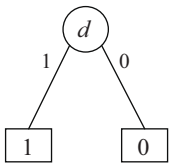


Figure 6. BDD representation of the product d

Consider forth product of the ENF that contains the literal c_{234689}. In this case $K = ac\bar{d}$, $k(u) = a\bar{d}$, $D = \bar{b}$ v 1 = 1.The equation $D = 0$ has no roots.

Consequently a probability of robust test pair manifestation for the path α corresponding to the literal c_{234689} is represented by the formula $a\bar{b}d$ and is equal to 1/2*1/2*1/2 = 1/8.

Here we suppose 1 value probability of each variable is equal to 1/2.

C. *A probability calclation of robust PDF manifestation for state variable*

When we consider a state variable we have to regard all paths (literals) connected with this variable. Derive for each path a probability of robust PDF manifestation in the above mentioned way. Then we have to summarize these probabilities.

Notice that a test pair for one path does not change the signal values of other path corresponding to the same variable as all products representing another paths are contained in a set ***K*** formed for the path considered. It means that sensitizations of different paths of the same variable are statistically non-compatible events.

Let c be state variable. We additionally have to find a probability of robust PDF manifestation for the path corresponding to the literal $\bar{c}_{23459}$. Then $K = b\bar{c}de$, $k(u) = bde$, $D = \bar{a}$.

BDD originates the only root: product a. Consequently a probability of robust PDFs manifestation for the path corresponding to the literal $\bar{c}_{23459}$ is represented by the formula $abde$. It is equal to 1/2*1/2*1/2*1/2 = 1/16. Then a probability of manifestation of robust PDFs for the state variable c is as follows: 1/8+1/16 = 3/16.

We have got the following experimental results.

Table 1. Experimental results on Iscas89 benchmarks

Circuit	Flip-flops	State Variable	*Pc*	*Po*
s27	3	G11	**0.172**	**0.125**
		G10	0.457	**0.063**
		G13	0.344	0.250
s344	15	ACVG3VD1	0.331	**0.219**
		ACVG2VD1	0.260	**0.219**
		AM3	**0.125**	0.875
		ACVG4VD1	**0.180**	**0.219**
		CNTVG2VD	**0.188**	0.375
		AM2	0.281	0.875
		CNTVG3VD	**0.188**	0.500
		AM1	0.281	0.875
		AM0	0.281	0.875
		MRVG3VD	0.500	**0.125**
		MRVG4VD	0.641	**0.125**
		MRVG1VD	0.406	**0.125**
		MRVG2VD	0.406	**0.125**
		CNTVG1VD	0.594	0.375
		ACVG1VD1	0.280	**0.219**
s444	21	G58	**0.209**	**0.141**
		G112	0.335	**0.000**
		G49	**0.063**	**0.063**
		G111	**0.224**	**0.000**
		G45	**0.094**	**0.188**
		G41	**0.125**	**0.125**
		G113	**0.130**	**0.000**
		G162BF	**0.129**	0.453
		G80	**0.133**	0.343
		G70	**0.091**	**0.186**
		G101	0.375	0.500
		G66	**0.113**	0.275
		G110	**0.233**	**0.000**
		G62	**0.119**	**0.230**
		G109	**0.099**	**0.000**
		G84	**0.120**	0.382
		G92	**0.108**	0.362
		G155	0.375	0.500
		G88	**0.115**	0.402
		G114	**0.026**	**0.000**
		G37	0.578	**0.000**

The controllability *Pc* and the observability *Po* have been calculated for each state variable of some benchmarks. Flip-flops corresponding to state variables with low controllability or/and observability can be selected for including in enhanced scan chains. We suppose that a probability is low if it's less than 0.25 (such probabilities are marked with bold).

The controllability *Pc* was calculated as the sum of probabilities of 10 and 01 transitions.

The flip-flop observability is estimated as a probability of manifestation of robust PDFs of the same state variable. The algorithms of estimation calculations are based on ENF analysis and using BDDs. ENF is very complicate formula for real circuits. It is possible to use Marked OR/AND tree to present all paths of a circuit [4]. These trees are used for finding the estimations for benchmarks of the Table 1.

We may use for description of the ENF of a circuit the system of the Marked OR-AND trees. The complexity of the system is linear function of the number of the circuit gates [4]. To accelerate the calculation of flip-flop observability it is possible joint application of the system of OR-AND trees and the corresponding system of SSBDDs [5]. We hope that these techniques will allow calculating flip-flop observability for more complicate circuits.

III. CONCLUSION

The approach for choosing flip-flops for including them into partial enhanced scan chains is suggested. For that the special estimations of flip-flop controllability and observability are developed. Possibilities of their application to more complicate circuits are noted.

REFERENCES

[1] G. Xu and A. D. Singh," Achieving High Transition Delay Fault Coverage with Partial DTSFF Enhanced Scan Chains" Proceedings of International Test Conference, 2007,pp. 1-9.

[2] A. Matrosova, "Random simulation of logical circuits," Avtomatika and Telemechanika, № 1, 1995, pp. 156-164.

[3] A. Matrosova, V. Lipsky, A. Melnikov, V. Singh, "Path Delay Faults and ENF," Proceeding of EWDT Symposium, Russia, St. Petersburg, September, 2010, pp. 164-167.

[4] A. Matrosova, A. Andreeva, A. Melnikov, E. Nikolaeva, "Multiple Stuck-at Fault and Path Delay Fault Testable Circuits," Proceedings of, EWDT Symposium, Lvov, Ukraine, October 9-12, 2008, pp. 356-364.

[5] R.Ubar. Multi-Valued Simulation of Digital Circuits with Structurally Synthesized Binary Decision Diagrams.OPA (Overseas Publishers Assotiation) N.V. Gordon and Breach Publishers, Multiple Valued Logic, Vol.4,pp.141-157.

Author Index

A

Masayuki Arai..... 90

B

Bernd Becker..... 35
Lakshmanan Balasubramanian 23

C

Alexander Czutro..... 35
Gunnar Carlsson..... 61

D

Bishnu Prasad Das..... 113

E

K. Enokimoto..... 31

F

Hideo Fujiwara..... 9,16,77
Hiroaki Fujiwara..... 16
Katsuya Fujiwara..... 77
Satoshi Fukumoto..... 90

H

Toshinori Hosokawa.....16,69
Fumiyuki Hafuri 108

I

Ryoichi Inoue..... 16
Urban Ingelsson..... 61
Kenta Imai..... 90
Tsuyoshi Iwagaki.....108
Hideyuki Ichihara.....108
Tomoo Inoue.....108

J

Jie Jiang 35

K

S. Kajihara..... 31
Samayeh Sadeghi-Kohan.....53
Parisa Sha'afi Kabiri....53,100
Yoshifumi Koyama..... 90

L

Erik Larsson..... 61
Huawei Li.....119
Xiaowei Li.....119

M

Toshiya Mukai.....108
A. Matrosova.....47,125
A. Melnikov.....125
R. Mukhamedov.....125
R.K. Mittal..... 23
Kohei Miyase.....31

N

E. Nikolaeva..... 47
Zainalabedin Navabi53,83,100
Reza Nakhjavani..... 83

O

Chia Yee Ooi..... 9
S. Ostanin.....47,125
Hidetoshi Onodera.....113

P

Norlina Binti Paraman..... 9
Ilia Polian..... 35

R

M. Sonza Reorda..... 94

S

Ahmad Zuri Bin Sha' ameri... 9
Matthias Sauer.....35
V. Singh..... 47,125
Ali Shahabi..... 83
Saeed Safari.....83
Aromhack Saysanasongkham 90
E. Sanchez..... 94

T

H. Tanaka..... 31
Hideo Tamamoto..... 77

U

Kenji Ueda..... 108

V

Ghazaleh Vazhbakht..... 53

W

X. Wen..... 31
Motohiro Wakazono..... 69

X

Dong Xiang..... 41

Y

Hiroshi Yamazaki..... 69
Masayohi Yoshimura..... 69

Z

Farrokh Ghaani Zadegan..... 61
Xuefeng Zhu.....119