2nd IEEE International Workshop on Reliability Aware System Design and Test

In conjunction with
The International Conference on VLSI Design
January 6-7, 2011
Chennai, India

RASDAT 2011

Table Of Contents

Message from the Chairs	1
Technical Program committee	2
Program Schedule	3

Session – I **Topic: Power Aware Design and Test** **Session Chair: Seiji Kajihara, Kyushu Institute of Technology, Japan**	
A Test Pattern Optimization to Reduce Spatial and Temporal Temperature Variations *Tomokazu Yoneda, Makoto Nakao, Michiko Inoue, Yasuo Sato, and Hideo Fujiwara* *(NAIST, Japan, and Kyushu Institute of Technology, Japan)*	7
Test Scheduling for 3D Stacked ICs Under Power Constraints *Breeta SenGupta, Urban Ingelsson, and Erik Larsson* *(Linkoping University, Sweden)*	13
Low Power Programmable Controllers for Reliable and Flexible Computing *Masahiro Fujita, Hiroaki Yoshida, and Jaeho Lee* *(Tokyo University, Japan)*	19

Session – II **Topic: VLSI Test** **Session Chair: Mark Zwolinski, Southampton Univ., UK**	
Dynamic Scan Clock Control in BIST Circuits *Priadarshini Shanmugasundaram and Vishwani Agrawal* *(Auburn University, USA)*	25
A Pattern Partitioning Algorithm for Field Test *Senling Wang, Seiji Kajihara, Yasuo Sato, Xiaoxin Fan, and Sudhakar Reddy* *(Kyushu Institute of Technology, Japan, University of Iowa, USA)*	31
Optimal Universal Test Set for Bridging Faults Detection in Reversible Circuit Using Unitary Matrix *Susanta Chakraborty, Pradyut Sarkar, and Bikramadittya Mondal* *(BESU Sibpur, Simplex Infrastructure, and BPPIMT, Kolkata)*	37

Session -- III:Advances in Test **Topic: Reliable Systems** **Chair: Tomokazu Yoneda, NAIST, Japan**	
On The Design of Self-Recovering Systems *Yang Lin and Mark Zwolinski* *(Southampton University, UK)*	43
Approximate and Bottleneck High Performance Routing for Self-healing VLSI Circuits *Achira Pal, Tarak Mandal, Alak Datta, Rajat Pal, Atal Chaudhuri* *(HSBS, PMI Service Centre Europe Krakow Poland, Asam Univesrity Silchar, Jadavpur University Kolkata)*	48
A Scalable Heuristic for Incremental High-Level Synthesis and Its Application to Reliable Computing *Shohei Ono, Hiroaki Yoshida, and Masahiro Fujita* *(Tokyo University, Japan)*	54

A Study of Failure Mechanisms in CMOS & BJT ICs and Their Effect on Device Reliability *Adithya Thaduri, M. Rajesh Gopalan, Gopika Vinod, and Ajit Verma* *(IIT Bombay, BARC, and IIIT Pune)*	60

Posters	
Test Data Compression Technique for IP Core Based SoC using Artificial Intelligence *Usha Mehta, KS Dasgupta, Nirjan Devashrayee, and Harikrishna Parmar* *(Nirma University Ahmedabad and ISAC Ahmedabad)*	65
A Design Methodology for Specification and Performances Evaluation of Network-on-Chip *Adrouche Djamel, Sadoun Rabah, and Pillement Sebastien* *(Ecole Nationale Politechnique Algeraia, and Univ. of Rennes)*	71
FOCAS: A Novel Framework for System-On-Chip Datapath Validation *Balvinder Khurana, and Atul Gupta* *(Freescale Semiconductors, Noida, India)*	77
Synthesis of Reversible Logic Circuit using Unitary Matrix *Susanta Chakraborty, Bikramadittya Mondal and Pradyut Sarkar* *(BESU Sibpur, BPPIMT Kolkata, and Simplex Infrastructure Kolkata)*	84
Efficient SOPC-Based Multicore System Design Using NOC *Arunraj Subramanyan, Vanchinathan Thankavel* *(Anna University and Femto Logic Design)*	89

Author Index	95

Message from the Chairs

Welcome to RASDAT 2011, the Second Annual International Workshop on Reliability Aware System Design and Test, being held in conjunction with the International Conference on VLSI Design in Chennai on January 6 and 7, 2011. In launching this new series of annual workshops last year, the Steering Committee's vision was to complement the broader International Conference on VLSI Design by bringing into sharper focus key challenges and design issues related to reliability and test of aggressively scaled microelectronic systems. The overwhelming success of the first RASDAT workshop has encouraged us to plan an even more ambitious and interesting program this year.

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 reliability challenges resulting from increasing defect rates, manufacturing variations, soft errors, wearout, etc. need to be addressed by innovative new design and test methodologies, 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 the first RASDAT program. On a day and a half, we will have a keynote addresses, an embedded tutorial, four invited talks, a panel and three sessions where 10 papers and 5 posters will be presented. The workshop begins on January 6 with a keynote address by John Carulli (Texas Instruments, USA), followed by an embedded tutorial on Post Silicon Debug by Masahiro Fujita (Tokyo University, Japan), two invited talks by Arun Somani (Iowas State University, USA), and Sanjit Seshia (University of California, Berkeley). On January 7, it will start with a keynote address by Michel Renovell (LIRMM, France). We will have a special session on hardware security and the speakers will be Jacob Abraham (University of Texas, Austin, USA), Michiko Inoue (NAIST, Japan), and Rajat Moona (IIT kanpur, India) and an invited talk by M. Ravindra (Indian Space Research Organization, India), . The 10 papers accepted to RASDAT are organized into the three sessions entitled power aware design and test, VLSI test, and reliable systems. In addition 5 posters will be presented in interactive session during coffee break. The workshop will be concluded by a panel entitled "Volume Diagnosis: Power in Numbers?".

Any technical meeting is the result of the work of many dedicated volunteers; RASDAT-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 RASDAT-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 thankful to VLSI Society of India (VLSI), Indian Institute of Technology Madras, Chennai, LSI Technologies, and AMD for sponsoring the meeting. Last but not least we are also thankful to IISc, Bangalore and MNIT, Jaipur for their support.

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

Adit Singh
Virendra Singh
Michiko Inoue
Sreejit Chakravarty

General Co-Chairs

Erik Larsson
Rubin Parekhji
Ilia Polian
MS Gaur

Program Co-Chairs

Technical Program Committee

- M. Azimane, *NXP Semiconductors, The Netherlands*
- B. Bhattacharya, *ISI, India*
- P. Bernardi, *Politecnico di Torino, Italy*
- K. Chakrabarty, *Duke University, USA*
- K.T. Cheng, *University of California, Santa Barbara, USA*
- G. Di Natale, *LIRMM, France*
- E. Fernandes Cota, *Universidade Federal do Rio Grande do Sul, Brasil*
- P. Harrod, *ARM, United Kingdom*
- U. Ingelsson, *Linkoping University, Sweden*
- G. Jervan, *Tallinn University of Technnology, Estonia*
- P. Girard, *LIRMM, France*
- S. K. Goel, *USA*
- V. Hahanov, *Kharkov National University of Radioelectronics, Ukraine*
- K. Hatayama, *Semiconductor Technology Academic Research Center, Japan*
- S. Hellebrand, *Universität Paderborn, Germany*
- T. Inoue, *Hiroshima University, Japan*
- V. Kamakoti, *IIT Madras, India*
- S. Kajihara, *Kyushu Institute of Technology, Japan*
- R. Kapur, *Synopsys, USA*
- H. Ko, *McMaster University, Canada*
- S. Kundu, *University of Massachusetts Amherst,USA*
- S. Kumar, *University of Jonkoping, Sweden*
- V. Laxmi *(Malaviya National Institute of Technology, India)*
- Y. Makris, *Yale University, USA*
- E. Marinissen, *IMEC, Belgium*
- S. Mitra, *Standford University, USA*
- C. Metra, *University of Bologna, Italy*
- Z. Navabi, *Tehran University, Iran*
- N. Nicolici, *McMaster University, Canada*
- C.Y. Ooi, *Malaysia Technology University, Malaysia*
- A. Osseiran, *Edith Cowan University, Australia*
- S. Othake, *Nara Institute of Science and Technology, Japan*
- J. Raik, *Tallinn University of Technnology, Estonia*
- S. Ravi, *Texas Instruments, India*
- M. Renovell, *LIRMM, France*
- B. Rouzeyre, *LIRMM, France*
- M. Sonza Reorda, *Politecnico di Torino, Italy*
- N. Tamrapalli, *AMD, India*
- P. Thaker, *Analog Devices, India*
- P. Varma, *Bluepearlsoftware, USA*
- V. Vedula, *Intel, India*
- B. Vermeulen, *NXP Semiconductors, The Netherlands*
- M. Violante, *Politecnico di Torino, Italy*
- H.-J. Wunderlich, *Universität* Stuttgart, Germany
- Q. Xu, *The Chinese University of Hong Kong, China T. Yoneda, Nara Institute of Science and Technology, Japan*
- Z. You, *Hunan University, China*
- M. Zwolinski, *University of Southampton, United Kingdom*

Advance Program

Day-1: **6th January (Thursday)**

Venue: IIT Madras, Chennai

Time	***Program***
9:00 am – 2:00 pm	Registration
2:00 pm – 2:15 pm	Inaugural Ceremony
2:15 pm – 3:00 pm	Keynote Address Speaker: John Carulli, Texas Instruments, USA
3:00 pm – 3:45 pm	Embedded Tutorial Speaker: Masahiro Fujita, Tokyo University, Japan Topic: Post Silicon Debug
3:45 pm – 4:15 pm	Coffee Break
4:15 pm – 4:45 pm	Invited talk –I Speaker: Arun Somani, Iowa State University, USA Topic: RAKSHA: Reliable and Aggressive frameworK for System design using High-integrity Approaches
4:45 pm – 5:45 pm	Session – I Topic: Power Aware Design and Test Session Chair: Seiji Kajihara, Kyushu Institute of Technology, Japan
5:45 pm – 6:15 pm	Invited Talk – II Speaker: Sanjit Seshia, Univ. of California, Berkeley Topic: On Voting Machine Design for Verification and Testability
6:30 pm – 8:30 pm	Cultural program and Banquet

Day-2: **7th January (Friday)**

Venue: IIT Madras

Time	*Program*
8:30 am - 9:15 am	Keynote - II Speaker: Michel Renovell, LIRMM, France
9:15 am - 10:45 am	Special Session on Hardware Security Chair: Ilia Polian, Univ. of Passau, Germany Speakers Jacob Abraham, Univ. of Texas, Austin, USA Michiko Inoue, NAIST, Japan Rajat Moona, IIT Kanpur, India
10:45am -11:15 am	Coffee Break and Poster
11:15am -12:15 pm	Session–II Topic: VLSI Test Session Chair: Mark Zwolinski, Southampton Univ., UK
11:15 am -12:45pm	Invited Talk - III Speaker: M. Ravindra, ISRO, India
12:45 pm - 1:45 pm	Lunch Break
1:45 pm -3:15 pm	Session -III: Advances in Test Topic: Reliable Systems Chair: Tomokazu Yoneda, NAIST, Japan
3:15 pm - 3:45 pm	Coffee Break
3:45 pm -5:15 pm	Panel Discussion Topic: Volume Diagnosis: Power in Numbers? Moderator: Sudhakar Reddy, Iowa University, USA
	Panelists: Srikant Venkataraman, Intel, USA Nagesh Tamrapalli, AMD, India Rubin Parekhji, Texas Instruments, India Adit Singh, Auburn University, USA Vishwani Agrawal, Auburn University, USA

Time	*Program*
5:15 pm - 5:30 pm	Closing

Session-I: Power aware design and test

S1.1. A Test Pattern Optimization to Reduce Spatial and Temporal Temperature Variations
Tomokazu Yoneda, Makoto Nakao, Michiko Inoue, Yasuo Sato, and Hideo Fujiwara
(NAIST, Japan, and Kyushu Institute of Technology, Japan)

S1.2. Test Scheduling for 3D Stacked ICs Under Power Constraints
Breeta SenGupta, Urban Ingelsson, and Erik Larsson
(Linkoping University, Sweden)

S1.3. Low Power Programmable Controllers for Reliable and Flexible Computing
Masahiro Fujita, Hiroaki Yoshida, and Jaeho Lee
(Tokyo University, Japan)

Session-II: VLSI Test

S2.1. Dynamic Scan Clock Control in BIST Circuits
Priadarshini Shanmugasundaram and Vishwani
Agrawal (Auburn University, USA)

S2.2. A Pattern Partitioning Algorithm for Field Test
Senling Wang, Seiji Kajihara, Yasuo Sato, Xiaoxin Fan, and Sudhakar Reddy
(Kyushu Institute of Technology, Japan, University of Iowa, USA)

S2.3. Optimal Universal Test Set for Bridging Faults Detection in Reversible Circuit Using Unitary
Matrix Susanta Chakraborty, Pradyut Sarkar, and Bikramadittya Mondal
(BESU Sibpur, Simplex Infrastructure, and BPPIMT, Kolkata)

Session-III: Reliable Systems

S3.1. On The Design of Self-Recovering
Systems Yang Lin and Mark Zwolinski
(Southampton University, UK)

S3. 2. Approximate and Bottleneck High Performance Routing for Self-healing VLSI Circuits
Achira Pal, Tarak Mandal, Alak Datta, Rajat Pal, Atal Chaudhuri
(HSBS, PMI Service Centre Europe Krakow Poland, Asam Univesrity Silchar, aJadavpur University Kolkata)

S3.3. A Scalable Heuristic for Incremental High-Level Synthesis and Its Application to Reliable Computing
Shohei Ono, Hiroaki Yoshida, and Masahiro Fujita
(Tokyo University, Japan)

S3.4. A Study of Failure Mechanisms in CMOS & BJT ICs and Their Effect on Device Reliability Adithya Thaduri, M. Rajesh Gopalan, Gopika Vinod, and Ajit Verma (IIT Bombay, BARC, and IIIT Pune)

Posters

P1. Test Data Compression Technique for IP Core Based SoC using Artificial Intelligence Usha Mehta, KS Dasgupta, Nirjan Devashrayee, and Harikrishna Parmar
(Nirma University Ahmedabad and ISAC Ahmedabad)

P2. A Design Methodology for Specification and Performances Evaluation of Network-on-Chip Adrouche Djamel, Sadoun Rabah, and Pillement Sebastien
(Ecole Nationale Politechnique Algeraia, and Univ. of Rennes)

P3. FOCAS: A Novel Framework for System-On-Chip Datapath Validation Balvinder Khurana, and Atul Gupta
(Freescale Semiconductors, Noida, India)

P4. Synthesis of Reversible Logic Circuit using Unitary Matrix
Susanta Chakraborty, Bikramadittya Mondal and Pradyut sarkar
(BESU Sibpur, BPPIMT Kolkata, and Simplex Infrastructure Kolkata)

P5. Efficient SOPC-Based Multicore System Design Using NOC Arunraj Subramanyan, Vanchinathan Thankavel (Anna University and Femto Logic Design)

A Test Pattern Optimization to Reduce Spatial and Temporal Temperature Variations

Tomokazu Yoneda†*, Makoto Nakao†*, Michiko Inoue†*, Yasuo Sato‡* and Hideo Fujiwara†*
†Nara Institute of Science and Technology, Ikoma-shi, Nara, 630-0192, Japan
‡Kyusyu Institute of Technology, Iizuka-shi, Fukuoka, 820-8502, Japan
* Japan Science and Technology Agency, CREST, Chiyoda-ku, Tokyo, 102-0075, Japan

Abstract

Circuit failure prediction is essential to achieve high filed reliability in the deep sub-micron process. One of the key techniques for circuit failure prediction is online delay measurement or delay test to capture the gradual delay shift caused by transistor aging. However, the delay shift are caused not only by transistor aging but also by environmental conditions such as temperature and voltage. Therefore, for accurate failure prediction, it is important to eliminate the environmental delay variations from the measure delays for capturing the actual delay shift caused by aging. This paper presents a novel test pattern optimization method to reduces spatial and temporal temperature variations during delay test. Consequently, we can easily eliminate the temperature -induced delay variations from the measured delays for accurate failure prediction. Experimental results for ITC'99 benchmark circuits show the effectiveness of the proposed method compared to the existing approaches.
keywords: *test pattern ordering, thermal-uniformity, delay variation, reliability, failure prediction.*

1 Introduction

Rapid advances in silicon manufacturing technologies have led to their drastic increases in transistor density and operational frequency. Among the many challenges imposed by the technologies, high field reliability is becoming major concern and periodical online self-test is essential for overcoming reliability challenges such as early-life failures and transistor aging [1, 2]. One of the major applications of online self-test is circuit failure prediction. Circuit failure prediction predicts the occurrence of a circuit failure during normal system operation before the appearance of any error [3], and it is ideal for predicting transistor aging because of the gradual nature of delay degradation [3, 4]. Failure prediction can be achieved in multiple ways but the basic principle is to capture the gradual delay shift caused by transistor aging inside a chip. However, the delay shift are caused not only by transistor aging but also by environmental conditions such as temperature and voltage. Accordingly, for accurate failure prediction, circuit delay as well as temperature and voltage are collected periodically, and the collected data is analyzed on-chip or off-chip to predict failures [5, 6, 7]. In [6, 7], delays are measured with temperature and voltage, and the measured delays are translated into delays at a reference temperature and voltage using the measured temperature and voltage to eliminate the environmental delay variations. Then, the translated delays are compared to the past translated delay values stored in on-chip or off-chip memory to estimate the actual delay shift caused by aging.

The execution of online self-test may last for a long time [8]. For example, the extremely thorough test patterns that specifically target aging [9] may take several seconds to run. Besides, in nanometer technologies, the temperature-induced delay variations during test is as much as that is caused by on-chip process variations [10]. Temperature difference between different locations in a circuit can be as high as 40°C to 50°C, and typical time intervals for temperature changes over time are very short time of milliseconds. This indicates that we need to embed a lot of temperature sensors into several locations within a chip to collect spatially and temporally accurate temperature profile during test. However, this is not the cost effective solution since it incurs a lot of overhead. Even if we could accept the overhead, it requires (1) a lot of data to be stored in the memory and (2) a complex procedure to eliminate the temperature induced delay variations from the measured delay values for failure analysis. Therefore, we need a cost effective solution to eliminate the spatial and temporal temperature-induced delay variations for accurate failure prediction.

Bahukudumbi et al. proposed a power management framework for wafer-level test-during-burn-in (WLTBI) where the junction temperature needs to be maintained at a constant value [11, 12]. They addressed the problem of temporal power variations during WLTBI and proposed methods that manipulate test patterns to minimize the temporal variance in power consumption on a cycle-by-cycle basis. However, as they used a global power value per circuit for each clock cycle, the spatial variations of power consumption on a circuit were not considered. Since spatial variations are a local features, the global power value per circuit would not be enough to guarantee thermal uniformity, especially, for a large circuit. Recently, we proposed a thermal-uniformity-aware X-filling technique that minimizes spatial temperature variation for high quality and accurate delay testing [13]. We also showed the effect on the final temperature profiles as well as the power profiles. However, there still exist temporal temperature variations that should be minimized to eliminate temperature-induced delay variations for accurate failure prediction.

In this paper, we present a novel test pattern optimization method to minimize spatial and temporal temperature variations during delay test. Since the optimized test sequence creates a spatially and temporally uniform temperature profile during test, we can collect the temperature profile accurately with few sensors and easily eliminate the temperature -induced delay variations from the measured delays for accurate failure prediction. The proposed method consists of an X-filling technique and a test pattern ordering technique for given test sequences with unspecified bits (X's). Therefore, it is easily embedded into the current design flow without any loss of fault coverage. To the best of our knowledge, this is the first paper that addresses a viable solution to achieve spatial and temporal thermal-uniformity during test. The main contributions of the paper are as follows.

- It shows that test patterns generated by a commercial ATPG create significant amount of spatial and temporal temperature variations, and emphasizes the importance of achieving spatial and temporal thermal-uniformity for high quality and accurate delay testing.
- It presents a novel test pattern ordering method that mini-

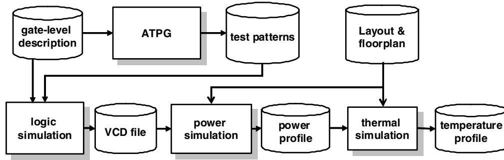


Figure 1. Flow to estimate power and temperature.

mizes temporal temperature variations while preserving the spatial temperature variations.

- The thermal simulation results for ITC'99 benchmark circuits show the effectiveness of the proposed test pattern optimization technique.

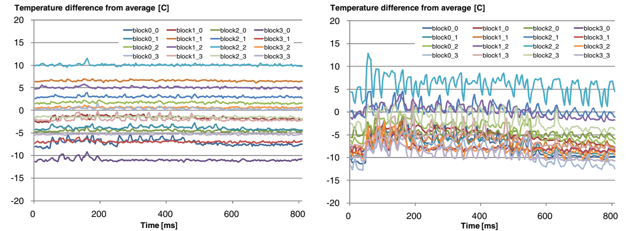
The rest of this paper is organized as follows. Section 2 evaluates test patterns generated by a commercial ATPG using the thermal analysis flow. The evaluation shows the need of thermal-uniformity for high quality delay test. Section 3 defines power and temperature characterization used in this paper. Section 4 presents a test pattern optimization method that achieves thermal-uniformity. Section 5 shows some experimental results using ITC'99 benchmark circuits. Finally, Section 6 concludes the paper.

2 Thermal Analysis of Current Test Patterns

In this section, we evaluate the temperature profiles of test patterns generated by a commercial ATPG. We developed the flow shown in Figure 1, and used ITC'99 benchmark circuit $b17$ and a 45nm library [14] to evaluate the spatial and temporal temperature variations of the tests. We generated launch-off-capture (LoC) test patterns with unspecified bits (X's) targeting transition delay faults. 10,000 patterns were generated using n-detection mode with dynamic compaction. Then, we performed random fill and minimum transition fill of X's respectively to generate two types of test sequences with different characteristics. The circuit was virtually divided into 4×4 square layout blocks with the same size when creating the power and temperature profiles. In order to mimic the behavior of complex and large circuit, we did the following transformation so as to match to the predicted values in ITRS [15] when we ran thermal simulations. We first enlarged the circuit size to 140 mm^2, and then elevated the power density to 2.14 W/mm^2 while preserving the power variations of the original circuit. These values are the predicted circuit size and power density for the year of 2020 in ITRS [15].

Figure 2 shows the temperature profiles for $b17$ obtained by the above flow. In these examples, we assumed that the initial value starts from a steady state temperature for highlighting its variations during test. The steady state temperature is the temperature that converged to when the circuit constantly consumes the average power in the power profile. Figure 2(a) shows that the test sequence with random fill leads to a significant spatial temperature variations. On the other hand, Figure 2(b) shows that the test sequence with minimum transition fill causes large amount of temporal variations as well as spatial variations. We could observe no apparent similarity among any temperature profiles of the layout blocks. These trends are the same as those of the test sequences with 0-fill and 1-fill in our preliminary experiments.

Figure 3 shows the temperature-induced delay sensitivity of a ring oscillator (27 stages, 180nm process technology) [6]. It shows that there is approximately 170 ps delay increase in timing for every 10°C rise in temperature. Back to Figure 2, the maximum spatial temperature difference among layout blocks is around 20°C



(a) Random fill (b) Minimum transition fill

Figure 2. Temperature profiles for $b17$.

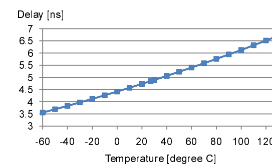


Figure 3. Temperature-induced delay sensitivity.

in both cases, and the maximum temporal temperature difference over time exceeds 15°C in the minimum transition fill. This suggests that there may exist 300-340 ps spatial and temporal delay variations during test.

These examples indicate that (1) we need a lot of sensors to collect temperature profiles accurately and (2) even if the accurate temperature profiles during test are available, it is difficult to eliminate the temperature induced delay variations from the measured delay values since they are complicated. These examples motivate the need of thermal-uniformity during test for accurate failure prediction.

3 Power and Temperature Characterization

In this section, we discuss the characteristics of the power consumption and temperature during test, and define some metrics to evaluate temperature variations.

Although the power consumption at each cell in a circuit varies significantly fast at every clock-cycle during test, it is known that the temperature does not change as rapidly as the power consumption does. Typical time intervals for temperature changes over time are known to be in the order of milliseconds [10]. Therefore, we use the power with pattern-accuracy (i.e., an average power between two capture cycles for each test pattern as our temporal power resolution. For the spatial power resolution, we use the power consumption with block-accuracy (i.e., one power value per layout block), since a intra-block heating is a localized feature, which occurs much faster than a circuit-wide heating.

In this paper, we use the term "temperature for test pattern i in block j, $T_{i,j}$" to denote the temperature in block j at the time when the response for i is captured. We also use the following metrics to evaluate the temperature variations during test for a circuit with N blocks and for a given test sequence of M test patterns.

- Spatial variance in temperature for test pattern i

  $$V_T^{spa}(i) = \frac{1}{N}\sum_{j=1}^{N}(T_{i,j} - T_{i,ave})^2$$

  where $T_{i,ave}$ is the average temperature among all the blocks for test pattern i.

- Spatial variance in temperature for the given test sequence

  $$V_T^{spa} = \frac{1}{M}\sum_{i=1}^{M} V_T^{spa}(i)$$

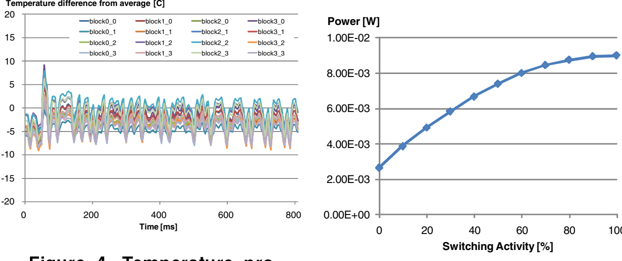


Figure 4. Temperature profile for $b17$ after thermal-uniformity-aware X-fill.

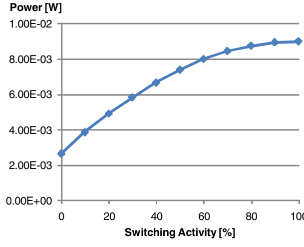


Figure 5. An example of the static power analysis.

- Temporal variance in temperature of block j

  $V_T^{temp}(j) = \frac{1}{M}\sum_{i=1}^{M}(T_{i,j} - T_{steady,j})^2$

  where $T_{steady,j}$ is the steady state temperature in block j.
- Temporal variance in temperature for the circuit

  $V_T^{temp} = \frac{1}{N}\sum_{j=1}^{N} V_T^{temp}(j)$

4 Test Pattern Optimization for Spatial and Temporal Thermal-Uniformity

4.1 Overview

In this section, we present a test pattern optimization method to reduce temperature-induced delay variations for accurate failure prediction. The goal of this paper is minimizing the spatial variance in temperature V_T^{spa} and the temporal variance in temperature V_T^{temp} defined in the previous section simultaneously. The optimization problem for spatial and temporal thermal-uniformity P_{STTU} is formally defined as follows.

Definition 1 P_{STTU}: Given a circuit with a test sequence S_X (an ordered test set) of M test patterns with unspecified bits (X's), physical information (xy-coordinates of each cells) and number of layout blocks N for temperature analysis, determine X-fill values for the unspecified bits and an order of the test patterns so that the spatial temperature variance V_T^{spa} and the temporal temperature variance V_T^{temp} are minimized.

We solve P_{STTU} applying the following two steps.

Step1: Thermal-Uniformity-Aware X-Filling

The proposed method starts with a test sequence S_X including unspecified bits (X's) generated by a commercial ATPG. As the 1st step, the thermal-uniformity-aware X-filling technique [13] is performed to obtain a test sequence S_{fill} of test patterns with fully specified bits. In this step, after test pattern $i-1$ is completed, all of the X's bits for each test pattern i are specified so as to minimize the spatial temperature variance for the test pattern $V_T^{spa}(i)$ while preserving the power and temperature at relatively low level. Figure 4 shows the temperature profile for $b17$ after Step1. Compared to the temperature profiles of the random fill or minimum transition fill shown in Figure 2, the spatial temperature variation is reduced significantly and each block has a high correlation in temperatures with other blocks. However, temporal temperature variations (around 20°C in this example) are still remaining.

Step2: Thermal-Uniformity-Aware Test Pattern Ordering

The objective of this step is determining an order of the test patterns in S_{fill} with fully specified bits so that the temporal temperature variance V_T^{temp} is minimized while preserving the spatial temperature variance V_T^{spa} achieved in the 1st step. The main idea is to adopt a sub-sequence-based ordering strategy, not pattern-by-pattern ordering, and minimize the number of sub-sequences used for the ordering. The spatial thermal-uniformity is valid only for the current order in S_{fill} obtained in Step1. More precisely, they are valid only for the current response-pattern pair which are simultaneously shifted during test in the current order. Therefore, the sub-sequence-based ordering itself can preserve the spatial thermal-uniformity without any consideration if the length of sub-sequences is long enough, and focus our efforts on minimizing the temporal temperature variance. This simplification reduces the computational cost of this step significantly since thermal simulation is iteratively used (explained later).

Since the thermal-uniformity-aware X-filling technique was already proposed in [13], we don't mention the detail of the technique in this paper, and we focus on the procedure of the test pattern ordering method, which will be explained in the next section.

4.2 Thermal-Uniformity-Aware Test Pattern Ordering (Step2)

The proposed thermal-uniformity-aware test pattern ordering procedure consists of the following three steps: (1) *Power Profile Estimation*, (2) *Test Sequence Partitioning* and (3) *Sub-Sequence-based Ordering*. The detailed procedure for each step is explained in the following sub-sections.

Step 2.1: Power Profile Estimation

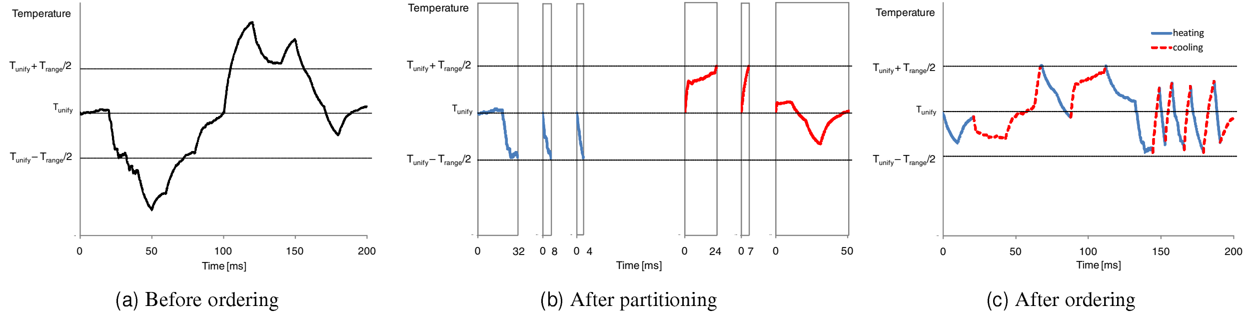
The first step is a generation of power profile for the test sequence S_{fill} obtained in Step1. The power profile for S_{fill} is required to create thermal profiles of each sub-sequence and the final test sequence in the following steps. Since we target the test sequence with spatial thermal-uniformity and adopt a sub-sequence-based ordering, the spatially global power metric (i.e., single power which is summation of all the block-accurate power consumption) can be used to evaluate the temperature profile of the circuit as shown in Figure 6. However, the flow using the logic and power simulations shown in Figure 1 is still computationally expensive to create power profile. Therefore, we use a simple and accurate power estimation scheme that consists of the following two steps.

1. Pattern-Independent Static Power Analysis: Using the static power analysis mode in power simulator, we first estimate the average power consumption when SA of every FF in the circuit is assumed to be d%, where d is varied from 0% to 100% with 10% increment. The SA of each FF is propagated to logic cells for quick calculation of the average power consumption of a circuit. The calculated power value includes dynamic and leakage power of FFs as well as those of logic cells. An example of the power analysis results are shown in Figure 5.

2. Linear Interpolation: We calculate the SA for each test pattern, and estimate the power consumption using the information of SA as follows. Suppose that the two adjacent reference SAs assumed in the static power analysis are given by the coordinates (x_a, y_a) and (x_b, y_b) where x and y represent SA and corresponding power consumption, respectively. For a pattern i with switching activity SA_i in the interval (x_a, x_b), the corresponding power consumption P_i is linearly interpolated by the following equation.

$$P_i = y_a + (SA_i - x_a) \times \frac{y_b - y_a}{x_b - x_a}$$

Step 2.2: Test Sequence Partitioning



(a) Before ordering (b) After partitioning (c) After ordering

Figure 6. Ordering example.

The goal of this step is to divide S_{fill} into the sets of sub-sequences $\{s_1, s_2, ..., s_K\}$ so that (1) the temperature profile of each sub-sequence s_i is within a specified temperature range T_{range} and (2) the number of sub-sequences K is minimized. In order to ensure that the temperature profile of each sub-sequence does not exceed T_{range}, thermal simulation is iteratively performed when we divide S_{fill}. Figures 6(a) and (b) show examples of temperature profiles before and after the test sequence partitioning, respectively. T_{range} is a key parameter to control the temporal variance in temperature V_T^{temp}, and we will evaluate the relation between T_{range} and V_T^{temp} in the experimental results section. The procedure of this step is as follows.

1. Calculate steady state temperature T_{unify} for S_{fill} by thermal simulation using the power profile PF estimated in the previous step.
2. Set lower and upper bound temperatures T_L and T_U as follows: $T_L = T_{unify} - T_{range}/2$ and $T_U = T_{unify} + T_{range}/2$.
3. Repeat Processes 4 and 5 until PF becomes empty.
4. Run thermal simulation from T_{unify} using PF to find the first pattern i so that temperature for i "T_i" becomes less than T_L or more than T_U.
5. Cut the first $i-1$ patterns from S_{fill} as new sub-sequence s, and remove the corresponding power values from PF. Classify s into the group of *heating* sub-sequences if $T_i > T_{unify}$. Otherwise, classify s into the group of *cooling* sub-sequences. Similarly, classify s into the group of *long* sub-sequences if the length of the sub-sequence is more than the specified constant L_{th}. Otherwise, classify s into the group of *short* sub-sequences.
6. Repeat Process 7 until the numbers of *heating* and *cooling* sub-sequences become the same.
7. Select the longest sub-sequence, and divide it into two sub-sequences: the first L_{new} (a specified constant) patterns and the remaining patterns. Update the status of the two sub-sequences: *heating* or *cooling*, and *long* or *short*.

In the above procedure, when we create a new sub-sequence, we always run the thermal simulation from the steady state temperature T_{unify} because of the following reasons. Since T_{unify} is calculated from the average power consumption in PF, it is close to the average temperature during test as shown in Figure 6(a). Therefore, it is suitable as the uniform temperature and we can guarantee that the same number of *heating* and *cooling* sub-sequences are created.

The sub-sequences divided by the above procedure have the following thermal characteristics. Every *short* sub-sequence creates monotonic and rapid temperature profile as shown in Figure 6(b). If a sub-sequence is classified into the group of *heating* such as sub-sequence 11 (sub11) in Figure 6(b), the temperature profile shows monotonic and rapid increase. Otherwise, it shows monotonic and rapid decrease such as sub-sequences 2 (sub2) and 3 (sub3) shown in Figure 6(b). On the other hand, *long* sub-sequences create non-monotonic temperature profile. Even if it is classified into the group of *heating* (*cooling*), the temperature profile might show the rapid or moderate decrease (increase) at the beginning.

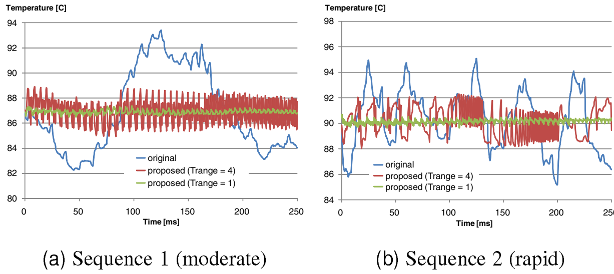
Step 2.3: Sub-Sequence-based Ordering

The third step is an ordering of the sub-sequences divided in Step2.2. The basic strategy of the proposed method is to order the *heating* and *cooling* sub-sequences in an interleaving manner as shown in Figure 6(c) since it is guaranteed that the numbers of *heating* and *cooling* sub-sequences are identical in the partitioning step. We determine the order one by one sequentially from the beginning. Suppose that we have already determined the first $i-1$ sub-sequences. When we determine i th sub-sequence, we check the thermal characteristic (*heating* or *cooling*) of $i-1$ th sub-sequence. Then, we always select a sub-sequence that has opposite thermal characteristic to it. We also check the current temperature T_{cur} (i.e., temperature at the end of $i-1$ th sub-sequence) using thermal simulation. If T_{cur} is close to T_U or T_L, we select a *short* sub-sequence so that the temperature at the end of i th sub-sequence is the closest to T_{unify}. If we select a *long* sub-sequence, the temperature might further increase beyond T_U or decrease below T_L at the beginning of i th sub-sequence since it creates non-monotonic temperature profile. If T_{cur} is close to T_{unify}, we can select any sub-sequence since the temperature profile of each sub-sequence is guaranteed to be within T_{range}. In this case, we select a longest sub-sequence. The procedure of this step is summarized as follows. In the procedure, we use a parameter α to indicate when we select a *long* or *short* sub-sequence.

1. Set $i = 1$, $T_{cur} = T_{unify}$.
2. Repeat Processes 3 to 5 until there is no more unselected sub-sequence.
3. If $T_L + \alpha \leq T_{cur} \leq T_U - \alpha$, select a sub-sequence s so that (1) s has the opposite thermal characteristic to $i-1$ th sub-sequence and (2) s is longest sub-sequence.
4. If $T_{cur} > T_U - \alpha$ or $T_{cur} < T_L + \alpha$, select a sub-sequence s so that (1) s has the opposite thermal characteristic to $i-1$ th sub-sequence, (2) s is *short* sub-sequence (if exists) and (3) the temperature at the end of s is closest to T_{unify}.

Table 1. Characteristics of benchmark circuits.

circuit	#FF	#gate	block	#sc	#pat	x ratio
b12	121	1,847	3 × 3	9	10,000	0.834
b14	215	5,579	3 × 3	9	10,000	0.871
b15	416	10,093	3 × 3	9	10,000	0.902
b17	1,317	32,871	4 × 4	16	10,000	0.961
b20	430	10,294	4 × 4	16	10,000	0.763
b21	430	10,165	4 × 4	16	10,000	0.787
b22	613	15,303	4 × 4	16	10,000	0.785



(a) Sequence 1 (moderate) (b) Sequence 2 (rapid)

Figure 7. Results of test pattern ordering for two handcrafted power profiles.

5. Set s as i th sub-sequence. Update T_{cur} by thermal simulation and set $i = i + 1$.

5 Experimental Results

In this section, we present experimental results using several ITC'99 benchmark circuits. We evaluate the following two main aspects of this paper.

- effectiveness of thermal-uniformity-aware test pattern ordering for reducing temporal temperature variation
- effectiveness of overall test pattern optimization method (including thermal-uniformity-aware X-filling) for reducing spatial and temporal temperature variations

We used the same experimental setup explained in Section 2 to generate a initial test sequence of 10,000 test patterns with unspecified bits for each circuit. The characteristics of the circuits used in the experiments are listed in Table 1. In our experiments, we used N square layout blocks with the same size for the power and temperature analyses and assumed each block has one scan chain that consists of all of the FFs in the block. For comparison purpose, the five existing X-filling techniques are applied to the initial test sequence without changing the order. They are the spatially thermal-uniformity-aware fill [13], random fill (rand-fill) and three low-power-oriented fills (minimum transition fill(min-fill), 0-fill and 1-fill). We also used the same flow explained in Section 2 to evaluate the final power and temperature profiles of each method.

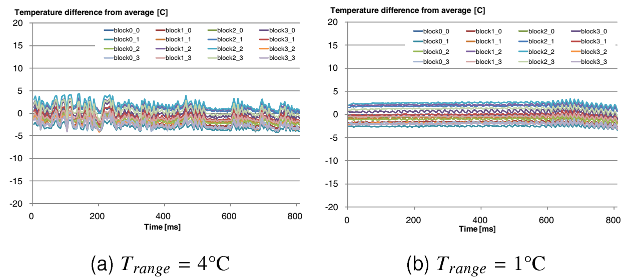
5.1 Test Pattern Ordering

We evaluate the effectiveness of the proposed test pattern ordering itself for reducing temporal temperature variation. It it noted that even though the proposed method consider both spatial and temporal temperature variations, here we focus on the evaluation of temporal variance V_T^{temp} in the *spatially global temperature* and relation between T_{range} and V_T^{temp} to highlight the effectiveness of the proposed ordering itself.

We handcrafted two spatially global power profiles of 50,000 patterns with different characteristics and the corresponding spatially global temperature profiles are shown as blue lines in Figure 7. In the same figure, the temperature profiles after ordering with

Table 2. Summary of test pattern ordering for two handcrafted power profiles.

	Profile 1 (moderate)			Profile 2 (rapid)		
#patterns	50,000			50,000		
	original	proposed		original	proposed	
T_{range}(°C)	-	4	1	-	4	1
T_{max} (°C)	93.40	88.89	87.37	95.09	92.30	90.70
T_{min} (°C)	82.26	84.94	86.48	85.17	88.03	89.64
T_{diff} (°C)	11.14	3.95	0.89	9.92	4.27	1.06
V_T^{temp}	10.86	0.81	0.04	5.38	1.32	0.04
ΔV_T^{temp} (%)	-	92.58	99.61	-	75.53	99.24
#sub-seq	1	272	1572	1	112	1180



(a) T_{range} = 4°C (b) T_{range} = 1°C

Figure 8. Temperature profiles for $b17$ after test pattern ordering.

$T_{range} = 4$ and $T_{range} = 1$ are shown as red and green lines, respectively. The results are also summarized in Table 2. ΔV_T^{temp} denotes the relative difference in temporal variance between "original" and "proposed", and "#sub-seq" denotes the number of sub-sequences created by the proposed method. From the results, we can observe that the temperature profiles are effectively controlled by the proposed method and the temporal variances are drastically reduced. The smaller T_{range} is, the more the number of sub-sequences is to reduce the temporal variation. In case of $T_{range} = 1$, the reduction is more than 99% for both power profiles. Another important observation is that minimizing the temporal variance also leads to a reduction of peak temperature. In case of $T_{range} = 1$, we obtained 6.0°C and 4.1°C reduction for the profile 1 and 2, respectively.

5.2 Overall Test Pattern Optimization

In this section, we evaluate the overall test pattern optimization method (including both the thermal-uniformity-aware X-filling [13] and ordering presented in Section 4.2) in terms of the spatial variance, temporal variance and peak temperature.

First, we show the block-accurate temperature profiles of the final test sequences for $b17$ obtained by the proposed method in case of $T_{range} = 3$ and $T_{range} = 1$ in Figure 8. Compared to the temperature profile before the ordering procedure shown in Figure 4, we can see the following observations. First, even though we ignore the spatial variation and use the spatially global power and temperature metrics in the ordering procedure to reduce the computational cost, the proposed ordering method can preserve the spatial thermal-uniformity. Second, the spatially global power and temperature metrics are effectively controlled the temperature profiles of all the layout blocks in the ordering procedure.

Then, we highlight the relative differences in "spatial variance ΔV_T^{spa}", "temporal variance ΔV_T^{temp}" and "peak temperature ΔT_{max}" between the proposed method and the existing techniques. Table 3 summarizes the results for all the benchmark circuits used in the experiments. In comparison with the spatially thermal-uniformity-aware fill [13], we observe that a significant reduction in temporal variance was obtained by the proposed test pat-

Table 3. Reduction in spatial and temporal temperature variance and maximum temperature by the proposed method.

circuit	T_{range} (°C)	[13] ΔV_T^{spa} (%)	[13] ΔV_T^{temp} (%)	[13] ΔT_{max} (%)	min-fill ΔV_T^{spa} (%)	min-fill ΔV_T^{temp} (%)	min-fill ΔT_{max} (%)	0-fill ΔV_T^{spa} (%)	0-fill ΔV_T^{temp} (%)	0-fill ΔT_{max} (%)	1-fill ΔV_T^{spa} (%)	1-fill ΔV_T^{temp} (%)	1-fill ΔT_{max} (%)	rand-fill ΔV_T^{spa} (%)	rand-fill ΔV_T^{temp} (%)	rand-fill ΔT_{max} (%)
b12	3	4.9	29.9	0.6	93.7	72.9	-9.3	95.4	78.4	-9.3	92.9	75.6	-7.8	86.1	-150.6	7.3
	1	17.5	80.6	1.4	94.6	92.5	-8.4	96.0	94.0	-8.4	93.9	93.2	-7.0	88.0	30.5	8.0
b14	3	3.1	40.2	3.2	72.2	69.4	-9.2	69.1	81.0	-9.9	74.3	77.9	-4.7	69.5	-98.3	11.8
	1	7.3	90.8	5.0	73.4	95.3	-7.3	70.4	97.1	-8.0	75.4	96.6	-2.9	70.8	69.4	13.3
b15	3	5.4	82.3	7.5	55.5	83.6	-6.8	62.8	86.6	-6.7	61.9	87.8	-6.3	73.8	-298.7	14.9
	1	6.3	97.5	8.8	56.0	97.7	-5.3	63.2	98.1	-5.2	62.3	98.3	-4.7	74.0	43.7	16.2
b17	3	2.5	69.5	4.7	86.5	55.4	-27.6	87.7	68.4	-24.5	88.4	59.4	-26.8	91.8	-857.9	21.0
	1	3.4	96.2	5.9	86.7	94.5	-26.1	87.8	96.1	-23.0	88.6	95.0	-25.3	91.9	-18.3	21.9
b20	3	0.7	37.9	0.9	59.9	78.7	-10.2	66.9	79.4	-6.4	67.8	79.1	-7.2	69.4	-176.4	8.9
	1	2.2	86.6	1.8	60.5	95.4	-9.1	67.4	95.5	-5.4	68.3	95.5	-6.1	69.8	40.3	9.7
b21	3	3.3	61.2	1.3	68.6	76.3	-8.3	70.8	82.5	-8.0	65.8	79.0	-6.6	58.3	-205.4	9.1
	1	4.6	93.6	1.4	69.0	96.1	-8.2	71.2	97.1	-7.9	66.2	96.5	-6.5	58.9	49.6	9.2
b22	3	1.4	57.0	0.8	74.2	72.8	-12.1	76.8	75.6	-9.0	79.3	74.3	-5.3	72.0	-214.5	6.4
	1	3.1	94.3	2.2	74.7	96.4	-10.4	77.2	96.8	-7.4	79.6	96.6	-3.8	72.5	58.6	7.8
average	3	3.0	54.0	2.7	72.9	72.7	-11.9	75.6	78.8	-10.5	75.8	76.2	-9.2	74.4	-286.0	11.3
	1	6.3	91.4	3.8	73.6	95.4	-10.7	76.2	96.4	-9.3	76.3	96.0	-8.0	75.1	39.1	12.3

tern ordering. We obtained 54% and 91% reduction on average in temporal variance in case of $T_{range} = 3$ and $T_{range} = 1$, respectively. Moreover, we obtained 3% and 6% reduction on average even in spatial variance in case of $T_{range} = 3$ and $T_{range} = 1$, respectively, In comparison with the three low-power-oriented X-fill techniques, we obtained around 75% reduction on average in spatial variance. Furthermore, we obtained more than 72% and 95% reduction on average in temporal variance in case of $T_{range} = 3$ and $T_{range} = 1$, respectively. Finally, in comparison with random-fill, around 75% reduction on average in spatial variance was obtained. Generally, temporal variation of random-fill is quite small as shown in Figure 2(a). However, in case of $T_{range} = 1$, we obtained 39% reduction on average even in the temporal variance. In terms of peak temperature, the proposed method incurred 8 to 12% increase on average compared to the three low-power-oriented X-fill techniques. However, they are still 3.2% and 11.8% lower than those of [13] and random-fill on average, respectively. The results show that the proposed test pattern optimization method effectively reduces the spatial and temporal variations with a little increase in peak temperature.

6 Conclusions

In this paper, we have shown the importance of achieving spatial and temporal thermal-uniformity for accurate failure prediction. We have presented a novel test pattern optimization method for thermal-uniformity, which minimizes spatial and temporal temperature variations on a circuit. Consequently, it makes easy to eliminate the temperature-induced delay variations from the measured delay values, which is important for accurate failure prediction. Experimental results for ITC'99 benchmark circuits have shown the effectiveness of the proposed method compared to the existing approaches. Since the proposed method consists of the thermal-uniformity-aware X-filling and test pattern ordering for given test sequences with unspecified bits, it is easily embedded into the current design flow without any loss of fault coverage.

Acknowledgments

This work was supported in part by Japan Society for the Promotion of Science (JSPS) under Grants-in-Aid for Young Scientists(B)(No.21700059). The authors would like to thank Prof. Seiji Kajihara in Kyusyu Institute of Technology, Prof. Yukiya Miura in Tokyo Metropolitan University and Prof. Satoshi Ohtake in Nara Institute of Science and Technology for their invaluable discussions.

References

[1] Y. Li, S. Makar, and S. Mitra, "CASP: Concurrent autonomous chip self-test using stored test patterns," in *Proc. Design, Automation, and Test in Europe*, pp. 885–890, Mar. 2008.

[2] H. Inoue, Y. Li, and S. Mitra, "VAST: Virtualization-assisted concurrent autonomous self-test," in *Proc. International Test Conference*, pp. 1–10, Oct. 2008.

[3] M. Agarwal, B. C. Paul, M. Zhang, and S. Mitra, "Circuit failure prediction and its application to transistor aging," in *Proc. VLSI Test Symposium*, pp. 277–284, May 2007.

[4] T. W. Chen, K. Kim, Y. M. Kim, and S. Mitra, "Gate-oxide early failure prediction," in *Proc. VLSI Test Symposium*, pp. 111–118, May 2008.

[5] Y. Li, Y. M. Kim, E. Mintarno, D. S. Gardner, and S. Mitra, "Overcoming early-life failure and aging for robust systems," *IEEE Design and Test of Computers*, vol. 26, pp. 28–39, Nov./Dec. 2009.

[6] Y. Sato, S. Kajihara, Y. Miura, T. Yoneda, S. Ohtake, M. Inoue, and H. Fujiwara, "A circuit failure prediction mechanism (DART) for high field reliability," in *Proc. International Conference on ASIC*, pp. 20–23, Oct. 2009.

[7] H. Yi, T. Yoneda, M. Inoue, Y. Sato, S. Kajihara, and H. Fujiwara, "Aging test strategy and adaptive test scheduling for SoC failure prediction," in *Proc. International On-Line Testing Symposium*, pp. 21–26, Jul. 2010.

[8] Y. Li, O. Mutlu, and S. Mitra, "Operating system scheduling for efficient online self-test in robust systems," in *Proc. International Conference on Computer-Aided Design*, pp. 201–208, Nov. 2009.

[9] A. B. Baba and S. Mitra, "Testing for transistor aging," in *Proc. VLSI Test Symposium*, pp. 215–220, May 2009.

[10] S. A. Bota, J. L. Rossello, C. D. Benito, A. Keshavarzi, and J. Segura, "Impact of thermal gradients on clock skew and testing," *IEEE Design and Test of Computers*, vol. 23, pp. 414–424, Sep./Oct. 2006.

[11] S. Bahukudumbi and K. Chakrabarty, "Test-pattern ordering for wafer-level test-during-burn-in," in *Proc. VLSI Test Symposium*, pp. 193–198, Apr. 2008.

[12] S. Bahukudumbi and K. Chakrabarty, "Power management for wafer-level test-during-burn-in," in *Proc. Asian Test Symposium*, pp. 231–236, Nov. 2008.

[13] T. Yoneda, M. Inoue, Y. Sato, and H. Fujiwara, "Thermal-uniformity-aware x-filling to reduce temperature-induced delay variation for accurate at-speed testing," in *Proc. VLSI Test Symposium*, pp. 188–193, Apr. 2010.

[14] Nangate, "Nangate 45nm open cell library." http://www.nangate.com/.

[15] "International technology roadmap for semiconductors, 2007 edition."

Test Scheduling for 3D Stacked ICs under Power Constraints

BreetaSenGupta Urban Ingelsson Erik Larsson
Department of Computer and Information Science
Linköping University
SE-581 83 LINKÖPING, SWEDEN
Email: (breeta.sengupta, urban.ingelsson, erik.larsson) @liu.se

Abstract- **This paper addresses test application time (TAT) reduction for core-based 3D Stacked ICs (SICs). Applying traditional test scheduling methods used for non-stacked chip testing where the same test schedule is applied both at wafer test and at final test to SICs, leads to unnecessarily high TAT. This is because the final test of 3D-SICs includes the testing of all the stacked chips. A key challenge in 3D-SIC testing is to reduce TAT by co-optimizing the wafer test and the final test while meeting power constraints. We consider a system of chips with cores equipped with dedicated BIST-engines and propose a test scheduling approach that reduces TAT while meeting the power constraints. Depending on the test schedule, the control lines that are required for BIST can be shared among several BIST engines. This is taken into account in the test scheduling approach and experiments show significant savings in TAT.**

I. INTRODUCTION

Integrated circuits (ICs) with multiple chips (dies), so called 3D Stacked ICs (SICs), have recently attracted a fair amount of research [1-5]. A 3D-SIC is obtained by stacking and bonding individual chips, which are connected by Through-Silicon Vias (TSVs). Due to imperfections in IC manufacturing, each individual chip must be tested. Recent research has addressed test architecture design for 3D-SICs [6], testing the TSVs [1-6] and 3D-SIC-specific defects [1, 2], but no previous work has addressed test scheduling under power constraints for 3D-SICs, which is the topic of this paper.

Testing each individual chip is required for both 3D-SICs and traditional non-stacked ICs. While IC packaging is costly [7], each chip is tested twice: (1) in wafer sort test, where the bare die is tested (pre-bond test), and (2) in final test where the packaged IC is tested (post-bond test). For non-stacked chips the same test schedule is applied in both pre-bond and post-bond test. However, for a 3D-SIC the process is different. As will be discussed in this paper, applying the same test schedule for both pre-bond and post-bond tests in a 3D-SIC leads to sub-optimal Test Application Time (TAT). TAT is defined as the sum of the testing times for pre-bond tests and post-bond tests. TAT is a major part of the overall test cost. Hence it is important to schedule the tests for 3D-SIC such that TAT is minimized, which is addressed in this paper.

Much work has addressed test scheduling for non-stacked chips with the objective of minimizing TAT [8, 9]. For core-based systems where each core is to be tested, the main method of reducing TAT is to perform core tests concurrently. However, performing tests concurrently leads to higher power consumption than performing them sequentially. The test power consumption must be kept under control [9], to avoid false test positives due to voltage drop and damage due to overheating. For core-based systems, Chou *et al.*[9] proposed a method to schedule tests in sessions while taking resource conflicts and power consumption into account. A session is a group of tests that start at the same time. In the context of systems where each core has an dedicated Built-In Self Test (BIST) engine, all the core tests that are scheduled in the same session can be initiated using a single control line. As a rule, a low number of sessions is beneficial, since it leads to a low number of control lines and implies that several tests are performed concurrently, leading to a low TAT [8, 9, 10]. The studies in [8, 9, 10] address test scheduling for non-stacked chips under power constraints. However, no work has yet addressed test scheduling for 3D-SICs under power constraints, which is the topic of this paper. We propose a power constrained test scheduling approach, which considers a two-chip 3D-SIC design, consisting of cores, each equipped with a dedicated BIST-engine. In this context we present an analysis of the test scheduling problem in Section II leading to an approach in Section III. The experimental results are in Section IV and the conclusions are in Section V.

II. PROBLEM ANALYSIS

Figure 1 shows a chip with three cores where each core is tested by its BIST test. Associated with each test are the parameters test time and power consumption. The test controller, which is a Finite State Machine (FSM) determines when the test for each core is initiated. Figure 2 shows a test schedule for the tests of the three cores in Figure 1, which have been scheduled as per [8] where the TAT is minimized and the power consumption at any moment is less than the maximal allowed power consumption P_{max}, which is indicated by a horizontal line. The test schedules are represented with blocks for the core tests, where the height of a block is the power consumption for the test and the width of the block is the test time. The x-axis shows the time taken to perform the tests, and the y-axis marks the power consumption. Two types of constraints are considered for the test schedule. The first constraint type is resource constraints which determine that two tests are not to be performed concurrently and the second constraint type is a constraint regarding the maximum power consumption, P_{max}, which cannot be exceeded. The test schedule contains three sessions: Session1, Session2 and Session3, as marked in the figure. This chip is a single-die IC, so the same test schedule is applied at pre-bond test (wafer sort) and post-bond test (final test). $TAT = C1 + C1$, as the same test schedule is run twice.

Figure 3 shows a 3D-SIC where Chip1 (from Figure 1) is stacked on top of Chip2. The testing of the 3D-SIC requires pre-bond tests of Chip1 and Chip2 and a post-bond test of the stacked chip that tests the whole SIC by including tests for the cores in Chip1 and Chip2. While testing of TSVs is important,

the actual test time is fixed and relatively low. In this paper, the testing of TSVs is not addressed. The test durations and power consumption values for each core tests are provided in Table 1. The power constraint value is P_{max} = 20 units.

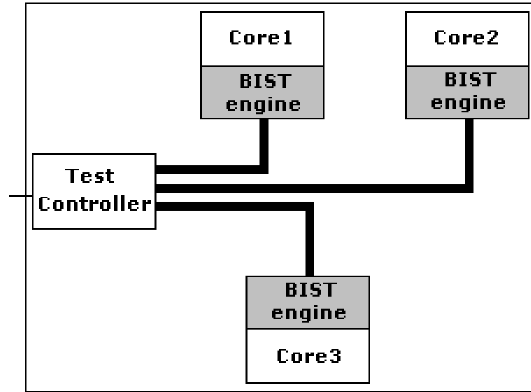


Figure 1: Chip with 3 cores.

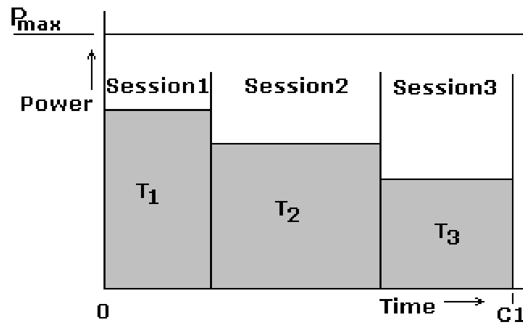


Figure 2: Test schedule of Chip1.

Prior to bonding chips into 3D-SIC each chip can be considered as individual non-stacked chips and the methods in [8, 9] apply for generating the pre-bond test schedules. Figure 4 shows an example of the pre-bond test schedules for the two chips, Chip1 and Chip2, from Table 1. The test schedule for Chip1 contains three sessions (Session1, Session2 and Session3) and the test schedule for Chip2 contains two sessions (Session4 and Session5) as shown in the figure. The test time for the schedules as obtained by [8] are *C1* and *C2* for Chip1 and Chip2, respectively.

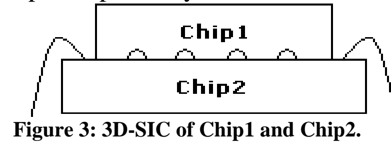


Figure 3: 3D-SIC of Chip1 and Chip2.

Table 1: Test time and power consumption for core tests in Chip1 and Chip2.

Chips	Tests	Duration	Power
Chip1	T1	5	15
	T2	8	12
	T3	6	9
Chip2	T4	2	7
	T5	7	8
	T6	5	9

Once the chips have been stacked, each core of the chips again requires testing. We define three different approaches for test scheduling depending on the available knowledge from the pre-bond test. In this paper, the three approaches are called Serial Processing (SP), Partial Overlapping (PO) and ReScheduling (RS).

In case the only knowledge of the pre-bond test schedules consist of the test time for the schedules and the fact that the pre-bond test schedules are within the power constraint, the limited knowledge available restricts the test schedules that are possible. In this case the post-bond tests are scheduled by *Serial Processing*, which is illustrated in Figure 4. With Serial Processing we mean that the test schedules of individual chips are run serially during post-bond testing. It should be noted that, no tests from different chips are performed concurrently, because otherwise we would risk exceeding the power limit. For Serial Processing, the time taken to run the post-bond test schedule is equal to the sum of the time taken to test the individual chips, which has been denoted by TAT_{SP} in the figure. For the schedule in Figure 4, $TAT_{SP} = C1 + C1 + C2 + C2$.

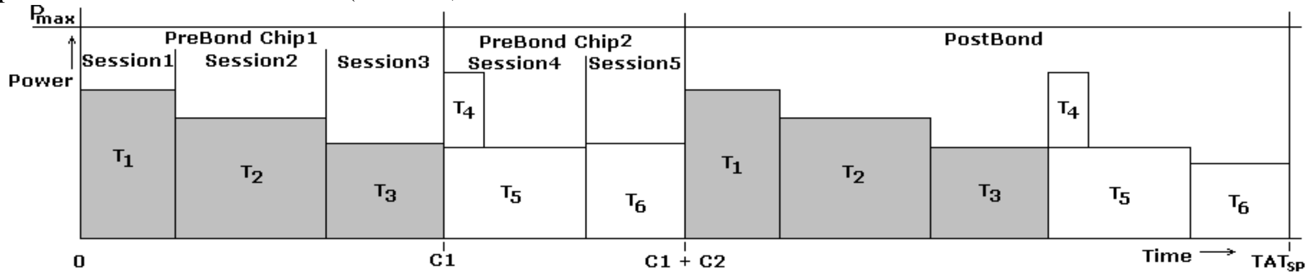


Figure 4: Serial Processing.

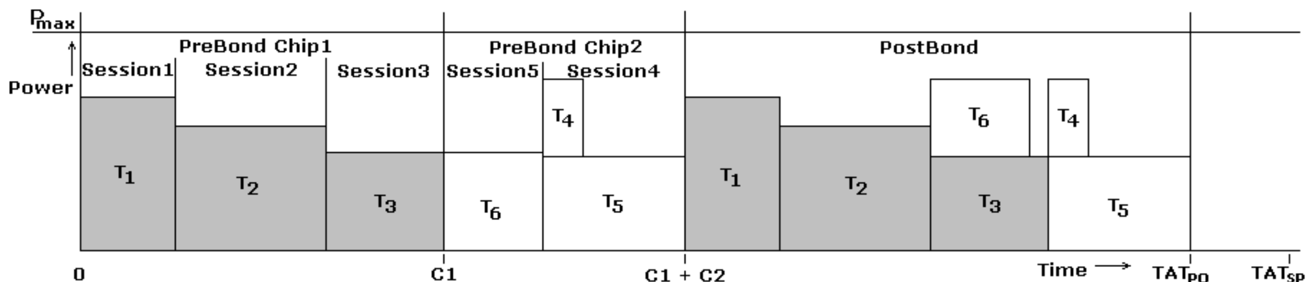


Figure 5: Partial Overlapping

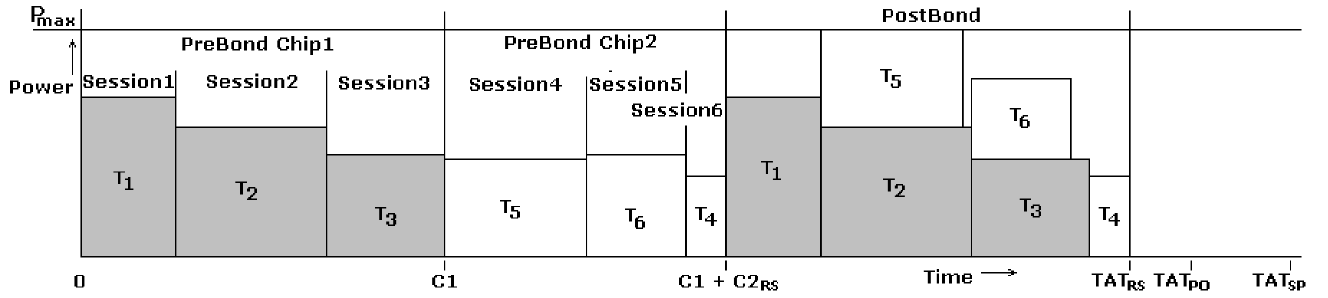


Figure 6: ReScheduling

If the maximum power reached by individual sessions and the test time for the sessions are known, post-bond scheduling by *Partial Overlapping* is possible. In Partial Overlapping, we utilize the knowledge of the test sessions to determine the power compatible test sessions of different chips that can be performed concurrently without exceeding the power constraint. Figure 5 shows the Partial Overlapping test schedule. In the post-bond test schedule, test T_3 of Chip1 (Session3) and test T_6 of Chip2 (Session4) are performed concurrently because they are power compatible. The pre-bond schedule of the chips remain unchanged, but there is a reduction in the TAT equal to the length of test T_6 (Session4) and the resulting TAT_{PO} is lesser than TAT_{SP}.

When full knowledge is available concerning individual tests and sessions of the pre-bond test schedules, *ReScheduling* of the existing schedules can be performed. In the ReScheduling approach, knowledge of the pre-bond test schedules is utilized to create a post-bond test schedule to reduce test time. ReScheduling may cause changes to the pre-bond schedules. In this context, changing the pre-bond schedule means to split a session and replace it with two new sessions. The benefit of splitting a session is that the two new sessions can be scheduled concurrently with sessions of the other chip during post-bond test, if that reduces TAT. Figure 6 depicts the result of the ReScheduling approach. In the original pre-bond test schedule (Figure 4), Session4 consisted of tests T_4 and T_5. In the post-bond test schedule, after rescheduling, test T_4 is performed serially with test T_1, while test T_5 is performed together with test T_2. This results in a reduction of the post-bond test time equal to the duration of test T_5. ReScheduling results in splitting Session4 and renumbering the sessions, as shown in Figure 6, Session4 is test T_5, Session5 is test T_6 and Session6 is test T_4. But because of the splitting of the original Session4, there is an increase in the pre-bond test time for Chip2 from *C2* to $C2_{RS}$. The increase is equal to the duration of test T_4, which is now performed serially with test T_5. Compared to SP, the reduction in TAT is equal to the sum of the durations of tests T_5 and T_6, minus the duration of test T_4. From the above, it can be seen that ReScheduling leads to lower TAT as compared to Serial Processing and Partial Overlapping, as is shown in Figure 6. However, in contrast to Serial Processing and Partial Overlapping, ReScheduling can lead to an increase in the number of control lines, as a result of splitting sessions. Our approach, detailed below, takes control lines into account.

III. PROPOSED APPROACHES

In this section we first detail the two approaches, Partial Overlapping (PO), and ReScheduling (RS), and then discuss the complexity of the approaches.

PO can be considered as a special case of RS. PO considers only the knowledge of individual sessions, and no sessions are split in the process. Hence, PO can be obtained by disregarding Step1 of RS.

RS is an approach in two steps as is described in the following. Before the first step, the initial pre-bond test for each chip is generated by the heuristic from [8]. Each session of the pre-bond test schedules is given a unique number.

Step1: Figure 7 shows an 11 stage process for implementing Step1 of RS. The key idea is to group the tests of two pre-bond test sessions from different chips in two post-bond test sessions such that the long tests are grouped together and the short tests are grouped together. This way, there will be one long test session and one short test session, instead of the previous two long sessions. The sum of the newly formed session duration is less than for the two original test sessions. In stage 1 of Figure 7, we consider two sessions, Sx and Sy, from the original pre-bond test schedules of two different chips, ChipX and ChipY, respectively. In stage 2, the tests of Sx and Sy are arranged in descending order of length in a list called *M*. A post-bond session, Sa, is produced in stage 3. Starting from the first test in *M*, *i.e.* the test with the longest test time, move the test from *M* to the post-bond session Sa, as shown in stage 4, and stage 5 checks if the total power of Sa is within the power constraint. Stage 4 and stage 5 are iterated until the power constraint is met. As soon as P_{max} is exceeded as a result of moving a test from *M* to Sa, that test is moved back to M (stage 6). A new post-bond session, Sb, is created with the remaining tests of list *M*, which is shown in stage 7. Post-bond sessions Sa and Sb contain all the tests from *M*. The pre-bond sessions Sx and Sy are split into test sessions (say Sxa, Sxb, Sya and Syb) according to how the tests were allocated in Sa and Sb. The modified TAT is calculated in stage 8. Stage 9 considers the TAT. If the new test schedule (pre-bond and post-bond) is shorter than the test schedule for SP, the value is included in Table 2 as in stage 10, as the entry for session Sx and session Sy. Otherwise, if there is no reduction the value is set to be zero.

The process described above is repeated for all possible combinations of two sessions from the pre-bond test schedules

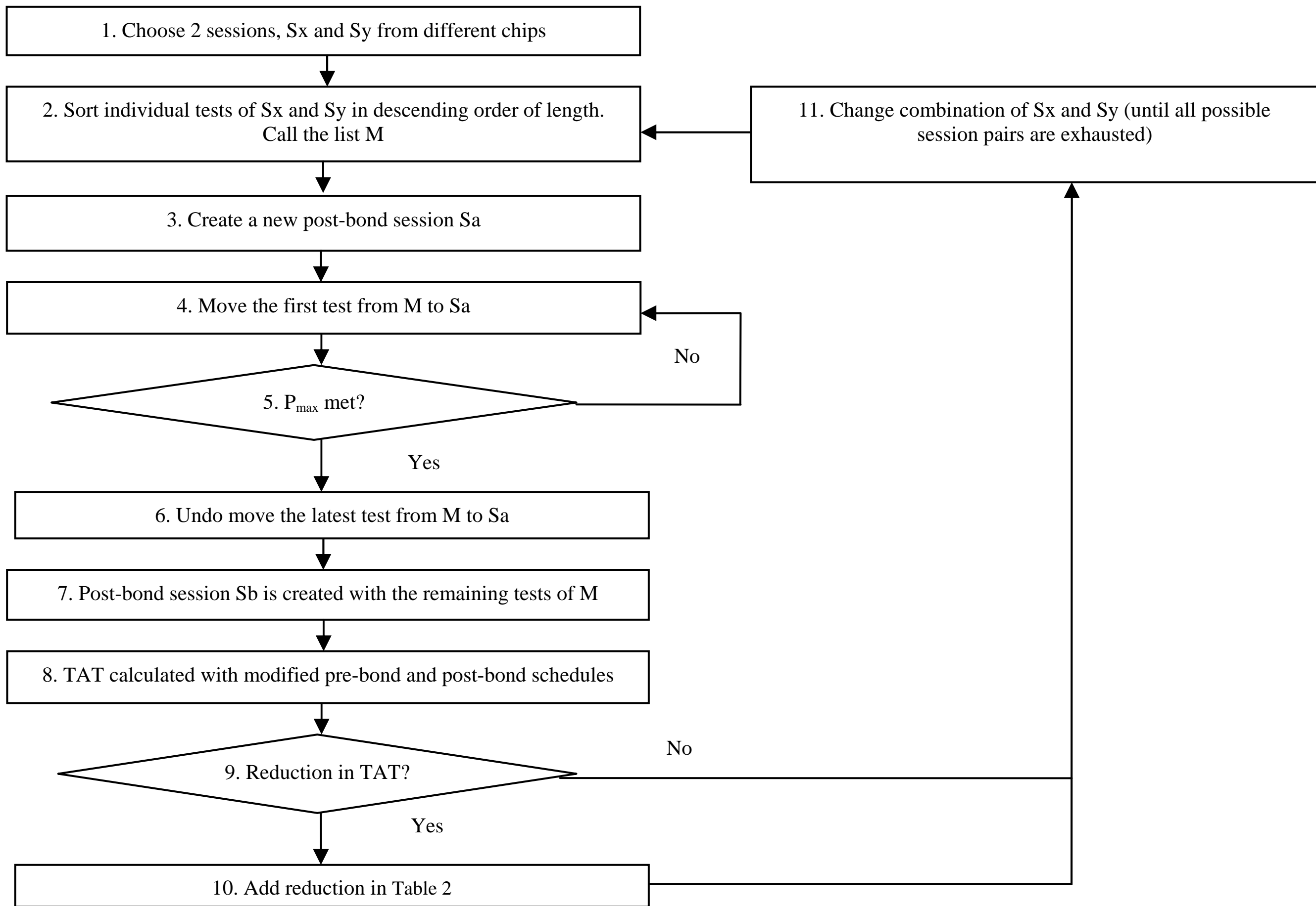


Figure 7 Flow diagram for test scheduling

of the two chips, as is shown in stage 11 of Figure 7.

A key observation from the above is that pairs of sessions can be handled independently. If combining a pair of sessions as described by stage 1 to stage 11 leads to a reduction in TAT compared to the test schedule in SP, a new test schedule can be constructed by combining several such session pairs. The total reduction in TAT can be summed up from the reductions in test time when all session pairs have been considered, while each session has been taken into account only once.

Step2: This step involves the calculation of the maximum reduction in TAT from SP to RS (or PO) by considering all possible session pairs. Table 2 shows the possible reduction in TAT as a result of rescheduling a session of ChipX, as denoted by the column number, with a session of ChipY of the corresponding row number. Given Table 2, a schedule is generated by rescheduling each session of one chip with different sessions of the other chip, such that every session is considered only once. The sessions that are not rescheduled are added to the final schedule without any modification. The objective is to find the combination of rescheduled session pairs, which give the minimum TAT. The values in Table 2 are obtained by rescheduling sessions of the example used during the problem formulation. For example, with respect to Figure 4, considering Session2 from Chip1 and Session4 from Chip2 results in a reduction of 3 time units on rescheduling (as discussed in Step1), compared to the time required to perform the original Session2 of Chip1 and Session4 of Chip2 sequentially, as in SP. In case of PO, where no sessions are split, the values in the table would either be zero (when the sessions are not power compatible), or equal to the length of the smaller session. For example, it was not possible to reduce TAT by combining Session1 with Session4 as marked by 0 in Table 2. The test schedule and the total reduction in TAT are obtained by rescheduling each session of ChipY, (Chip2 in the example) with sessions of ChipX (Chip1 in the example). As discussed before, tests from Session2 of Chip1 and tests from Session4 of Chip2 upon rescheduling, result in a reduction of 3 time units, while, Session5 of Chip2 with Session3 of Chip1 give a reduction of 5 time units. The sessions that result from the marked session pairs are included in the post-bond test schedule with the summed total of test time reduction adding up to $3 + 5 = 8$ time units. The test time of the rescheduled session pairs are added to the remaining sessions to give TAT. Thus, the final post bond schedule has Session1 in series with the combination of Session2 with Session4 and Session3 with Session5. Thus TAT is 54 time units, obtained by reducing 8 time units from TAT_{SP}, which has 31 time units in both pre-bond and post-bond.

Table 2: Maximum possible time reduction of sessions where ChipX and ChipY refers to the algorithm and Chip1 and Chip2 refers to the example.

		ChipX(Chip1)		
		Session1	Session2	Session3
ChipY(Chip2)	Session4	0	3	2
	Session5	0	0	5

Finding the combination of session pairs that give the minimum TAT on rescheduling requires comparison of all possible

combinations of session pairs, which is complex. To arrive at the complexity of exploring all possible schedules from Table 2, say ChipX and ChipY have x and y number of sessions respectively, and that $x \geq y$. Then there are x columns and y rows. The first row has x values to choose from. Once a value is chosen, the row and column to which the value belongs are ignored and there remains $x - 1$ values to choose from in the second row. Thus, as we process each row, the number of choices decreases by one. This accounts for a factorial function that describes the number of possible sets of session pairs. But, when $y - 1$ rows have been traversed the last value can be chosen from the remaining $x - y + 1$ columns. Thus, the total number of ways, N, in which values can be selected from Table 2, with each value from a unique row or column, is given by $N = (x - y + 1) \cdot y!$. Hence, for a total number of ten sessions each in two chips, N becomes as large as 3628800. Thus, it can be seen that the problem of selecting session pairs from Table 2 to explore all possible test schedules is difficult.

Existing heuristics can be applied to obtain a schedule from Table 2. In the following we describe the greedy heuristic that has been used. Prior to applying the heuristic, the rows are first sorted in descending order of the highest value in each row and then the same procedure is applied for the columns. After the table has been sorted, an arbitrary starting value is chosen from the table. The highest value from the neighboring row is then considered along with it. The process is continued until all rows are exhausted. The sum of all the values corresponding to the session pairs considered for rescheduling give the net reduction in test time. Sessions that were not joined with other sessions are added to the list of session pairs to form the schedule. The particular combination of session pairs that lead to the schedule correspond directly to the pre-bond and post-bond test schedules for the stacked 3D design. The combination of session pairs that gives the largest reduction in terms of TAT and an acceptable number of BIST control lines, as determined by the designer of the stacked 3D chip, can be considered as the final schedule.

To arrive to the final schedule, the heuristic is iterated K times, where K is the sum of the number of rows and columns, with different session pairs as starting point to produce a number of solutions that can be evaluated by the designer of the stacked 3D chip with regard to the acceptable number of control lines. This results in Table 3 for the considered example. Schedule1 is the result of combining Session2 with Session4 as well as Session3 with Session5. Schedule 2 is the result of combining Session2 with Session5.

Table 3: TAT reduction versus increase in number of BIST control lines

Schedule	1	2	3	4	5
TAT Reduction	8	2	3	5	0
BIST Control Line Increase	1	1	1	0	0

ReScheduling of sessions resulting in a reduction of TAT can lead to a corresponding increase in the number of BIST control lines due to splitting of sessions. Table 3 shows an example providing the reduction in TAT and the number of additional control lines for five of the test schedules produced by the proposed RS approach.

Complexity of the approach: Here we study the complexity of the RS approach. The approach consists of two steps, Step1 and Step2.

In Step1 of the RS approach, the tests from two sessions are initially sorted by their test durations and stored in the list M. The average time complexity for quick-sort is O($N \log N$) for N tests.

Step2 of the problem involves obtaining the maximum sum of individual elements from the matrix, taking one element from each row or column. As discussed, the solution space is large, so the greedy heuristic has been applied, which has a average time complexity of O($T \log T$), where T is the number of elements in the matrix. Prior to applying the heuristic, the rows and columns were sorted in descending order of the value of individual elements. The sorting here is also done by quick-sort, which has a complexity of O($\log T$). Step 2 is iterated K times, where K is the sum of the number of rows and columns of the corresponding matrix.

Thus the overall complexity of the approach is O($T \log T$).

IV. EXPERIMENTAL RESULTS

To demonstrate the benefits of the proposed test scheduling approach, this section describes an experiment to compare TAT achieved by Partial Overlapping (PO) and ReScheduling (RS) with TAT achieved by the straight forward Serial Processing (SP) approach, which is used as baseline. In the experiment, the power constraint is met and the number of BIST control lines required by different test schedules is taken into account. As the RS approach yields a table such as Table 3 with several different test schedule solutions where the acceptable number of control lines determines the final test schedule selection, the experiment is performed with the test schedule that results in the largest TAT reduction (8 time units in the case of Table 3). The initial pre-bond test schedules were generated by the approach in [8] and our approaches where applied for generating the post bond test schedule. The approach proposed in Section III was used to find the maximum reductions in TAT while considering the number of BIST control lines as the number of sessions in the example designs were in a reasonable range.

The experiments are performed with the circuits ASIC Z [10], System L [10] and Muresan [8] (marked by Z, L and M respectively in Table 4) and these circuits were used to create 3D-SICs. These designs are seen as single-die chips and have 9 [10], 14 [11] and 10 [8] cores, respectively. To make a 3D-SIC, two of the three single-die chips are combined. Groups of columns marked Chip1 and Chip2 respectively, show the chips that are combined into 3D-SICs, and groups of columns marked Chip1 & Chip2 and TAT contain experimental results regarding the combined 3D-SICs. To combine the Muresan design with ASIC Z or System L, such as in the fifth row of results in Table 4, adjustments were made on parameters because the values of those parameters in the original designs were given in different orders of magnitude. In the cases marked M* and M**, the parameter values (test time, test power consumption and power constraint) were scaled such that the pair of single-die chips that are combined into 3D-SICs

Table 4: Maximum possible reduction in time with increase in number of control lines. In the table: Z: ASIC Z, L: System L, M:Muresans' Design, SP: Serial Processing Time, PO: Partial Overlapping Time, RS: ReScheduling Time, R($=\frac{T_{SP}-T_{RS}}{T_{SP}}$): Reduction

	Chip1					Chip2				Chip1 & Chip2				TAT				Incr. in control lines
	Pre-bond Test					Pre-bond Test				Post-Bond Test				Pre-bond + Post-bond				
	T_{SP}	T_{PO}	T_{RS}	R (%)		T_{SP}	T_{PO}	T_{RS}	R (%)	T_{SP}	T_{PO}	T_{RS}	R (%)	T_{SP}	T_{PO}	T_{RS}	R (%)	%(orig)
Z	300	300	300	0	Z	300	300	300	0	600	560	560	6.7	1200	1160	1160	3.3	0 (6)
L	1374	1374	1374	0	L	1374	1374	1592	-15.9	2748	2107	1592	42.1	5496	4855	4558	17.1	3 (36)
M	26	26	27	-3.8	M	26	26	27	-3.8	52	52	48	7.7	104	104	102	1.9	20 (10)
Z	300	300	300	0	L	1374	1374	1374	0	1674	1374	1374	17.9	3348	3048	3048	9.0	0 (16)
Z	300	300	300	0	M*	520	520	520	0	820	780	780	4.9	1640	1600	1600	2.4	0 (8)
L	1374	1374	1374	0	M**	1040	1040	1040	0	2414	1824	1824	24.4	4828	4238	4238	12.2	0 (18)

have their parameter values in the same order of magnitude. It should be noted that this scaling of parameter values is only performed to enable the experiments. The results are collected in Table 4. The first group of four columns marked Pre-bond test for Chip1 show the test times for SP, PO and RS for the pre-bond schedules for Chip1. The fourth column in the group shows the relative reduction in pre-bond test time of RS compared to SP. It should be noted that a negative reduction is an increase. Similarly, the second group of four columns shows the Pre-bond test for Chip2. The third group of four columns marked Chip1 & Chip2, Post-bond test, show test time for the post-bond test schedule generated by the three approaches, and gives the relative amount of post-bond test time reduction achieved comparing the result for SP with the result for RS. The group of columns marked TAT includes the sum of the pre-bond test times and post-bond test times. The first three columns in the group of four show TAT for the SP, PO and RS approaches, respectively. The relative reduction in TAT is shown in the last of the four columns where RS is compared against SP. The right-most column of Table 4, shows the relative increase in the number of control lines that result from splitting sessions in the RS approach. The number of control lines for the SP approach is shown in parenthesis.

From Table 4, it can be seen that RS can achieve up to 42.1% reduction in the post-bond test schedule time in comparison to SP, when two chips of System L are stacked to form the 3D-SIC. This is for the 3D-SIC consisting of two System L chips. This result can be explained by a high power constraint, which enables a beneficial post-bond test schedule where many core tests are performed concurrently. In particular for the design with two System L chips, one session was split, resulting in an additional control line and an increase in the pre-bond test schedule duration. The reduction in TAT was 17.1%. It should be noted that other 3D-SICs consisting of two identical chips (such as the pair of ASIC Z chips) does not lead to the same result. For the 3D-SIC design made up by a pair of ASIC Z chips, TAT was reduced by 3.3% and RS and PO achieved the same result. This corresponds to a case when it is not possible to reduce TAT by splitting sessions. In the six experiments, only two experiments led to splitting of sessions, which increased the number of BIST control lines. For the other four experiments, the reduction in TAT was achieved without splitting sessions and the best result achieved without splitting sessions was 12.2% reduction in TAT.

V. CONCLUSION

In this paper, the problem of power-constrained test scheduling for 3D Stacked Integrated Circuits (SICs) has been addressed for the first time. It is shown that the test planning for 3D-SICs is different, compared to the test planning for non-stacked ICs, and requires specific test scheduling solutions. The paper proposes two test scheduling approaches, Partial Overlapping and ReScheduling that minimize test application time while taking power-constraints and the number of BIST control lines required to implement a test schedule into account. The two scheduling approaches and a straight forward approach (Serial Processing) have been implemented and experiments with several benchmarks show up to 17.1% reduction in test application time and an average reduction of 7.7% in test application time with a 3.8% average increase in the number of BIST control lines over the Serial Processing scheme.

REFERENCES

[1] E. J. Marinissen, Y. Zorian. Testing 3D Chips Containing Through-Silicon Vias. *IEEE ITC, paper ET1.1,* pp. 1-11, 2009

[2] H.-H. S. Lee and K. Chakrabarty. Test Challenges for 3D Integrated Circuits. *IEEE Design and Test of Computers, Special Issue on 3D IC Design and Test*, pp. 26-35, Oct 2009

[3] D. L. Lewis and H.-H. S. Lee. A Scan-Island Based Design Enabling Pre-bond Testability in Die-Stacked Microprocessors. *IEEE ITC*, paper 21.2, pp. 1-8, 2007

[4] X. Wu, P. Falkenstern, and Y. Xie. Scan Chain Design for Three-Dimensional Integrated Circuits (3D ICs). *ICCD*, pp. 208–214, 2007

[5] Y.-J. Lee and S. K. Lim. Co-Optimization of Signal, Power, and Thermal Distribution Networks for 3D ICs. *Electrical Design of Advanced Packaging and Systems Symposium*, pp. 163-166, 2008

[6] B. Noia, S. K. Goel, K. Chakrabarty, E. J. Marinissen and J. Verbree. Test-Architecture Optimization for TSV-Based 3D Stacked ICs. *IEEE ETS*, pp. 24-29, May 2010

[7] J. Verbree, E. J. Marinissen, P. Roussel and D. Velenis. On the Cost-Effectiveness of Matching Repositories of Pre-Tested Wafers for Wafer-to-Wafer 3D Chip Stacking. *IEEE ETS*, pp. 36-41, May 2010

[8] V. Muresan, X. Wang, V. Muresan and M. Vladutiu. Greedy Tree Growing Heuristics on Block-Test Scheduling Under Power Constraints, *JETTA*, pp. 61-78, 2004

[9] R. M. Chou, K. K. Saluja and V. D. Agrawal. Scheduling tests for VLSI systems under power constraints. *IEEE Trans. VLSI Systems,* vol. 5, no.2, pp. 175-185, June 1997

[10] Y. Zorian. A Distributed BIST Control Scheme for Complex VLSI devices. *IEEE VTS,* pp. 6–11, April 1993

[11] E. Larsson and Z. Peng. An Integrated Framework for the Design and Optimization of SOC Test Solutions, *JETTA*, *Special Issue on Plug-and-Play Test Automation for System-on-a-Chip*, vol. 18, no. 4, pp. 385-400, August 2002

Low Power Programmable Controllers for Reliable and Flexible Computing

Masahiro Fujita Hiroaki Yoshida Jaeho Lee
University of Tokyo
Tokyo, Japan
fujita@ee.t.u-tokyo.ac.jp

ABSTRACT

In order to guarantee higher reliability for embedded systems, there must be mechanisms of programmability even after the systems are shipped. Programmability can be utilized for rectifying designs in fields to accommodate designs bug fixes and changes of specifications. Moreover, even if there are some manufacturing faults or electrical errors, programmability in fields may be able to avoid to use those faulty portions of designs. One way to realize programmability in embedded systems is to introduce programmable controllers for the control parts of embedded systems. Even if the datapath parts of embedded systems are fixed, by changing control sequences in programmable controllers, some amount of bug fixes/specification changes as well as various faults and electrical errors can be overcome in fields. There have been researches which study for implementing programmable circuits on efficient high performance device, such as programmable loop accelerator [1]. However, it causes the energy overhead due to the access large size of memory. It is well known that utilizing memory hierarchy is effective for reducing the power consumption. In this paper, we demonstrate that the same technique is effective for programmable accelerators.

1. INTRODUCTION

Demands of customers for embedded system products with more functionality and higher performance is continuously increasing. Time-to-market is now a top priority when developing embedded systems. Customers are also very enthusiastic to have new products at their hand as soon as possible, shrinking life-cycles of products in the competitive market. Although significant amount of efforts are paid to verification processes of embedded systems before manufacturing them, some bugs can really escape such processes. Also, there can be changes of specifications in the later designs stages or even after manufacturing. Therefore, some sorts of programmability is essential in the developed systems to be reliable and flexible. For example, figure 1 shows a loop from the faad2 application, which is a commonly used free audio decoder for the Advanced Audio Coding (AAC) standard. Figure 1 a) shows an example of a bug fix in a loop. In this case, the code changes from

Submission to RASDAT 2011

VLSI Design and Eduation Cetner
University of Tokyo
2-11-16 Yayoi, Bunkyo-ku
Tokyo 113-0032 Japan
.

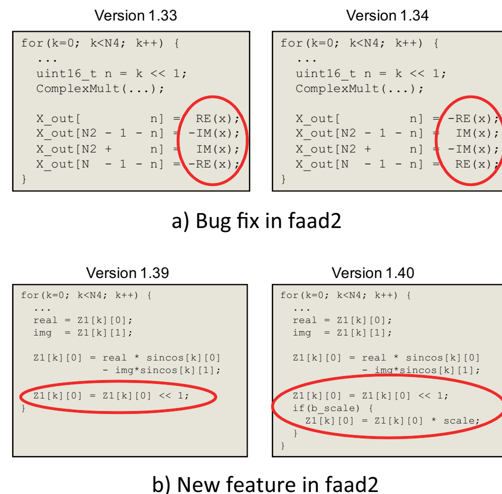


Figure 1: Feature addition and bugfix to mdct.c in faad2

version 1.33 to 1.34 consist of sign changes on the right hand sides of some assignment statements, as might occur in bug fixes. Figure 1 a) shows that between revisions 1.39 and 1.40 of the software, the loop has been modified with the addition of an if-clause, while the rest of the loop remains the same. This represents the addition of a new feature that requires certain new code in the loop. In either case, the number of operations does not change, but the communication among operations changes, and the hardware should be flexible enough to accommodate these. It can be seen that loops in real applications undergo minor changes over time. Typically, the bulk of the computation in the loop remains the same, but small changes need to be made to fix bugs or implement new features. Since the changes do not alter the loops significantly, it is possible to design custom hardware to accelerate the original loop, supporting just enough programmability to continue to accelerate the loop efficiently as the source code evolves with programmable controllers.

In development processes, changes to design specifications after manufacturing often occur as can be seen from the above. In such situations, we need to spend a lot of time and designers' efforts to revise the designs if they are implemented on traditional hardwired application specific integrated circuit (ASIC)s. Chip re-spin costs significantly in ASIC designs and the cost is often dominant in the entire development cost. Thus, the demand for programmable circuit after manufacturing is rapidly increasing. Programmability

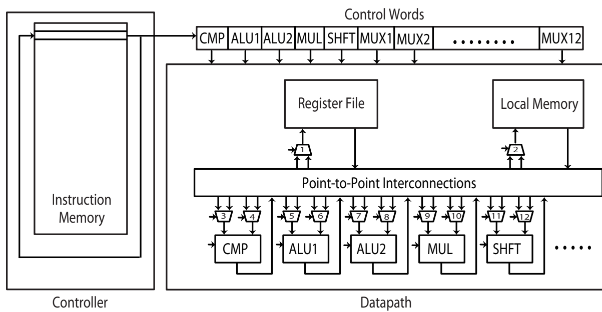


Figure 2: Template Architecture of Programmable Loop Accelerator

can provide a lot of cost saving by getting rid of high recurring cost during developments and provides fast time to market.

Compared to traditional hardwired ASIC, we can take the following advantages of programmable devices. First, we can revise their functionality whenever we want to in fields. Also, we can reuse existing circuit designs developed elsewhere. Hardware reuse decreases the entire development time so that we can shorten time to market. These are essential for reliable computing as well. The processors are also programmable circuits in which we can run any applications on it but they are very inefficient in terms of power consumption per tasks to be computed and sometimes can be tremendously slow for some types of applications compared to customised hardware. There are specialized processors for increasing performance and efficiency like DSP. However, they still cannot compete with custom hardwares in both of speed and energy efficiency. Currently, we are facing increase in capability of programmable devices such as Field Programmable Gate Array (FPGA). FPGAs offer flexible programmability which enables users to change circuit configurations after fabrication. However, these fully programmable devices are not highly efficient in both of speed and energy efficiency. They occupy large area, which generally means higher power consumption, and have limited performance. They cannot still satisfy the demand for the modern portable devices; to integrate several functionalities in small size with lower power.

As another solution, programmable hardware accelerators have been proposed to deal with the trade-offs between programmability, performance, implementation area and power consumption. There is a couple of previous existing programmable accelerators which use a microcode-based programmable controllers and a custom hardwired datapaths. NISC [2] [3] and programmable loop accelerator [1] [4] are good examples of current programmable accelerator. They use same programmable controllers which store instructions for datapaths in memory. NISC was created for shrinking time-to-market and to maximize circuit's productivity in order to cover broad applications with a single design methodology and framework. PLA has similar purpose to NISC but it tries to minimize the high performance and energy efficiency degradation and give programmability only for inside of loop because loops consume most resources in embedded system applications. Microcode-based programmable controller generates control words in every cycle. The generated control words are fed as inputs of corresponding components in datapaths. Users can specify different control words by modifying memory contents within programmable controllers.

However, programmable controllers which are not designed for single application consume more power than hardwired ASICs due to the use of memory for controller parts. In order to run multiple application in single hardware, larger size of memory is required and larger memory consumes more power. Especially for small devices, this increase in memory size cause a lot of energy overhead. There are techniques to reduce instruction delivery power by changing memory hierarchy. Cache [5], scratch-pad [6] [7] and filter cache [9] are methods that reduce energy of general processors. The organizations of programmable accelerators which instructions are directly moved to datapath are slightly different compared to those of general processors. However, the concept of reducing power consumption caused by instruction delivery from memory is the same. Memory hierarchy is changed in such a way that the power is reduced by accessing smaller memory instead of accessing only large memory. Therefore, in this paper, we would like to apply these general processors' power reduction methods to programmable loop accelerator and evaluate the amount of energy consumption savings.

The rest of this paper is organized as follows: In Section 2, we discuss related works which include the programmable loop accelerator and memory hierarchy for power reduction. In Section 3, we draw out a typical scenario based on a real example, which we can achieve power savings by introducing memory hierarchy, to describe the motivation of our work. In Section 4, we describe experimental result. Section 5 gives conclusions.

2. RELATED WORKS

2.1 Programmable Loop Accelerator

The circuit can be divided to controller and datapath. Controller is controlling operations of datapath in circuit. Sometimes, it is represented as Finite State Machine (FSM). In the case of controller, there are an output logic and a next-state logic in general. The output logic generates corresponding control signals for datapath and the next-state logic select a next state from candidates. Figure 2 shows how controller is organized and connected to datapath in programmable loop accelerator[4]. Controller in programmable loop accelerator (PLA) is composed of instruction memory block. One instruction from instruction memory generates control words and also includes next state's candidates in each state. Each of corresponding blocks in control words is connected to each component in datapath. Control words are control bits of functional units, multiplexers, and register files as shown at Figure 2. Datapath is a collection of store unit parts, interconnection parts, and functional units, such as registers, multiplexers, arithmetic logic units, and multipliers. We can see there are several components in datapath as described at Figure 2. Datapath performs data processing operations or computations.

A programmable loop accelerator needs to have a programmable unit in order to change the behavior of circuit. Therefore, controller in PLA is implemented by memory which is called instruction memory block in Figure 2. In contrast to hardwired controller in ASICs, users can modify a programmable controller by changing instructions in instruction memory on PLA. Instruction includes microcoded control signals which are equivalent to control words in Figure 2. If a user rewrites different contents in the instruction memory, the user can easily change the entire behavior of circuits in PLA. Instruction memory stores whole state's instructions which has same size as number of states in circuit. Also, addresses of instruction memory represents corresponding states. For example, address 0 is corresponding to first state, and address 1 is corresponding to second state, and so on. In each state, single control words is generated and is transfered to datapath within each cycle.

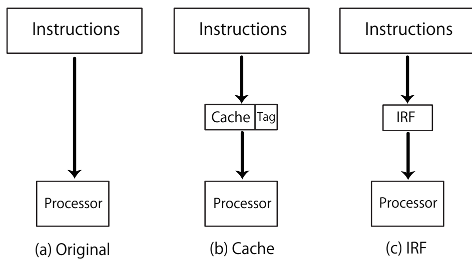


Figure 3: Instructions Delivery Methods

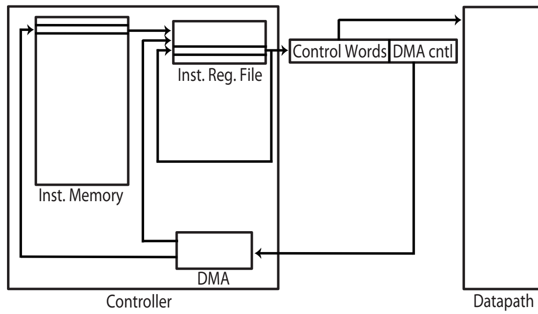


Figure 4: Template Architecture of PLA with IRF

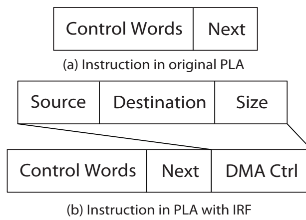


Figure 5: Instruction and DMA control

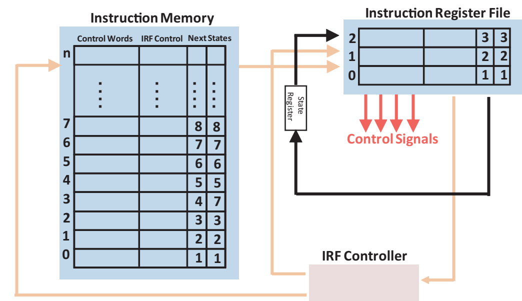


Figure 6: Microcoded Programmable Controller with IRF

In contrast to programmable controller implemented by memory, components at datapath are all hardwired in PLA. Functional units, interconnection parts, and storage units are all connected with hardwire which is same as datapath in ASICs. This is the reason that programmable loop accelerators guarantee high performance even though it gives programmability from programmable controller. In addition, datapath has to support enough functional units, register files, and interconnections to run multiple applications in PLA so it has more components compared to ASIC. However, there is almost no performance degradation since datapath is all hardwired. As datapath in programmable loop accelerator might have more components than ASIC, the size of memory in programmable controller is larger as well. In order to run multiple applications on a single hardware, the size of memory cannot help increasing. Also, the size of memory is dependent on the largest application. It is very difficult to optimize memory size especially for small applications which use a large memory. The large size of memory causes the overhead of energy consumption and it leads the energy inefficiency of PLA compared to ASIC.

2.2 Memory Hierarchy for Power Reduction

In previous section, we saw that programmable controller has a large energy overhead due to large size of memory. We know that if we use smaller size memories in programmable hardware, the energy consumption will be improved. In this section, we would like to discuss how we can reduce energy consumption of controllers. There are a couple of methods to reduce instruction delivery power of processors. Cache [5], scratch-pad [6] [7] and filter cache [9] are the common methods to reduce instruction delivery power by using temporal locality. We intend to review these methods for our reference and apply them to PLA in order to reduce power overhead. In Figure 3 a) shows that how instructions are stored at controller's and they are delivered to processor. Instructions are directly connected to processor so corresponding instruction for each state is fed into processor. In Figure 3 b) presents the cache structure. There is a cache block with tag between instructions and processor. The cache stores some of frequently used instructions which are initially stored at instruction memory. The way how instructions are delivered to processor is that firstly, tag is checked whether necessary instruction exists in cache or not. If there is an appropriate instruction in cache, it goes directly to processor. Otherwise, if necessary instruction is not in cache, instruction from memory is delivered to processor. At this moment, the instruction is also copied to cache at the location where currently least frequently used instruction's position. The cache structure takes more than one cycle in case of cache miss so that this cache structure does not fit to our target architecture. Control words need to be generated at every cycle in PLA because the datapath operates within single cycle. Also, another disadvantage of cache is that if cache miss happens frequently, the energy overhead occurs by delivering instructions to datapath from memory and copying it in cache.

In Figure 3 c) presents method called Instruction Register File (IRF) structure. Scratch-pad [6] [7] and filter cache [9] are possible implementation of this method. There is an IRF block between instructions and processor. It has smaller size of memory and stores some of instructions from memory. In IRF architecture, instructions are transfered always from IRF to processor because there are always necessary instructions at IRF. All instructions are copied from memory before delivering to processor in advance. All instructions at IRF are moved from memory dynamically but the locations are statically assigned. The location of copying destination for each instruction is scheduled statically so that there is no cache miss. Therefore, delivery of a instruction to the processor can be performed at one cycle. We will apply this IRF method to our target PLA architecture in order to decrease power overhead.

The details of IRF architecture is shown at Figure 4. It is modified based on original programmable loop accelerator which is shown at Figure 2. IRF and DMA block differentiate the IRF struc-

ture compared to basic programmable loop accelerator architecture. Also, the organization of instructions is slightly different from basic PLA. In IRF architecture, we need DMA extra control signals to operate DMA block so that DMA control can be performed based on instructions. Figure 4 shows that now instruction has control words and DMA control. We can observe how structure of instruction is different between original one and IRF in Figure 5

IRF block behaves scratch-pad as we discussed in Figure 2. DMA block manage copying procedure to copy instructions from memory to IRF. DMA control signals specify which instruction is need to be copied from memory to IRF, where it has to be located, and how many instructions. Figure 5 b) shows how DMA control signal is organized. The first portion presents the source which is address of instruction memory, the second region is the destination which is address of IRF, and last section is the length of transfer. The size is equivalent to number of instructions copied from memory into IRF. Note that copying one instruction takes one cycle so that if several instructions are copied, it takes as many cycles as size. The copying procedure is executed at the same time when control word is transfered to datapath. More details of the microcoded programmable controller is shown in Figure 6.

3. MOTIVATIONAL EXAMPLE

In this section, we would like to demonstrate how an example operates with various size of IRF. Figure 7 a) shows the example CRC32 written in C code. Iteration of *for* loop is 1024 and 3 arrays are used in C code. In Figure 7 b) C code is translated into control data flow graph (CDFG) and scheduled with resource constraints that only 3 multiplication units, 3 ALUs, 2 shift units, and one load and store unit are available. The most left side column represents states in scheduled graph in Figure 7 b). Based on scheduling result, number of states in the controller is decided and there are 11 states include the end-state which is S10 in Figure 7. Instructions from state 0 to state 6 are executed in *for* loop. There are 11 instructions in total and 7 instructions are necessary to complete loop for CRC32 application. State transition diagram of scheduling result is shown in figure 7 c). Figure 7 d) shows instructions stored in memory. Each instruction is corresponding to each state and single instruction has its control words and possible next-state candidates. Left side of column are the representations of control signals and instruction for S10 does not have any operation which corresponds to no-operation. Right column are possible next-states. Only state 0 has two next-state candidates because it goes either S1 or S7 based on the result of comparison result from datapath. In addition, as we mentioned before, the size of memory varies depending on the largest application. Even though there are only 11 instructions for CRC32 example but the size of memory for PLA is larger. The figure 5 d) only shows valid memory content but note that we assumes the size of memory is set to 512. The 11 instructions are stored as memory has size as 512 in the example.

In order to minimize energy overhead with using small IRF, we would like to use either size of IRF is 4 or size of IRF is 8. In the case of size of IRF is 4, number of IRF is only 4 so we cannot store more than 4 instructions on IRF. Therefore, we have to copy some of instructions from memory into IRF by triggering DMA control. If copying instruction happens frequently, we need to access memory more and it causes more energy consumption. Therefore, we need to pin instructions in IRF as many as possible to minimize copying procedure. Figure 8 presents instructions stored in instruction memory for IRF when the size is 4. The mapping of instructions at memory into IRF is also shown in Figure 8 a). There are 7 states in for loop and 3 instructions from loop are fastened to IRF and other 4 instructions are overwritten at address of 3 in IRF con-

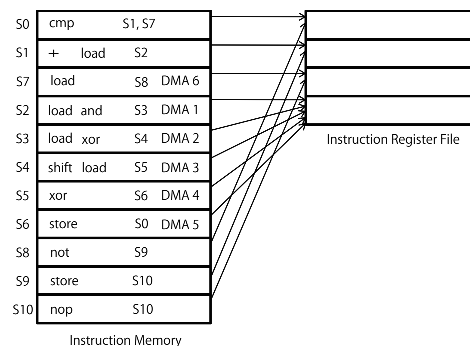


Figure 8: Mapping When Size of IRF is 4

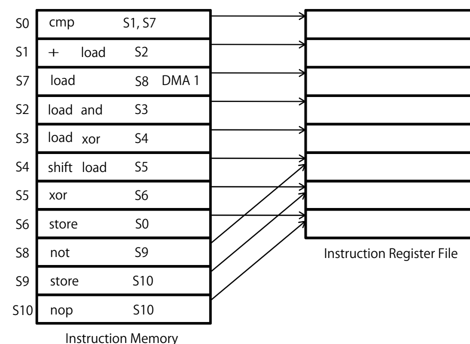


Figure 9: Mapping When Size of IRF is 4

tinuously in every cycle. The order of instructions in memory is reorganized compared to original PLA to number of copying procedure. We assume that 4 sequential instructions from instruction memory are copied to IRF initially. After S0 is executed, either S1 or S7 can be executed. Therefore, storing S7 in IRF with S1 at same time we can reduce total number of copying, unless S7 is in IRF we need to copy S7 into IRF.

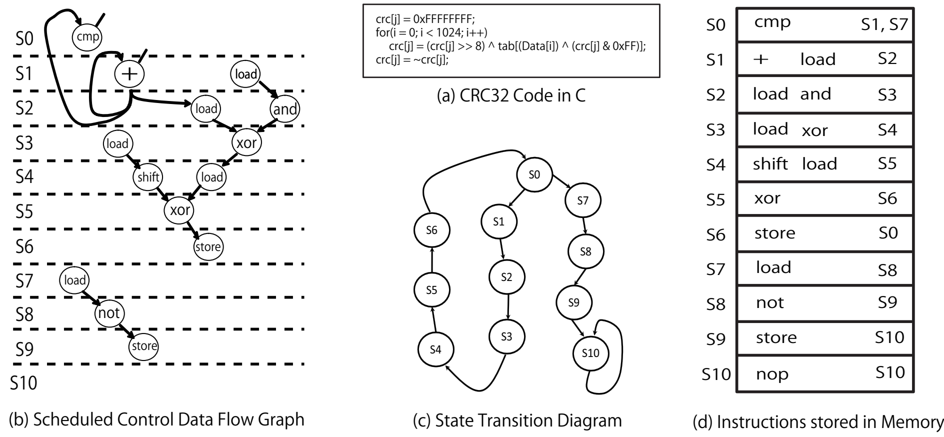
Next states are corresponding to the address of IRF because whole instructions are stored in IRF before execution. The mapping address in IRF is shown in Table 1. DMA control in detail is also described in Table 2. Note that DMA0 is initially called before executing S0. Figure 9 shows that how mapping is done and how memory is organized when IRF size when 8. In this case, we can store all instruction from *for* loop since number of states in loop is 7 so that we can save much more energy overhead by reducing number of copy procedures during the loop. In contrast to the scenario where size of IRF was 4 and copying operations were performed many times, copying procedure is done only 11 times in the case of size of IRF is 8. The mapping of states into addresses in IRF is shown in Table 1, and control signals for each DMA operation is shown in Table 2.

Due to space limits, here only explanations based on examples are shown. Details of the algorithms discussed above can be found in [10].

4. EXPERIMENTAL RESULTS

4.1 Experimental Setups

In this section, we would like to evaluate how energy various in Programmable Loop Accelerator with different sizes of IRF, com-



(a) CRC32 Code in C

(b) Scheduled Control Data Flow Graph

(c) State Transition Diagram

(d) Instructions stored in Memory

Figure 7: Example of CRC32

Table 1: Mapping Address in IRF

State	Mapping Address	
	Size of IRF = 4	Size of IRF = 8
S0	0	0
S1	1	1
S2	3	3
S3	3	4
S4	3	5
S5	3	6
S6	3	7
S7	2	2
S8	0	5
S9	1	6
S10	2	7

Table 2: DMA Control

DMA	Size of IRF = 4	Size of IRF = 8
DMA0	0 0 4	0 0 8
DMA1	4 3 1	4 3 3
DMA2	5 3 1	
DMA3	6 3 1	
DMA4	7 3 1	
DMA5	3 3 1	
DMA6	8 0 3	

pared to original PLA. The row function from IDCT example and other 3 examples are used as benchmarks. This is done by first building a Verilog file starting from C code as we illustrated in the previous section. First, we generate a Verilog code of each benchmark application which contains both controller and datapath. Next, multiple Verilog codes are generated from the Verilog file for different implementations. They are corresponding to original PLA and PLA with IRF architecture proposed in Section 4. With various sizes of IRF, multiple number of Verilog files of benchmarks are used at experiments. Also, in order to observe variation with Instruction Memory size, several different size of memories are used. All Verilog files are synthesized in same manner which contains the same constraints. Note that datapath is same among all Verilog files. Only structure of controller are different whether IRF exists or not and what size of memory and IRF are used. Instructions stored at Instruction Memory on controller are generated in order to simulate each verilog file. To evaluate power consumptions of Verilog files, we used our power model described in previous section. Table 3 summarizes the scheduled benchmark's characteristics used for both the original PLA and PLA with IRF.

Table 3: Resource Constraints, Scheduling Length, Size of Instructions, and Number of Registers Used for Benchmark

Benchmarks	IDCTrow	CRC32	FIR	Bubble Sort
Total Cycles	217	7172	188	30199
Instructions	28	11	8	15
Copies with IRF 4	204	5127	159	1287
Copies with IRF 8	176	11	8	799
Copies with IRF 16	120	11	N/A	15
Copies with IRF 32	28	N/A	N/A	N/A

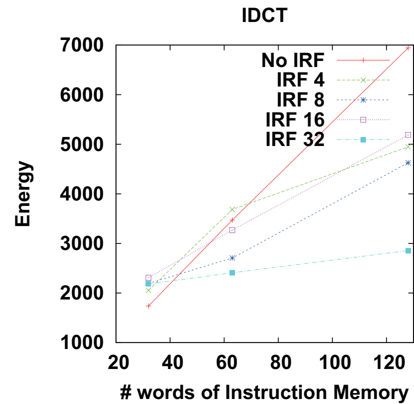


Figure 10: Power consumption with different memory size

4.2 Power Reduction Results

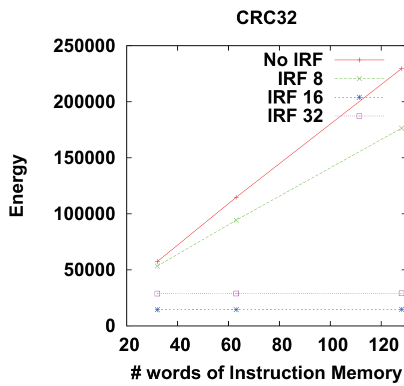


Figure 11: Power consumption with different memory size

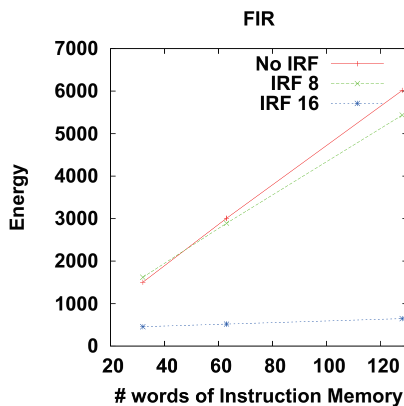


Figure 12: Power consumption with different memory size

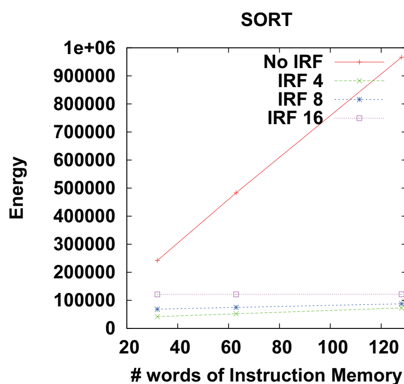


Figure 13: Power consumption with different memory size

In Figure 10, 11, 12, 13 the result is shown in graph with various memory height from 32 to 128 for both original Programmable Loop Accelerator and PLA with IRF size as 8, 16, and 32 respectively. Y axis in graph presents the energy consumption in number of accesses for IM size is 4 calculated from our power model. X axis resents the different memory size from 32 to 128. Four different lines from the top are represented original PLA and PLA with IRF 8, IRF 16, and IRF 32 respectively. From 10, firstly we can observe that bigger IRF consume smaller power in overall as each slope of line resents. If IRF has enough size that can include every instruction in *loop*, energy consumption will be widely decreased. Secondly, in the case of bigger memory size, energy reduction rate is very high compared to no-IRF architecture. IRF 32 consumes only 40% of energy compared to original PLA. Lastly, when using bigger size of IRF, energy consumption between small size of memory and larger size of memory is very similar. Using big-enough IRF size can prevent high increments of power consumption depending memory size. That is to say IRF can save the significant amount of energy consumption in Microcoded Programmable Controller which uses larger size of memory. Another observation of the case of IRF 32 with memory size of 32, it consumes a little more power compared to original PLA because IRF size is same as Instruction Memory and we also need to copy instructions from Instruction Memory to IRF. Also, another important observation of this experimental result is that IRF size is not proportional to energy reduction because power is dependent on number of loop iterations and number of copy.

5. CONCLUSION

When size of memory is larger, more power can be saved with IRF architecture. We can reduce number of times accessing the memory of larger size and substitute to accessing relatively small IRF. The difference of energy consumption between no IRF and IRF 32 is bigger in larger size of instruction memory. We can save more power if reasonable size of IRF is provided, especially when size of instruction memory is large. Therefore, we can conclude that utilizing memory hierarchy is effective for reducing power in PLA.

6. REFERENCES

[1] K. Fan, M. Kudlur, G. Dasika, and S. Mahlke. Bridging the Computation Gap Between Programmable Processors and Hardwired Accelerators In *Proc. of 15th Intl. Symposium on High-Performance Computer Architecture*, pages 313-322, Feburary 2009.

[2] B. Gorjiara, M. Reshadi, and D. Gajski. Merged Dictionary Code Compression for FPGA Implementation of Custom Microcoded PEs, In *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, pages 1-21, June 2008.

[3] B. Gorjiara, and D. Gajski. FPGA-friendly Code Compression for Horizontal Microcoded Custom IPs, In *International Symposium on Field-Programmable Gate Arrays (FPGA)*, February 2007.

[4] K. Fan, H.l Park, M. Kudlur, and S Mahlke. Modulo scheduling for highly customized datapaths to increase hardware reusability. In *Proc. of the 2008 International Symposium on Code Generation and Optimization*, pages 124-133, April 2008.

[5] D. A. Patterson and J. L. Hennessy. Large and Fast: Exploiting Memory Hierarchy, in *Computer Organization & Design The Hardware/Software Interface:* Morgan Kaufmann, 1994.

[6] A. Janapsatya, S. Parameswaran, and A. Ignjatovic. Hardware/software managed scratchpad memory for embedded system. In *Proc.of the 2004 IEEE/ACM International conference on Computer-aided design*, pages 370-377, November 2004.

[7] R. Banakar, S. Steinke, B. Lee, M. Balakrishnan, and P. Marwedel. Scratchpad memory: design alternative for cache on-chip memory in embedded systems. In *Proc. of the tenth international symposium on Hardware/software codesign*, May 2002.

[8] P. Ranjan Panda, N. D. Dutt, and A. Nicolau. Efficient Utilization of Scratch-Pad Memory in Embedded Processor Applications. In *Proc. of the European conference on Design and Test*, page 7, March 1997.

[9] J. Kin, M. Gupta, and W. H. Mangione-Smith. The Filter Cache: An Energy Efficient Memory Structure. In *IEEE Micro*, December 1997.

[10] Jaeho Lee. Power Reduction Technique of Microcoded Programmable Controller. Master Thesis, Department of Electrical Engineering, University of Tokyo, 2010.

Dynamic Scan Clock Control in BIST Circuits

Priyadharshini Shanmugasundaram and Vishwani D. Agrawal
Auburn University
Auburn, Alabama 36849
pzs0012@auburn.edu, vagrawal@eng.auburn.edu

***Abstract*—We dynamically monitor per cycle scan activity to speed up the scan clock for low activity cycles without exceeding the specified peak power budget. The activity monitor is implemented as on-chip hardware. Two models, one for test sets with peak activity factor of 1 and the other for test sets with peak activity factor lower than 1 have been proposed. In test sets with peak activity factors of 1, the test time reduction accomplished depends upon an average activity factor of α_{in}. For low α_{in}, about 50% test time reduction is analytically shown. With moderate activity, α_{in} = 0.5, simulated test data gives about 25% test time reduction for ITC02 benchmarks. BIST with dynamic clock showed about 19% test time reduction for the largest ISCAS89 circuits in which the hardware activity monitor and scan clock control required about 2-3% hardware overhead. In test sets with peak activity factors lower than 1, the test time reduction depends on an input activity factor of α_{in} and an output activity factor of α_{out}. For low α_{in} and high α_{out}, a test time reduction of about 50% is analytically shown.**

***Index Terms*—Scan test, test time reduction, test power, on-chip activity monitor, adaptive test clock, activity factor, BIST**

I. Introduction

Scan testing [1] spends a large fraction of the test time for loading (scan-in) and unloading (scan-out) test data in flip-flops that are chained as shift registers. During this process, random combinational logic activity can produce large unintentional power consumption resulting in power supply noise and heating. If this consumption is higher than that of the normal functional operation for which the circuit is designed the test can cause yield loss [2]. Therefore, scan testing uses a slower speed than the normal operation. The scan clock frequency is determined based on the maximum power consumption the circuit under test can withstand. The power P dissipated at a node is given by [2]:

$$P = \frac{1}{2} CV^2 \alpha f \tag{1}$$

where C is the capacitance of the node, V is supply voltage, f is clock frequency and α is a node activity factor.

$$\alpha = Number\ of\ transitions\ per\ clock\ cycle \tag{2}$$

The activity factor α for a clock signal is 2 because there are two (rising and falling) transitions per cycle. For a combinational node, α ranges between 0 (no transition) and 1 (a toggle every clock cycle). In the worst case, the frequency of the scan clock can be based on the maximum activity, i.e., $\alpha = 1$, so that the test power can never exceed the power limit. Therefore,

$$P_{budget} = \frac{1}{2} CV^2 f_{test} \tag{3}$$

where f_{test} is the scan clock frequency and P_{budget} is the maximum power dissipation the circuit can withstand without malfunctioning. Thus,

$$f_{test} = \frac{2P_{budget}}{CV^2} \tag{4}$$

In general, the worst case assumption $(\alpha = 1.0)$ can be modified for any value. Although all vectors are scanned in and scanned out at this frequency, many may not cause the maximum activity. It is possible to scan in these vectors at higher clock frequencies without exceeding the power budget. When the number of transitions in the circuit reduces to a $\frac{1}{i}$ of the maximum,

$$P = \frac{1}{2} CV^2 \frac{1}{i} f_{test} \tag{5}$$

From (3) and (5),

$$\frac{P}{P_{budget}} = \frac{1}{i} \tag{6}$$

The capacitance and the voltage are constant for a node and so the power is proportional to the product of activity and frequency. Since the circuit can withstand a power P_{budget}, the frequency can be multiplied by i, and the power dissipated in every cycle can still be kept within the allowed limit. Girard [2] defines peak power as the highest energy consumed during one clock period divided by the clock period and the average power as the total energy consumed during test divided by the test time. Since the power must never exceed P_{budget} in any clock cycle, both peak power and average power will be below P_{budget} in spite of the increased shift frequency. Also, instantaneous peak power [2] is consumed right after the application of the clock edge. This power depends on the vectors scanned in and is unaffected by changes in the scan clock frequency. Hence, it can be reduced only by changing the test vectors. In this work we assume that the vectors conform to the instantaneous peak power requirement.

During scan tests, gates are either driven by outputs of the scan flip-flops or by primary inputs. Primary inputs do not change during scan in and scan out. Thus, scan chain activity is a direct measure of the test power and by *monitoring and controlling* this activity, we can speed up the test as well as limit the test power. That is the idea presented in this paper.

Section II discusses previous work on test time optimization. Section III discusses implementations of the proposed technique. Section IV gives a mathematical analysis of the scheme. Section V explains the experimental results obtained. Section VI discusses the conclusion of this work.

II. Previous Work

Many test time reduction methods for scan circuits use compression. In a simple compression technique, the number of scan chains is increased reducing the number of flip-flops per chain. This reduces the time for shifting the input vector bits through scan flip-flops resulting in an overall reduction in test time. However, compression techniques require alterations in the design and may also suffer from linear dependencies.

One compression technique keeps the functionality of the ATE intact by moving the decompression task to the circuit under test [3]. Another technique [4] uses a dynamically reconfigurable scan tree that applies a part of the test sequence in scan tree mode and the other part in single scan mode. Reference [5] describes a decompression hardware scheme for test pattern compression. References [5] and [6] use compression algorithms with concurrent application of compaction

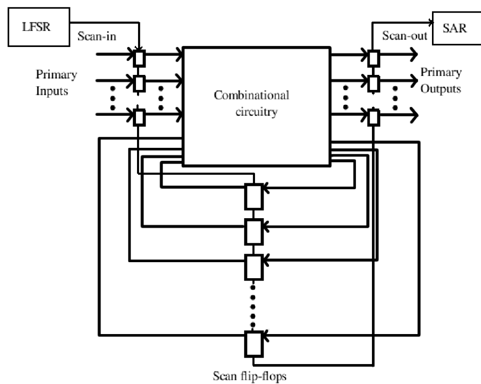


Fig. 1. Test-per-scan built-in self-test (BIST).

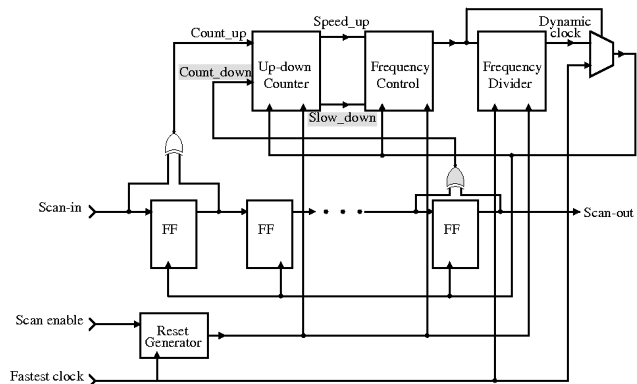


Fig. 2. Schematic of proposed dynamic frequency control.

and compression. Reference [7] implements a compression technique with embedded deterministic test logic on chip to provide vectors for the internal scan chains. Reference [8] employs alternating run-length codes [9] for test data compression.

Reference [10] employs a two phase testing strategy where the first phase is a scan-less phase for easy-to-detect faults and the second phase is a scan phase for hard to detect faults. Scan is performed only until all effective test bits are shifted to the right position and until all fault-affected response bits are shifted out. Reference [11] uses genetic algorithms to obtain compact test sets, which limit the scan operations. References [12] and [13] reduce test application time by generating a test for a sequential circuit using combinational test generation and sequential test generation adaptively. Reference [14] proposes a strategy to identify flip-flops to be removed from scan chains to increase the observability of the circuit so that faults activated during scan cycles can be observed at a primary output. The technique proposed in this paper can be applied to any scan circuitry, and can be used in addition to many of the methods mentioned above.

III. Implementation

A. BIST circuit with a single scan chain

We add flip-flops at primary inputs and outputs as shown in Figure 1 and connect all flip-flops into a single scan chain. A linear feedback shift register (LFSR), a signature analysis register (SAR) and a BIST controller are added to the circuit to implement the test per scan BIST architecture [14]. BIST vectors are scanned in and combinational outputs are captured through scan flip-flops. Application of a vector includes scanning in LFSR bits into flip-flops, normal mode capture and scan out (overlapped with next scan in) into SAR. The proposed dynamic frequency control is shown in Figure 2. The shaded parts of the circuit are not used for this implementation. As test vectors are scanned in, the activity (or inactivity) in the scan chain is monitored at the first flip-flop of the chain. The entering transitions ripple through other flip-flops in subsequent cycles. This activity does not change if there are inversions in the chain. When a transition passes through an inverting flip-flop, a rising transition becomes a falling transition and vice-versa, leaving the number of transitions unchanged.

An XNOR gate between the input and output of the first flip-flop monitors the activity. The output of the XNOR gate is 0 when a transition enters the scan chain and is 1 when a non-transition enters. The XNOR output is fed to a counter, which counts up for each 1, i.e., a non-transition. The counter is set to 0 at the start of every scan in sequence. According to (6), the scan frequency can be raised as the number of non-transitions entering the scan chain increases. This is accomplished through frequency control and frequency divider blocks in Figure 2. We assume that the response captured from the combinational circuit for the previous vector has a transition density of 1, i.e., the scan chain is filled with alternating 1s and 0s before scan-in begins. This pessimistic worst-case assumption guarantees that the power budget shall not be exceeded. Correspondingly, the scan in of each vector begins with the slowest frequency, f_{test}, permitted by the power budget for $\alpha = 1$. The f_{test} clock is the lowest frequency generated by the frequency divider that divides the frequency of an externally supplied fast tester clock. The frequency control circuit monitors the state of the counter. As the count goes up it lowers the frequency division ratio of the clock divider in several steps.

The counter states at which the clock is sped up can be found by simulation, which establishes correlation between the circuit activity and scan chain activity. If each transition in the scan chain causes a large number of transitions in the circuit, power consumption reaches large values for low scan chain transition numbers. Thus, a large number of scan chain non-transitions should be counted before the scan clock frequency is stepped up. Similarly, if a transition in the scan chain has a small effect on the circuit activity, then only a few non-transitions in the scan chain are sufficient to increase the scan clock frequency.

The reset generator in Figure 2 applies a reset signal to the counter, frequency control block and frequency divider at the positive edge of the scan enable signal, i.e., at the start of scan-in for every combinational vector. Since the frequency divider cannot generate a f/1 (divide by 1) clock, a multiplexer selects either the frequency divider output or the fastest clock.

Let us consider a circuit with 1000 flip-flops. If the slowest scan clock period based on the power budget is 80ns and we raise the frequency in 8 steps, then a modulo 125 (1000/8) counter will be implemented. *Assuming the worst-case activity by the captured states*, every scan-in is started with the 80ns clock and counter set to 0. The count goes up by 1 at every clock in which a non-transition enters the scan chain. When the count reaches 125, the counter is reset and the frequency divider generates a 70ns clock to scan-in the subsequent bits. The counter may again count up to 125 and the clock period would be reduced to 60ns. This process repeats until all 1000 bits are scanned in. Thus, if the input were a series of 1000 1s, the first 125 bits are scanned in at a clock of period 80ns, the second 125 bits at 70ns, until the last 125 bits are scanned in using a clock period 10ns. If the scan-in bits were a series of alternating 0s and 1s, the counter would never count up since there are no non-transitions entering the scan chain and

hence the entire scan-in will use the 80ns clock. Notice that due to the *worst-case assumption* we start each scan-in with slowest clock and so the activity monitor only raises the clock rate without ever having to lower it during the same scan-in.

Clearly, a bit stream with fewer transitions will be scanned in faster than one with many transitions. Don't cares in deterministic ATPG patterns can be filled in such that the number of transitions is minimum [15]. Also, techniques to generate BIST patterns with low transition densities [16] may be useful. This technique would perform well for such patterns.

B. BIST circuit with multiple scan chains

When the circuit has multiple scan chains, the activity of all chains must be monitored. XNOR gates are added across the input and output of the first flip-flop in every scan chain. Outputs of XNOR gates are supplied to a parallel counter [17] that counts up by the number of 1s at its input. The rest of the circuitry remains unaltered and still resembles the unshaded part of Figure 2. When the count reaches a certain threshold value, the frequency is stepped up and the counter is reset. Except for the use of the parallel counter the control scheme is similar to that in the unshaded portion of Figure 2.

C. BIST circuit with single scan chain and $\alpha_{peak} < 1$

The proposed implementation works well for circuits with test vectors having peak activity factors of 1. It does not require simulation of test vectors in order to estimate the peak activity factor and has low area overhead. However, it is not suitable when the scan clock frequency is computed based on a peak activity factor (α_{peak}) lower than 1. In such cases, it becomes necessary to modify the proposed model.

The slowest scan clock frequency is chosen using Eq. 1 using values of α_{peak} and peak power limit. The number of transitions in the scan chain is continuously monitored at the input and output of the scan chain.

Figure 2 shows the implementation of the technique for BIST circuits with single scan chain and peak activity factors lesser than 1. The activity monitor comprises of an xnor gate connected between the input and output of the first flip-flop, and an xnor gate connected between the input and output of the last flip-flop. The former monitors the number of non-transitions entering the scan chain and the latter monitors the number of non-transitions leaving the scan chain. An up-down counter keeps track of the number of non-transitions in the scan chain. Thus, the former xnor drives the count_up signal and the latter drives the count_down signal of the up-down counter. The number of non-transitions in the scan chain during any cycle is the difference between that entering the scan chain and that leaving the scan chain.

Since power is proportional to the activity in the scan chain, test power is lower when the number of transitions in the scan chain is lower or, in other words, when the number of non-transitions in the scan chain is higher. As discussed earlier, from (6), the scan frequency can be increased when the number of non-transitions in the scan chain increases.

The up-down counter is reset to 0 at the start of scan-in. When a non-transition enters the scan chain, the counter counts up and when a non-transition leaves the scan chain, the counter counts down. When the counter counts up to a certain threshold value, it signals the frequency control block to increase the frequency of scan clock and the counter is reset to 0. Similarly, when the counter counts down to 0, the frequency control block is signaled to lower the frequency of scan clock and is reset to the threshold value. Thus, whenever the number of non-transitions in the scan chain increases, the frequency is increased and when the number reduces, the frequency is decreased. The rest of the circuitry functions the same as described earlier.

At the start of scan-in of a vector, the frequency control block is reset such that the frequency of scan clock is the slowest possible. This is based on the assumption that the activity factor of the vector captured in the scan chain before the start of scan-in equals α_{peak}. The scan clock frequency is never increased beyond the highest or decreased below the lowest possible frequency regardless of the signal from the counter.

It can be observed that this implementation can be easily modified for circuits with activity factors equal to 1, by removing the flip-flop at the end of the scan chain and tying the count_down signal of the up-down counter to 0.

D. BIST circuit with multiple scan chains and $\alpha_{peak} < 1$

When the circuit has multiple scan chains, the activity of all chains must be monitored. XNOR gates are added across the input and output of the first flip-flop and across the input and output of the last flip-flop in every scan chain. The outputs of the XNOR gates at the inputs of the scan chains are fed to the count_up inputs of a parallel counter [17] which counts up by the number of 1s at its count_up inputs. Similarly, the outputs of the XNOR gates at the end of the scan chains are fed to the count_down inputs of the parallel counter [17] which counts down by the number of 1s at its count_down inputs. The rest of the circuitry remains unaltered and still resembles Figure 2. When the count reaches a certain threshold value, the frequency is stepped up and the counter is reset. Except for the use of the parallel counter the control scheme is similar to that in Figure 2.

IV. Analysis

Let N be the number of flip-flops, k be the peak activity factor of the test vectors (k = α_{peak}), α_{in} be the activity factor of the scan-in vector, α_{out} be the activity factor of the vector captured in the scan chain prior to scan-in, A be the number of non-transitions that enter the scan chain per cycle, v be the number of frequencies and T be the time period corresponding to the fastest clock.

The period of the fastest scan clock is v times shorter than the slowest clock. Therefore, the period of the slowest clock is given by vT. If the vectors were scanned in at the slowest clock, the total scan-in time per vector would be NvT.

If the scan-in vector has a uniform activity factor that is higher than that of the captured vector, the number of non-transitions entering the scan chain will be lower than that leaving it and hence there will be no change in scan clock frequency. However, if the scan-in vector has a uniform activity factor that is lower than that of the captured vector, the number of non-transitions entering the scan chain exceeds that leaving the scan chain. Therefore, the scan clock frequency is continuously increased. The scan-in of test vectors is started at the slowest possible clock period which equals vT and then continuously increased.

The number of transitions in the scan chain can range from 0 to kN. Therefore the number of non-transitions in the scan chain can range from $N - kN$ to $N - 0$ i.e., from $N(1-k)$ to N. In order to simplify the values, $N(1-k)$ is subtracted from both limits. Thus, the number of non-transitions can be monitored between $N(1-k) - N(1-k)$ and $N - N(1-k)$ i.e. between 0 and kN.

Since the maximum number of non-transitions encountered by the activity monitor is kN, a scan clock frequency is specified for every $\frac{kN}{v}$ non-transitions, in order to enable frequency control for all ranges of non-transitions. The scan

frequency is therefore increased every time the counter counts up to $\frac{kN}{v}$. Since A is the rate at which non-transitions enter the scan chain, A non-transitions enter the scan chain in 1 cycle. Because 1 non-transition enters the scan chain in $\frac{1}{A}$ cycles, x non-transitions will enter the chain in $\frac{x}{A}$ cycles.

The first bit in the scan-in vector is shifted at the lowest possible frequency (first frequency employed) which corresponds to a time period of vT. The frequency is not increased until $\frac{kN}{v}$ non-transitions enter the scan chain as discussed earlier. Since a non-transition enters the scan chain every $\frac{1}{A}$ cycles, $\frac{kN}{v}$ non-transitions are encountered in $\frac{kN}{Av}$ cycles. Thus, the frequency is not increased until about $\frac{kN}{Av}$ cycles. The counter is then reset and the frequency is increased to the next step which corresponds to a time period of $(v-1)T$. The frequency is not increased any further until the counter counts up to $\frac{kN}{v}$, i.e., until the number of non-transitions in the scan chain increases by $\frac{2kN}{v}$. This occurs after about $\frac{2kN}{Av}$ cycles (since a non-transition enters every $\frac{1}{A}$ cycles). Thus, the scan clock frequency (second frequency employed) whose clock period is $(v-1)T$ is used between the cycles $\frac{kN}{Av}$ and $\frac{2kN}{Av}$. The clock period can reach a maximum of T (v^{th} frequency employed). This frequency is used when the number of non-transitions in the scan chain increases by a value in the range between $\frac{(v-1)kN}{v}$ and kN or in other words, this frequency is used between clock cycles $\frac{(v-1)kN}{Av}$ and $\frac{kN}{A}$. Thus, by observation, the ith frequency corresponds to a clock period of $(v-i+1)T$ when the scan chain has between $\frac{(i-1)kN}{v}$ and $\frac{ikN}{v}$ increase in number of non-transitions. The ith frequency is employed between clock cycles $\frac{(i-1)kN}{Av}$ and $\frac{ikN}{Av}$.

The scan clock initially has a clock period of vT in cycle 1. The scan clock period is decreased in steps until the Nth cycle. Thus, the clock cycle corresponding to the last scan clock frequency is N. If the maximum number of speeds the scan clock will reach, for any vector is given by i, then

$$\frac{ikN}{Av} = N \qquad (7)$$

$$i = \frac{Av}{k} \qquad (8)$$

The total scan-in time per combinational vector is the sum of all clock periods used. The test time at each frequency is given by the product of the number of cycles run at that frequency and the clock period. Total time per vector is given by

$$\sum_{i=1}^{\frac{Av}{k}} \{\{\lceil \frac{ikN}{Av} \rceil - \lceil \frac{(i-1)kN}{Av} \rceil\}(v-i+1)T\} \qquad (9)$$

where v is usually chosen as a power of 2 because we can design a divide by 2^n frequency divider with n flip-flops. Time per vector if a single speed is used is NvT, and hence, the reduction in test time is given by

$$\frac{NTv - \sum_{i=1}^{\frac{Av}{k}} \{\{\lceil \frac{ikN}{Av} \rceil - \lceil \frac{(i-1)kN}{Av} \rceil\}(v-i+1)T\}}{NTv} \qquad (10)$$

If N and v were chosen as powers of 2, Eq. 9 reduces to

$$\text{Total time per vector} = \sum_{i=1}^{\frac{Av}{k}} \{(\frac{kN}{Av})(v-i+1)T\}$$

TABLE I
SCAN-IN TIME REDUCTION VS. NUMBER OF SCAN CLOCK SPEEDS FOR ACTIVITY FACTOR $\alpha_{in} = 0.5$.

Number of scan clock speeds	Test time reduction (%)		
	Simulation	Eq. (10)	Eq. (14)
1	0.00	0.00	0.00
2	0.34	0.00	0.00
4	12.64	12.50	12.50
8	18.78	18.75	18.75
16	22.03	21.90	21.88
32	23.56	23.48	23.44
64	25.17	24.26	24.22
128	27.41	24.66	24.61

TABLE II
SCAN-IN TIME REDUCTION VS. ACTIVITY FACTOR α_{in} FOR 8 SCAN-IN CLOCK SPEEDS.

Activity factor, α_{in}	Test time reduction (%)		
	Simulation	Eq. (10)	Eq. (14)
0	43.75	43.75	43.75
0.1	38.63	38.85	38.75
0.2	34.00	33.95	33.75
0.3	28.97	28.99	28.75
0.4	23.51	23.94	23.75
0.5	18.78	18.75	18.75
0.6	14.92	14.04	13.75
0.7	9.60	9.36	8.75
0.8	4.79	4.68	3.75
0.9	0.00	0.00	0.00
1	0.00	0.00	0.00

$$= (\frac{kN}{Av})(v.\frac{Av}{k} - \frac{\frac{Av}{k}(\frac{Av}{k}+1)}{2} + \frac{Av}{k})T \qquad (11)$$

Time per vector if a single speed is used is NvT, and

$$\text{Reduction in test time} = \frac{\{NTv - NT(v - \frac{Av}{2k} + \frac{1}{2})\}}{NTv}$$

$$= \frac{A}{2k} - \frac{1}{2v} \qquad (12)$$

The number of non-transitions in the scan chain in any cycle equals the difference between the number of non-transitions entering and leaving the scan chain. Non-transitions enter the scan chain at a rate of $(1-\alpha_{in})$ and leave at the rate of $(1-\alpha_{out})$. The non-transition density, A is therefore given by $A = \alpha_{out} - \alpha_{in}$. Thus, the reduction in test time is given by

$$\frac{(\alpha_{out} - \alpha_{in})}{2k} - \frac{1}{2v} \qquad (13)$$

where $k = 1$ for the model where the peak activity factor is assumed to be 1. In this model, the scan chain is assumed to be filled with transitions prior to scan-in and hence, the scan-in vector is assumed to be the sole contributor of non-transitions in the scan chain is . Thus, non-transitions enter the scan chain at a rate of $(1-\alpha_{in})$ and hence, $A = 1 - \alpha_{in}$. The reduction in test time for this model is given by

$$\frac{(1-\alpha_{in})}{2} - \frac{1}{2v} \qquad (14)$$

A C program was written to generate random vectors for a circuit with 1000 flip-flops. The test time reduction for these vectors was estimated, and compared with the values obtained from the formula. Table I shows the test time reduction versus number of frequencies for an activity factor of 0.5. Table II shows the variation of test time reduction with activity factor when the number of frequencies is 8. Both tables compare the test times estimated for random vectors (column 2), with those obtained from the accurate formula (9) (column 3) and from the approximate formula (14) (column 4).

Tables I and II show that for a chosen number of frequencies, vectors with lower activity achieve higher reduction in test time. The test time reduction increases when the number

TABLE III
REDUCTION IN TEST TIME FOR ISCAS89 CIRCUITS - TEST PER SCAN BIST WITH SINGLE SCAN CHAIN.

Circuit	Number of scan flip-flops	Number of frequencies	Reduction in time (%)	Increase in area (%)
s27	8	2	7.49	14.72
s298	23	4	14.57	16.25
s420	35	4	13.81	13.02
s838	67	4	13.51	11.73
s1423	96	4	13.60	8.77
s5378	263	4	13.03	6.65
s9234	286	4	14.01	5.82
s13207	852	8	19.00	3.98
s15850	761	8	18.97	3.23
s35932	2083	8	18.74	2.55
s38417	1770	8	18.83	3.14
s38584	1768	8	18.91	2.13

of frequencies increases. The test time initially reduces rapidly for 8 frequencies and after that the reduction is gradual.

V. EXPERIMENTAL RESULTS

A. Circuits with $\alpha_{peak} = 1$

In verilog netlists of the ISCAS89 benchmark circuits flip-flops were added at all primary inputs and primary outputs. All flip-flops were converted to scan types and chained together. Thus, the number of flip-flops in the circuit would be the sum of the number of primary inputs, number of primary outputs and number of D-type flip-flops. A 23-bit linear feedback shift register (LFSR), a 23-bit signature analysis register (SAR), and a test-per-scan BIST controller were implemented [18], [19]. A single bit output of the LFSR supplied the scan input and the scan output was fed into the SAR. A suitable number for random patterns to achieve sufficient fault coverage for each circuit [20] was incorporated into the BIST controller. The sequential circuit along with the BIST circuitry was treated as the core circuit for test time and area analysis. The counter, frequency control circuitry, and frequency divider circuitry for dynamic frequency control were implemented as shown in the unshaded portions of Figure 2. The number of frequencies for each circuit was chosen according to the size of the circuit or the number of scan flip-flops.

ModelSim from MentorGraphics was used to simulate the circuits with and without the dynamic frequency control circuitry. The time required for test application was recorded in each case. DesignCompiler, a synthesis tool from Synopsys, was used to analyze the area of the circuits with and without the dynamic frequency control circuitry.

Since the LFSR generates pseudo random patterns, the activity factor is about 0.5. From (8), $x = 0.5v$, and hence, the number of frequencies the circuit will run at, is half the chosen number of frequencies. This corresponds to a clock period of $(0.5v + 1)T$. However, during power analysis, the next higher frequency is taken into consideration, in order to obtain pessimistic data. Thus, power analysis is done for a clock period of $0.5vT$, i.e., for a clock having twice the lowest frequency. Therefore, the power dissipated by the circuit for an activity factor 0.5 at every node and operating at twice the lowest frequency, was estimated for every circuit. The dynamic frequency control circuitry was included in this analysis. Table III shows the results. The number of frequencies chosen for each circuit is shown in column 3. The percentage reduction in test time with respect to the test time for the core circuit is shown in column 4 and the percentage increase in area with respect to the area of the core circuit is shown in column 5. At any node, the capacitance and the voltage are constant. From (1), the power dissipated at any node is proportional to the product of activity and frequency. Thus, the activity per unit time is a direct measure of power dissipated in the circuit. Therefore, an analysis to find activity per unit time

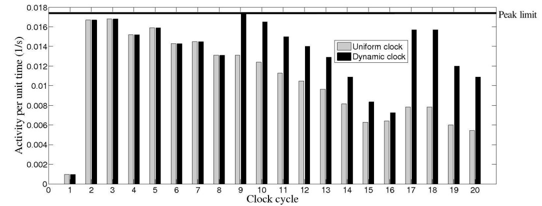


Fig. 3. Activity vs. number of clock cycle for s386 circuit.

TABLE IV
REDUCTION IN TEST TIME FOR ITC02 CIRCUITS.

Circuit	Scan flip-flops	Number of frequencies	Test time reduction (%)		
			$\alpha_{in} \approx 0$	$\alpha_{in} = 0.5$	$\alpha_{in} \approx 1$
u226	1416	8	46.68	18.75	0
d281	3813	16	46.74	21.81	0
d695	8229	32	48.28	23.36	0
f2126	15593	64	49.15	24.18	0
q12710	26158	128	49.45	24.53	0
p93791	96916	512	49.72	24.81	0
a586710	41411	256	49.73	24.77	0

was performed on the s386 benchmark circuit. The Synopsys power analysis tool, PrimeTime PX, was used. The activity per unit time in every cycle was found for the circuit for a scan vector with an activity factor of 1. The peak among these values was set as the limit for activity per unit time. The values of activity per unit time of the circuit in every cycle were found for a vector with an activity factor of 0.25 using uniform clock and dynamic clock methods. The results are shown in Figure 3. Notably, the activity per unit time in every cycle is closer to the peak limit when dynamic clock method is used. Also, the peak limit is never exceeded in both methods. A reduction of 11.25% was observed when the dynamic clock method was used.

The results for multiple scan chain implementation would be very similar to that obtained for single scan chain. The test time will not vary much since the activity of the circuit will be very similar in both single and multiple chain implementations. However, there would be a marginal increase in area due to the additional XNOR gates at the first flip-flop of every scan chain and also due to the use of a parallel counter as opposed to the simple counter used for the single scan chain.

These results for reduction in test time conform to the theoretical results given in Tables I and II. Two trends are clearly observed in Table III. As circuit size increases, the area overhead drops and test time reduction improves. These circuits are not very large from today's standard and we can expect better results as predicted by the analysis.

To estimate the test time reduction for larger circuits, an accurate mathematical analysis was applied to ITC02 circuits. Test time reduction was computed for best ($\alpha_{in} \approx 0$), moderate (α_{in} = 0.5) and worst ($\alpha_{in} \approx 1$) cases of scan chain activity factors. The test-per-scan BIST was assumed. Table IV shows the results. The number of scan flip-flops in column 2 is the sum of number of inputs, number of outputs and number of flip-flops. The number of frequencies for circuits are shown in column 3. The test time reductions achieved for best, moderate and worst case activity factors are shown in Columns 4, 5 and 6, respectively. Evidently, more test time reduction can be achieved in larger circuits. The reduction in test time varies from 0% for patterns causing very high activity to 50% for patterns with almost no activity. When external tests are used and an ATPG tool generates them, the vectors may have very few care bits. The don't care bits can be filled in using heuristics [21] to minimize scan transitions. Then, a dynamic control of scan clock will provide a large reduction in test time. This is illustrated using the ISCAS89 benchmark s38584. The Synopsys ATPG tool TetraMAX was used to generate two sets of vectors, a set of 961 vectors

TABLE V
REDUCTION IN TEST TIME IN T512505 CIRCUIT.

α_{in}	α_{out}							
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.65
0	0	7.59	15.29	22.98	30.67	38.36	46.06	49.9
0.1	0	0	7.59	15.29	22.98	30.67	38.36	42.21
0.2	0	0	0	7.59	15.29	22.98	30.67	34.52
0.3	0	0	0	0	7.59	15.29	22.98	26.83
0.4	0	0	0	0	0	7.59	15.29	19.13
0.5	0	0	0	0	0	0	7.59	11.44
0.6	0	0	0	0	0	0	0	3.75
0.65	0	0	0	0	0	0	0	0

with no don't care bits and another set of 14,196 vectors with don't care bits. The vector set without don't cares was found to have an activity factor around 0.5 and the vector set with don't care bits was found to have a low activity factor around 0.01. The don't care bits in the second set were filled using a minimum transition heuristic [21]. Reductions of 43.14% and 18.8% were achieved for the test vector sets with and without don't care bits.

In another typical scenario, a test set may initially contain few (say, 10%) high activity ($\alpha_{in} = 0.5$) vectors. These resemble fully-specified random vectors and achieve about 70-75% fault coverage. The latter 90% vectors then detect about 20-25% hard-to-detect faults and contain many don't cares, which may be filled in for reduced ($\alpha_{in} \leq 0.05$) activity. The adoptive test will be potentially beneficial in such cases.

B. Circuits with $\alpha_{peak} < 1$

In order to estimate the reduction in scan-in time achieved with the model proposed for dynamic scan clock frequency control in circuits with peak activity factors lower than 1, the t512505 ITC02 benchmark circuit was chosen. This circuit is large enough to employ 512 different scan clock frequencies because it has 76714 scan flip-flops.

The pattern sets of various large benchmark circuits were studied to analyze trends in peak activity factors. The mean value of peak activity factor (α_{peak}) in these pattern sets was found to be around 0.57 and the standard deviation (σ) was around 0.025. The value of mean + 3σ was found to be around 0.65. This indicates that the probability that the peak activity factor of the test patterns of a circuit would lie below 0.65 is 99.7%. Therefore, the peak activity factor for the t512505 circuit was set at 0.65. The pattern sets generated by TetraMAX ATPG for large benchmark circuits were analyzed and it was found that the peak activity factor in these test vectors never exceeded 0.65.

It is important to note that the value of 0.65 for peak activity factor can be used only for large circuits having flip-flop numbers in the range of a few hundreds. For smaller circuits with flip-flop numbers in the order of a few tens, the peak activity factor was found to be 1.

Mathematical analysis was used to estimate the reduction in scan-in time achieved in the t512505 circuit when $\alpha_{peak} = 0.65$ and 512 steps of frequencies were chosen. The results are listed in Table V. It can be seen from Table V that when the activity factor of the scan-out vector (α_{in}) is greater than or equal to the activity factor of the captured vector (α_{out}), there is no reduction in scan-in time. The frequency is increased only when the number of non-transitions in the scan chain increases. However, when $\alpha_{in} > \alpha_{out}$ the number of non-transitions (as counted by the counter) never increases and hence the scan-in is carried out at the starting frequency which is the frequency employed when dynamic scan clock frequency control is not implemented. Thus, the reduction in scan-in time is 0% in such cases.

Table V shows the variation of scan-in time reduction with variation in α_{in} and α_{out}. It indicates that scan-in time reduction is higher for lower values of α_{in} and for higher values of α_{out}. This can be explained from the perspective of number of non-transitions in the scan chain. If α_{in} is low, the number of non-transitions entering the scan chain is high and if α_{out} is high, the number of non-transitions leaving the scan chain is low. Thus, the net number of non-transitions in the scan chain is high giving a higher reduction in scan-in time.

VI. CONCLUSION

Reduction of test application time in power-constrained testing by adoptively adjusting the scan frequency to the circuit activity is demonstrated. On-chip hardware, whose overhead reduces as the circuit becomes large, provides the adoptive control. The technique is particularly beneficial when the peak circuit activity during test is very high but the average activity is quite low.

Acknowledgment – This research was supported in part by the National Science Foundation Grant CNS-0708962.

REFERENCES

[1] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Springer, 2000.
[2] P. Girard, "Survey of Low-Power Testing of VLSI Circuits," *IEEE Design and Test of Computers*, vol. 19, pp. 80–90, May–June 2002.
[3] I. Bayraktaroglu and A. Orailoglu, "Test Volume and Application Time Reduction through Scan Chain Concealment," in *Proc. Des. Automation Conf.*, pp. 151–155, 2001.
[4] Y. Bonhomme, T. Yoneda, H. Fujiwara, and P. Girard, "An Efficient Scan Tree Design for Test Time Reduction," in *Proc. IEEE European Test Symposium*, pp. 174–179, 2004.
[5] I. Bayraktaroglu and A. Orailoglu, "Decompression Hardware Determination for Test Volume and Time Reduction through Unified Test Pattern Compaction and Compression," in *Proc. 21st IEEE VLSI Test Symp.*, pp. 113–118, 2003.
[6] I. Bayraktaroglu and A. Orailoglu, "Concurrent Application of Compaction and Compression for Test Time and Data Volume Reduction in Scan Designs," *IEEE Trans. Computers*, vol. 52, pp. 1480–1489, Nov. 2000.
[7] J. Rajski, J. Tyszer, M. Kassab, N. Mukherjee, R. Thompson, H. Tsai, A. Hertwig, N. Tamarapalli, G. Mrugalski, G. Eide, and J. Qian, "Embedded Deterministic Test for Low Cost Manufacturing Test," in *Proc. Int. Test Conf.*, pp. 301–310, 2002.
[8] A. Chandra and K. Chakrabarty, "Reduction of SoC Test Data Volume, Scan Power and Testing Time Using Alternating Run-Length Codes," in *Proc. Int. Conf. Computer Aided Design*, pp. 673–678, 2002.
[9] A. Chandra and K. Chakrabarty, "Frequency-Directed Run-Length (FDR) Codes With Applicatin to System-on-A-Chip Test Data Compression," in *Proc. 19th IEEE VLSI Test Symposium*, pp. 42–47, 2001.
[10] W. J. Lai, C. P. Kung, and C. S. Lin, "Test Time Reduction in Scan Designed Circuits," in *Proc. European Des. Automation Conf.*, pp. 489–493, 1993.
[11] E. M. Rudnick and J. H. Patel, "A Genetic Approach to Test Application Time Reduction for Full Scan and Partial Scan Circuits," in *Proc. Int. Conf. on VLSI Design*, pp. 288–293, Jan. 1995.
[12] S. Y. Lee and K. K. Saluja, "Test Application Time Reduction for Sequential Circuits with Scan," *IEEE Trans. CAD*, pp. 1128–1140, Sept. 1995.
[13] S. Y. Lee and K. K. Saluja, "An Algorithm to Reduce Test Application Time in Full Scan Designs," in *Proc. Int. Conf. CAD*, pp. 17–20, 1992.
[14] H. C. Tsai, S. Bhawmik, and K. T. Cheng, "An Almost Fullscan BIST Solution - Higher Fault Coverage and Shorter Test Application Time," in *Proc. Int. Test Conf.*, pp. 1065–1073, Oct. 1998.
[15] R. Sankaralingam, R. R. Oruganti, and N. A. Touba, "Static Compaction Techniques to Control Scan Vector Power Dissipation," in *Proc. IEEE VLSI Test Symp.*, pp. 35–40, April-May 2000.
[16] S. Wang and S. K. Gupta, "LT-RTPG: A New Test-Per-Scan BIST TPG for Low Heat Dissipation," in *Proc. Int. Test Conf.*, pp. 85–94, 1999.
[17] E. E. Swartzlander, Jr., "A Review of Large Parallel Counter Designs," in *Proc. IEEE Computer Society Annual Symposium on VLSI*, pp. 89–98, Feb. 2004.
[18] V. D. Agrawal, C. R. Kime, and K. K. Saluja, "A Tutorial on Built-In Self-Test, Part 1: Principles," *IEEE Design and Test of Computers*, vol. 10, pp. 73–82, Mar. 1993.
[19] C. Stroud, *A Designer's Guide to Built-In Self-Test*. Springer, 2002.
[20] F. Brglez, D. Bryan, and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," in *Proc. Int. Symp. Circuits and Systems*, pp. 1929–1934, May 1989.
[21] N. Badereddine, P. Girard, S. Pravossoudovitch, C. Landrault, and A. Virazel, "Minimizing Peak Power Consumption during Scan Testing: Test Pattern Modification with X Filling Heuristics," in *Proc. Int. Conf. on Design and Test of Integrated Systems in Nanoscale Technology*, pp. 359–364, Sept. 2006.

A Pattern Partitioning Algorithm for Field Test

Senling Wang, Seiji Kajihara, Yasuo Sato
Dept. of Computer Science and Electronics
Kyushu Institute of Technology

Xiaoxin Fan, Sudhakar M Reddy
Dept. of Electrical and Computer Engineering
University of Iowa

ABSTRACT

A BIST-based test in field has been used as one of methods to guarantee high reliability of VLSIs. But it is not an easy task for field test to achieve high test quality due to the limitation of short test application time. Test partitioning and rotating test is an effective way to satisfy such a constraint. This paper proposes an algorithm to partition a test set into subsets so as to maximize average fault coverage of the subsets. The subsets of test patterns with high fault coverage can provide high test quality in rotating test. Experimental results show the effectiveness of the proposed partitioning algorithm.

Keywords

On-line test, field test, test partitioning, reliability, BIST, ATPG

1. INTRODUCTION

Reliability is one of the major concerns for deep-submicron VLSIs. There are some aging phenomena, which are HCI, NBTI and TDDB for nanoscale transistors [1-3], electro migration and stress migration for wires including via. Such aging phenomena cause the delay degradation, increase of leakage current or open/short faults. Conventionally, the reliability of the circuits has been guaranteed by burn-in test or voltage stress test. And enough timing margin has been added to the circuit at the design phase to prevent the circuit from causing the aging-induced delay faults [3]. However the excessive burn-in test or voltage stress test would deteriorate the good chips and result in their short lifetime [4]. Even the timing margin is difficult to be determined by predicting its aging speed in actual use. If a circuit is used more frequently or in higher temperature, then an aging occurs earlier. In addition, the environment where the circuit operates often relates to its aging speed. For example, NBTI-induced delay degradation is significantly accelerated in high temperature. Therefore, the timing margin, which is derived from the worst case estimation, may results in performance degradation.

Concurrent test to check the circuit during system operation is an effective method to detect aging-induced faults or soft errors, because it has an advantage of detecting errors as soon as they appear [6-9]. On the other hand, concurrent testing requires large overhead or performance degradation due to special circuit architecture or inserted redundancy in terms of hardware, time, or information. Therefore, such methods might not be accepted for general designs widely.

Recently on-line and off-line system test in field, which relies on BIST, began to be used for some systems that need high reliability such as automotive, communication, medical , etc. [10-12]. For example, power-on test is executed in field just before the start of system operation, it can detect a fault caused by aging with less impact on system performance than concurrent test. Also, because test infrastructure such as scan chains can be shared with manufacturing test, the impact on area overhead is relatively small. In [13], architecture for field test was proposed which has a BIST-based architecture. Even for non-stop systems, which do not often restart, the systems can run at test mode periodically in field. Thus field test like power-on test would be a promising method to detect aging-induced faults. However test quality of the field test has not been considered enough because the target fault model was not well-defined. In addition, there is a strong requirement for the field test that do not go for manufacturing test; that is very short test time, e.g. 10 msec.

Test partitioning with rotating test is an effective way to satisfy constraints on test application time and fault coverage [14]. A given test set for a chip is partitioned into a number of subsets, and apply only one subset to the chip at one field test. Through multiple opportunities of field test, all test patterns of the given test set are applied to the chip, and the subset applied once will be used repeatedly at future test in a rotating manner. In [14], the relation of the number of partitions and coverage was discussed but how to partition a test set was not discussed. Even if the number of partitions that corresponds to test sessions is fixed, fault coverage of partitioned test subsets depends on test vectors in the subsets.

In this paper, we propose an algorithm of test pattern partitioning for the rotating test. When the number of partitions is given, the number of test vectors in each subset is determined. The algorithm partitions the given test set so as to maximize the average fault coverage of subsets obtained by partitioning. Experimental results show that the rotating test is effective for high quality power-on test.

This paper is organized as follows: Section 2 revisits test methods related to this work. Section 3 describes the rotating test to satisfy the requirement of short test time for power-on test. Section 4 formulates a problem for test partitioning and proposes an algorithm to partition a test set into subsets. Section 5 shows some experimental results and Section 6 concludes this paper.

2. PRELIMINARY

2.1 Related works

There are two typical approaches of the field test to test aging-induced faults in a logic circuit. One is concurrent test

that checks the circuit behavior during system operation. For example, self-checking and signature monitoring are well-known techniques [6]. Several techniques were recently proposed, which try to detect unstable signal transition caused by aging-induced faults at flip-flops [7-9]. A disadvantage of these methods based on on-line test is large area overhead and/or performance degradation. In general on-line test relies on a special circuit architecture or redundancies in terms of hardware, time, or information [6]. In [7, 8] the size of the flip-flop with the special structure becomes approximately three times larger than the normal one.

Another approach to test aging-induced faults utilizes BIST-based techniques that are executed at test mode in field [10-16]. Power-on test is a kind of this approach. The methods in [10-12] were developed for chips in automotive, communication, and a processor for servers. In the recently proposed methods, there are ones that an operating system (OS) controls test mode [15, 16]. For example, the method in [15], which is applied for a multi-core processor, tests a core when it is in idle time. However, these dedicated methods that rely on OS are not applicable to SoCs or ASICs.

2.2 Field test

A test architecture for power-on test was proposed in [13] that are available for SoCs with low test cost. Since the field test including power-on test typically takes a BIST-based approach, its limitations are similar to those of BIST for the manufacturing test. For example, limitations and requests on area overhead and test data volume exist for the field test as well as the manufacturing test. Regarding test quality, the content of the requests may be slightly different because the fault model or the location of faults that should be detected is not the same.

The major differences between the field test and the manufacturing test exist in the limitation of test application time and the test opportunity. Due to the requirement of the systems, test application time of the field test is very short, e.g. 10 msec. Therefore, if the number of test patterns is large, it may be impossible to apply all the patterns to the circuit within the required test time. On the other hand, since the BIST-based manufacturing test does not have any technological requirements on test time, its limitation for test application time is less than the field test apart from an economical reason.

Regarding the test opportunity, the power-on test has a peculiar feature. Usually the opportunity of the production test is once just after manufacturing the chip. But since the power-on test is executed every time the system is starting up, its opportunities are more than once. This can be an advantage of the field test as described in the following section.

3. ROTATING TEST

3.1 Concept

If a given test set is too large to apply in one test session, we need to reduce the number of test patterns. It would cause fault coverage loss because of the missing test patterns. However, the test patterns that were not applied at the test session can be applied at the next or later test session, because the field test is executed repeatedly every time the system runs at test mode. Therefore, we partition the original test set into some test subsets as illustrated in Figure 1, and applying one subset for the circuit at one test session. The original test set is partitioned so that the number of test patterns of each subset never exceeds the pattern limit derived from the upper bound of test time, and the number of subsets is as small as possible. Therefore, the number of test patterns of each subset should be $\lceil N_{org}/N_{set} \rceil$ or $\lfloor N_{org}/N_{set} \rfloor$, where N_{org} and N_{set} are the number of test patterns of the original test set and the number of the subsets obtained by partitioning, respectively.

All the test patterns of the original test set can be applied through N_{set} opportunities of the tests. Each subset of test patterns is rotated and applied again at future test sessions, as shown in Figure 2. We call it "rotating test". In the example of Figure 2, sub test set T_1 is applied not only at the first power-on test but also at the $(i \times N_{set}+1)$-th power-on test, where i = 1, 2, 3, …

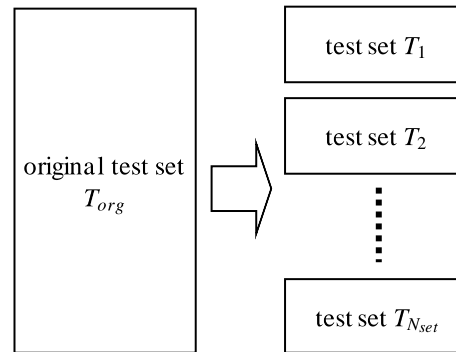


Figure 1. Test partitioning

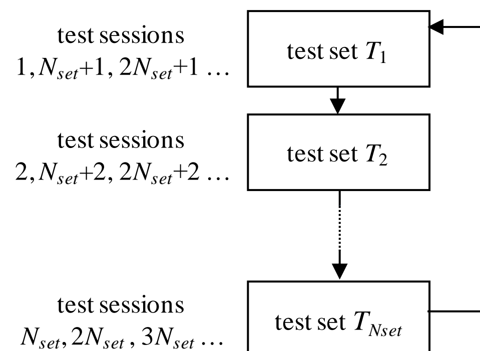


Figure 2. Rotating test

3.2 Quality of test partition for rotating test

Assume that all the subsets obtained by partitioning have the same size as N_{sub}. Then there are $(N_{org}-1)!/(N_{sub}!)^{Nset}$ combinations of partitions. Even though the original test set is the same, test quality of the rotating test could be generally different depending on partitions. We show an example below.

Suppose that we partition a test set T_{org} = {t_1, t_2, t_3, t_4, t_5, t_6} which are generated for nine faults f_1 to f_9 into three subset, i.e., N_{org} =6, N_{set}=3. Hence every subset size is 2 calculated from N_{org}/N_{set}. Each test pattern detects faults as shown in Table 1; for example, t_1 detects three faults f_1, f_2 and f_5. We consider a test partition P_1 as shown in Table 2(a). The subsets T_1, T_2 and T_3 consist of {t_1, t_2}, {t_3, t_4} and {t_5, t_6}. Individual fault coverages of subsets are 44% (= 4/9) for T_1 and T_2, and 33% for T_3, hence the average fault coverage of P_1 is 40.3%. Note that the sum of individual fault coverage for all subsets is more than 100% because faults f_5 and f_8 are detected in two subsets.

Next we consider an alternative test partition P_2 for T_{org} as shown in Table 2(b). In this case, individual fault coverage of subsets are 55%, 55%, 44% for T_1, T_2 and T_3, respectively, and the average fault coverage of partition P_2 is 51.3%, which is larger than that of P_1. It means that P_2 has higher test quality than P_1 because the field test aims at detecting aging-induced faults unlike the manufacturing test. Even if a fault has not occurred yet in a test session, the fault may occur before the next test session. Although a system failure is not caused necessarily as soon as a fault excites, high test quality would be derived if each fault can be detected by more test sessions frequently. Therefore, it is important for the rotating test to find a test partition so that the average fault coverage of individual subsets is as high as possible.

Table 1. Test pattern and detetcted fault

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
t_1	o	o			o				
t_2		o	o						
t_3				o	o			o	
t_4					o	o		o	
t_5							o	o	
t_6								o	o

Table 2. Examples of test partitions

(a) test partition P_1

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	*Flt. cov.*
$T_1(t_1, t_2)$	o	o	o		o					44%
$T_2(t_3, t_4)$				o	o	o		o		44%
$T_3(t_5, t_6)$							o	o	o	33%

(b) Alternative test partition P_2

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	*Flt. cov.*
$T_1(t_1, t_5)$	o	o			o		o	o		55%
$T_2(t_2, t_4)$		o	o		o	o		o		55%
$T_3(t_3, t_6)$				o	o			o	o	44%

4. TEST PATTERN PARTITIONING

4.1 Problem formulation

From the above observations on test quality in test partitioning, we formulate the following problem:

[Test Pattern Partitioning Problem] Given a test set T_{org} consisting of N_{org} patterns and the number of sub test sets N_{set}, find a partition such that

(1) the number of test patterns of each sub test set is $\lceil N_{org}/N_{set} \rceil$ or $\lfloor N_{org}/N_{set} \rfloor$, and

(2) the average fault coverage of individual sub test sets is maximized.

It is easy to satisfy condition (1) because the number of patterns of each subset is uniquely calculated from N_{org} and N_{set}. Therefore our discussion focuses on condition (2) below.

4.2 Upper bound of fault coverage

It is necessary to find an upper bound of the total number of detected faults *TD* so that we can know how far our results are from the possible maximum total detection. Given N_{set} test sessions, a fault can be detected by no more than N_{set} test sessions. Let *ND(f)* represents the number of sessions that detect the fault *f*. The maximum possible *TD* for a given test set T_{org}, N_{set} test sessions and N_F faults can be computed as:

(1) Pick a test pattern t_i from test set *T*, and perform fault simulation without fault dropping.

(2) For every fault f_j, if fault f_j is detected by test pattern t_i, increase $ND(f_j)$ by 1 if $ND(f_j)$ is less than N_S.

(3) Remove t_i from *T*, and go to (1) if *T* is not empty.

(4) Compute *TD* as: $TD = \sum_{i=0}^{N_F} ND(f_i)$.

The maximal possible average fault coverage also can be obtained. Let *Max_TD* denotes the upper bound of total detection as computed above, and *Max_Avg_FC* represents the upper bound of average fault coverage for test sessions. Then $Max_Avg_FC = Max_TD / N_{set}$.

4.3 Algorithm

In this section we explain a test pattern partitioning algorithm developed in this work. We first define a terminology. Given test set *T*, for a couple of test patterns t_i and t_j, if they can detect more the same faults, t_i and t_j are more similar, the number of these faults is defined as the similarity between t_i and t_j. For example of test patterns and faults in Table 1, f_2 is a detected fault of t_1, it also be detected by t_2. The similarity between t_1 and t_2 is 1. The same as t_1 and t_2, the similarity between t_2 and t_3 is 0.

In order to maximize average fault coverage of subsets, we need to partition the test set into subsets so that the test patterns in the same subset do not detect the same faults and the different subsets can detect more the same faults as far as possible. The definition of similarity shows that: the smaller similarity of a couple of test patterns, the more different faults

can be detected. During the partition, we need to comply with a rule: while distributing a pattern to a subset, the similarity between the new pattern and the patterns which already exist in the subset must be the smallest, in the meantime, the similarity between subsets must be the biggest. If we create a complete table as Table 1 shown with respect to test patterns and detected faults, we would have enough information to calculate the similarity. However, it is not efficient on both time and memory usage because it requires fault simulation without fault dropping and a table whose size is $O(N_{org} \times N_{flt})$ where N_{flt} is the number of faults of the circuit.

In order to calculate the similarity of test patterns, we employ fault simulation with fault dropping after a fault is detected N-times where N can be set arbitrarily. We consider that for N_{set} pattern partition, N_{set} - times fault dropping simulation can get enough information. Below is the outline of the test partitioning algorithm for given test set T and the number of subsets N_{set}:

Step 1) For T, perform fault simulation with fault dropping after N-times detection in an arbitrary order of test patterns, and for each fault record the ID of the first N patterns which detect the fault.

Step 2) For every pair of test patterns t_i and t_j ($i \neq j$), count the number of faults which are detected by the two test patterns simultaneously, and then create a two-dimensional table, as table 2.

Step 3) According with the two-dimensional table of similarity, distribute a test pattern into a subset by the following two criteria:

1) The similarity between subsets is large.

   For N_{set} pattern partitions, find out N_{set} test patterns which with the largest similarity between them from the two-dimensional table of similarity, and distribute each of them to different subsets. If M ($M > N_{set}$) test patterns exist, calculate the sum of similarity for t_i ($i \in N_{set}$) and each $t_j \in T$ ($j \neq i$), respectively. Then, find out N_{set} test patterns which with the smaller sum of similarity value, and distribute one of them to a subset.

2) The similarity between test patterns in each subset is small.

   While distributing a test pattern to a subset, from the two-dimensional table of similarity, seek one pattern which with the smallest similarity between this pattern and the patterns which already exist in the subset. If more than one pattern is qualified, calculate the sum of similarity for the pending pattern and the patterns already exist in the other subsets. Distribute the pattern which with the biggest sum value of similarity to the subset.

Step 4) According to the two criteria shown in Step 3, perform the partition procedure until all given test patterns are distributed.

In the fault simulation at Step 1, a fault is dropped from the target fault list when the N-times detection pattern for the fault was found. By fault dropping simulation, a detected fault list for every test pattern can be created. We give an example for test patterns in Table 1. By comparing the fault list of each pair of test patterns, we create a two-dimensional table to record the similarity of every couple of patterns at Step 2. Table 2 is obtained by Step 2. At Step 3, pattern partition must meet the two formulas simultaneously.

Table 3. Similarity of test vector pairs.

	t_1	t_2	t_3	t_4	t_5	t_6
t_1	-	1	1	1	0	0
t_2	1	-	0	0	0	0
t_3	1	0	-	2	1	1
t_4	1	0	2	-	1	1
t_5	0	0	1	1	-	1
t_6	0	0	1	1	1	-

5. EXPERIMENTAL RESULTS

To demonstrate the effectiveness of the proposed test partition algorithm, we implemented it using C language and performed experiments for ISCA'89 and ITC'99 circuits on a PC with Core(TM)2 Duo 2.66GHz, 1.99GB RAM. In these experiments, we used test patterns generated for single stuck-at faults by an in-house ATPG program.

In the first experiment, we partitioned the given compacted test set into eight subsets. Results are given in Figure 3 in which we draw the curve of average coverage for some circuits to compare our method with random partitioning. Y axis shows the average fault coverage, X axis shows the different partitions. Different curves show the result for different circuits. The origin of each curve shows the average fault coverage of subsets partitioned randomly, the other points show the average fault coverage of subsets partitioned by proposed method. We set the detection time of dropping simulation from 2 to 10. From the curves, we can see that for proposed partition method, with the detection time for dropping simulation increase, the average coverage increased. After 4 times dropping simulation the curves become gently, while the detection time for dropping simulation is set to 10, for most circuits, the proposed partition method found the best partition. Table 4 gives the result of proposed method using 10 times fault dropping simulation. The first two columns of the table shows circuit names and the numbers of the given test patterns. The third and forth columns shows the average fault coverage of eight subsets for our partitioning and random partitioning, respectively, and the fifth column gives the difference between two methods. These results show that it is

meaningful for high quality power-on test to find better partition for the rotating test. The last column of the table gives the computing time in second. Since the algorithm is based on fault dropping simulation, test size, circuit size and detection time for dropping simulation dominate the computing time. Table 5 gives the application time of proposed partition while setting the detection time of dropping simulation from 2 to 10. The first column shows the circuit names. From the second column we show the application time while setting the detection time of fault dropping simulation from 2 to 10 respectively. We can see that comparing with 2-times fault dropping simulation, proposed partition by 10-times dropping simulation cost almost 1.5 times application time. Because more accurate similarity table requires more faults information, and more computing time is consumed.

6. CONCLUSIONS

In this paper, we proposed a test partitioning algorithm to provide high test quality, to meet the limitation of short test time in field test. For the rotating test, the algorithm partitioned a given test set into some subsets, and applied only one subset at each test. Each subset is rotated and is applied at future test again. The proposed algorithm aimed at maximizing fault coverage of each subset obtained by partitioning. Experimental results showed that test partitioning for the rotating test is effective for high quality power-on test. As a future work, we are improving the algorithm for test partitioning, because the proposed algorithm in this paper is our first trial and it would not be efficient enough.

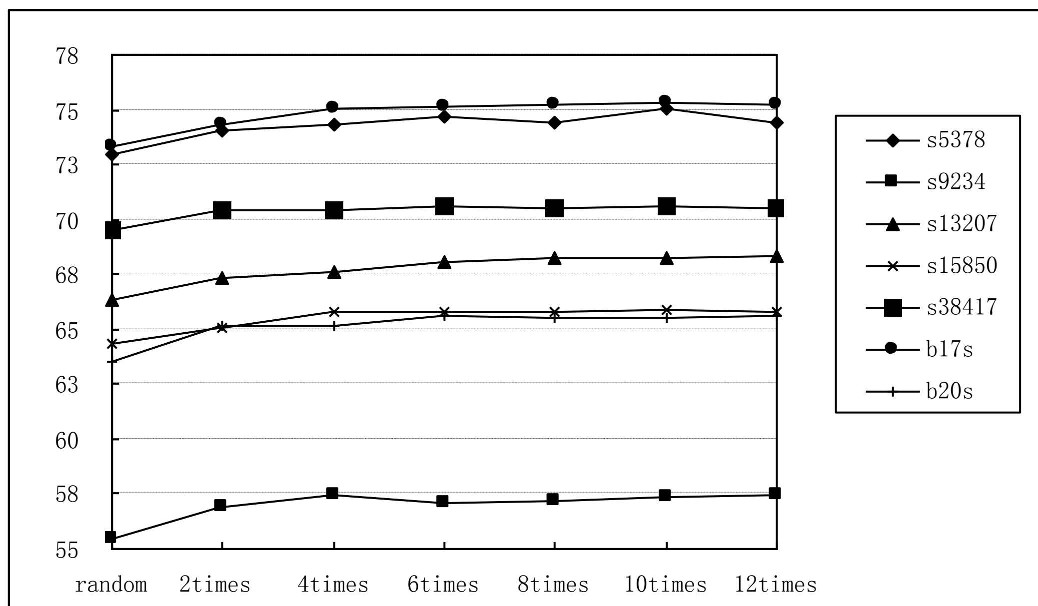


Figure 3. Experimental result for $N_{set} = 8$.

Table 4. Experimental result for random partition and the proposed partition with $N_{set} = 8$.

Circuit	#pattern	Random (%)	Proposed (%)	Diff(%)	Time(s)
s5378	100	72.983	75.057	2.074	1.592
s9234	111	55.427	57.361	1.933	2.177
s13207	235	66.333	68.216	1.883	6.526
s15850	97	64.314	65.867	1.553	5.847
s38417	87	69.534	70.586	1.052	37.570
b17s	1250	73.336	75.334	1.999	359.459
b20s	989	63.514	65.546	2.032	65.367
Average	-	-	-	1.789	-

Table 5. Computing time.

Circuit	Runtime (sec)					
	2times	4times	6times	8times	10times	12times
s5378	0.75	0.92	1.08	1.33	1.59	1.81
s9234	1.18	1.37	1.61	1.92	2.18	2.39
s13207	3.90	4.39	4.91	5.74	6.53	7.41
s15850	2.32	2.89	3.74	4.73	5.85	7.13
s38417	8.70	13.86	20.37	28.53	37.57	50.20
b17s	216.64	235.83	266.09	307.63	359.46	411.81
b20s	56.53	55.42	57.60	63.73	65.37	70.35

REFERENCES

[1] W. Wang, et al., "Compact Modeling and Simulation of Circuit Reliability for 65-nm CMOS Technology," *IEEE Trans. Device and Materials Reliability,* vol.7, no.4, pp.509-517, Dec. 2007.

[2] V. Reddy et al. "Impact of Negative Bias Temperature Instability on Product Parametric Drift," *International Test Conf.,* pp146-155, 2004

[3] S. Bhardwaj, et al., "Predictive Modeling of the NBTI Effect for Reliable Design," *Custom Integrated Circuits Conference*, pp. 189-192, 2006.

[4] V.Chandra, "Early Life Failures and Burn-in Testing," *IEEE Workshop on Design for Reliability and Variability,* Panel, pp.1-10, 2008.

[5] A. H. Baba, S. Mitra, "Testing for Transistor Aging," *VLSI Test Symposium*, pp.215-220, 2009.

[6] M. Nicolaidis, Y. Zorian, "On-line testing for VLSI – A Compendium of Approaches," *Journal of Electronic Testing: Theory and Applications*, vol.12, pp. 7-20, 1998.

[7] T. Nakura, K. Nose, and M. Mizuno, "Fine Grain Redundant Logic Using Defect-Prediction Flip-Flops", *International Solid-State Circuits Conference*, pp. 402-403, 2007.

[8] M. Agarwal, V. Balakrisinan, A. Bhuyan, K. Kim, B. C. Paul, W. Wang, B. Yang, Y. Cao, and S. Mitra, "Optimized Circuit Failure Prediction for Aging: Practicality and Promise," *Int'l Test Conf*, pp.1-10, 2008.

[9] T. Sakata, T. Hirotsu, H. Yamada, T. Kataoka, "A Cost-Effective Dependable Microcontroller Architecture with Instruction-Level Rollback for Soft Error Recovery," *International Conference on Dependable Systems and Networks*, pp. 256-265, 2007.

[10] E. Bohl, Th. Lindenkreuz, R. Stephan, "The Fail-stop Controller AE11," *Int'l Test Conf*, pp.1567-1578, 1997.

[11] S. Dikic, L.-J Fritz, D.Dell'Aquia, "BIST and Fault Insertion Re-use in Telecom Systems," *Int'l Test Conf*, pp.1011-1016, 2001.

[12] J. Braden, Q. Lin, B. Smith, "Use of BIST in FIRE™ Servers," *Int'l Test Conf*, pp.1017-1022, 2001.

[13] Y. Sato, et al., "A Circuit Failure Prediction Mechanism (DART) for High Field Reliability," *8th IEEE Int'l Conf. on ASIC,* pp. 581-584, Oct.20-23, 2009.

[14] J. P. Robinson, "Segmented Testing," *IEEE Trans. Computers,* Vol. C-34, No. 5, pp. 467-471, May 1985.

[15] Y. Li, S. Makar, and S. Mitra, "CASP: Concurrent Autonomous Chip Self-Test Using Stored Test Patterns," *Design Automation and Test in Europe,* pp. 885-890, 2008.

[16] O. Khan, and S. Kundu, "A Self-Adaptive System Architecture to Address transistor Aging", *Design Automation and Test in Europe*, pp. 81-86, 2009.

[17] J. –S. Chang and C. –S. Lin, "Test Set Compaction for Combinational Circuits," IEEE Trans. on Computer-Aided Design of ICs & Systems.,vol. 14, no. 11, pp. 1370-1378, 1995.

Optimal Universal Test Set for Bridging Faults Detection in Reversible Circuit Using Unitary Matrix

Pradyut Sarkar
Department of Research & Development of VLSI Technology
Simplex Infrastructures Limited, Kolkata, W.B, India (e-mail : pradyut_sarkar77@yahoo.com).
Bikromadittya Mondal
Dept. of Computer Sc. & Engg.
B. P. Poddar Institute of Management and Technology, Kolkata, W.B., India, bikmondal@gmail.com
Susanta Chakraborty
Dept. of Computer Sc. & Technology & School of VLSI Technology,
Bengal Engg. & Science University, Sibpur, W.B ,India
Communicating Author and Supervisor(e-mail: susanta_chak@yahoo.co.in)

Abstract— Detection of bridging fault in reversible circuit has received considerable interest in quantum computation. The single and multiple inputs bridging fault model of a reversible circuit is considered here. This paper presents a novel universal test set generation method of a reversible logic circuit based on the shift operation on unitary matrix. Unitary matrix is mapped to the identity matrix by that shift operations. It is shown that proposed $\lceil n/2 \rceil$ (n is the number of inputs) numbers of universal test vectors are sufficient for detection of all single and multiple input bridging faults of a n-input and n-output reversible circuit. A polynomial time algorithm is proposed for generating the universal test vectors to detect the all single and multiple input bridging faults of the reversible circuits. The experimental results on reversible benchmark circuits show that the number of universal test vectors are significantly reduced compared to the earlier works.

Keywords- Reversible circuit, Single and multiple input bridging faults, Unitary matrix, Identity matrix, Universal test set.

1. Introduction

Reversible circuits can be focused as a special case of quantum circuits because quantum computations are inherently reversible in nature [8], [6]. Reversible logic has a vital role in low power design, quantum computing, nanotechnology and cryptography. Quantum circuit is represented by a qubit. The unit of quantum information is called a *qubit*. A qubit can be in a zero or a one state represented by |0> and |1> respectively. A single qubit can also be a superposition of these states i.e $\alpha_0|0\rangle + \alpha_1|1\rangle$, where α_0 and α_1 are complex numbers called amplitude. An n-input, m-output Boolean function F is said to be reversible if and only if m = n, and F is bijective. A reversible combinational circuit must be fan-out free, acyclic, and should consist of only reversible gates. Various physical realizations of quantum gates are based on trapped ion technology [14], which uses certain spin and vibrational modes of electrically charged atoms as a qubit and nuclear magnetic resonance (NMR). Quantum computation can solve exponentially hard problems in polynomial time [8]. Most gates used in classical digital design are not reversible. Reversible gates can be used to design a reversible circuit for example the controlled –NOT (CNOT) gate proposed by Feynman [3], Toffoli gates [9], and Fredkin [4] gates. Syntheses of reversible logic circuits have studied in [17-19]. Universal testability of reversible logic circuit for detecting single and multiple stuck-at faults model have been investigated [12][13]. Recently several researchers have studied the different fault models. Polian et al has presented [14] single missing gate, multiple missing gate, partial missing gate and repeated missing gate faults models. In this paper, we have assumed that fault model may be classical AND/OR-bridging fault model. A simple polynomial time algorithm is described here for generating a minimal universal test set for detecting all single and multiple input bridging faults in a n-input reversible circuit. The paper addresses that, only $\lceil n/2 \rceil$ numbers of universal test vectors are sufficient for detection of all single and multiple input bridging faults in an n-input reversible circuit as compared to earlier methods [20]. Although actual implementation of the bridging fault model is yet to be established but the results may be applicable to the future emerging technologies of reversible logic circuit.

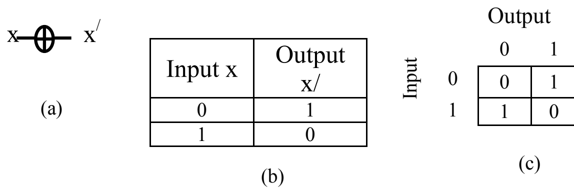
2.Preliminaries

Reversible gate: A gate is reversible, if the (boolean) function it computes is bijective. A necessary condition is that the gate has the same number of inputs and outputs and every distinct input gives a distinct output.

Unitary Matrix: A unitary matrix is a square matrix U whose entries are complex numbers and whose inverse is equal to its conjugate transpose U^*. This means that $U^*U = UU^* = I$, where U^* is the conjugate-transpose of U and I is the identity matrix. A unitary matrix in which all entries are real is the same thing as an orthogonal matrix. Just as an orthogonal matrix G preserves the (real) inner product of two real vectors, thus $\langle G\mathbf{x}, G\mathbf{y}\rangle = \langle \mathbf{x}, \mathbf{y}\rangle$, so also a unitary matrix U satisfies $\langle U\mathbf{x}, U\mathbf{y}\rangle = \langle \mathbf{x}, \mathbf{y}\rangle$ for all *complex* vectors x and y, where $\langle .,. \rangle$ stands now for the standard inner product on $\mathbf{C}^n$.

Reversible logic gates are represented by unitary matrices as shown below. The most common reversible gates operate on spaces of one or two qubits, just like the common classical logic gates operate on one or two bits. This means that as matrices, reversible gates can be described by 2 × 2 or 4 × 4 unitary matrices. The NOT (Fig 1) is a one input one output gate. It inverts the input. The CNOT gate (Fig 2) is a 2x2

gate. The value at the first input is left unchanged, and the value on the second input is inverted if and only if the value at the first input is 1, else remains unchanged. The Toffoli gate (2-CNOT) (Fig 3) is a 3x3 gate. It passes the first two inputs through and inverts third if the first two are both 1, else remain unchanged and a Fredkin gate(Fig 4) is a 3x3 reversible gate. It is a controlled-SWAP gate. It passes the first input through. If the first input is 0, passes the second and third inputs through, otherwise swap the second and third inputs.

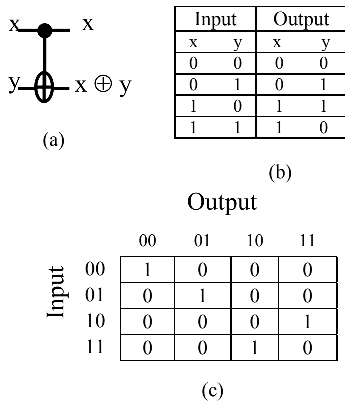


Input x	Output x/
0	1
1	0

(b)

	0	1
0	0	1
1	1	0

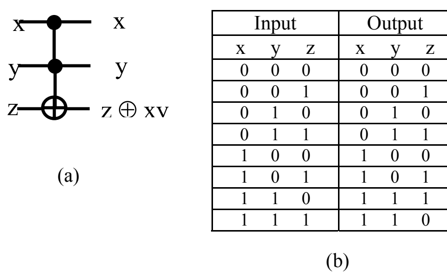
Fig 1: NOT gate representation (a) Symbol, (b) Truth table, (c) Unitary matrix



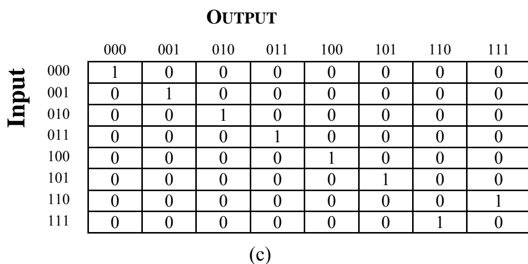
Input		Output	
x	y	x	y
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

	00	01	10	11
00	1	0	0	0
01	0	1	0	0
10	0	0	0	1
11	0	0	1	0

Fig 2: CNOT gate representation (a) Symbol (b) Truth table (c) Unitary matrix

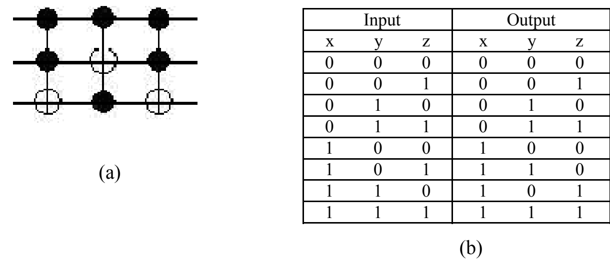


Input			Output		
x	y	z	x	y	z
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

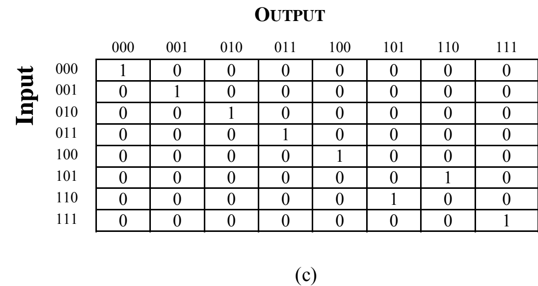


	000	001	010	011	100	101	110	111
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	0	0	0	1	0	0
110	0	0	0	0	0	0	0	1
111	0	0	0	0	0	0	1	0

Fig 3: 2-C-NOT gate representation (a) Symbol, (b) Truth table, (c) Unitary matrix



Input			Output		
x	y	z	x	y	z
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	1	1	1

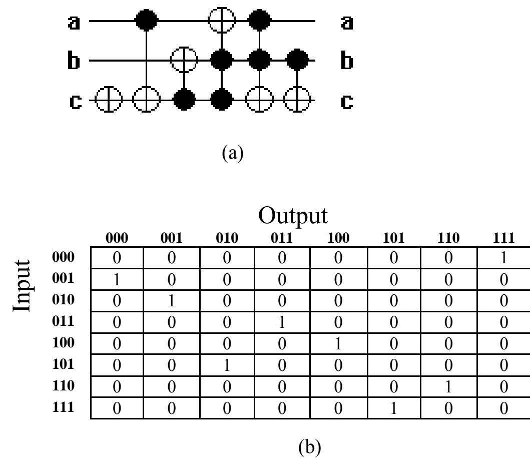


	000	001	010	011	100	101	110	111
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	0	0	0	0	1	0
110	0	0	0	0	0	1	0	0
111	0	0	0	0	0	0	0	1

Fig 4: Fredkin gate representation (a) Symbol, (b) Truth table, (c) Unitary matrix

2.1 Reversible Circuit represented by Unitary Matrix

Unitary matrix of a reversible circuit (Fig 5a) is shown in Fig 5b. The Unitary matrix of a reversible circuit is obtained from the output function of the circuit or multiplying the unitary matrices of the reversible logic gates present in the circuit.



	000	001	010	011	100	101	110	111
000	0	0	0	0	0	0	0	1
001	1	0	0	0	0	0	0	0
010	0	1	0	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	1	0	0	0	0	0
110	0	0	0	0	0	0	1	0
111	0	0	0	0	0	1	0	0

Fig 5: (a) A 3-qbit reversible benchmark circuit (b) Unitary matrix of the reversible circuit.

3. Bridging Fault Model in Reversible Circuit

A test generation problem of a reversible circuit is simpler than that of classical circuit due to inherent ease of controllability and observability and always yields an unique vector at the input. Bridging faults are caused by shorts between two (or more) lines. All the lines are involved in a bridging fault may be classical AND-OR type. A bridging fault is activated by input vector t, if the two lines are assigned opposite logic values ('01' or '10') then the error can be propagated to the output of the reversible circuit.

Since a reversible circuit is bijective, fault free output is different from faulty output of a reversible circuit. A single input bridging fault (SIBF) is a bridging fault, involving two input lines of a circuit. *A* multiple input-bridging fault (MIBF) is a bridging fault involving more than two input lines of a circuit.

3.1. Universal test set for bridging fault

A fault set F, generates a set of test vectors T that can be used to detect all faults (set of all possible single and multiple input bridging faults) of the reversible circuit. Such a test set is said to be complete test set. A universal test is a complete test set cover the set of all possible single and multiple input bridging faults of all reversible circuit.

4. Proposed Test Set Generation Method

We propose a technique to generate a universal test set based on the shift operation of unitary matrix that represents a reversible circuit**.** Consider an 2^n x 2^n fault free unitary matrix(UM) of a n qbit reversible circuit. The unitary matrix can be converted into an identity matrix by right circular shifting of '1' in each row of the unitary matrix. Count the number of right circular shifting of '1' in each row of the unitary matrix and stored these shifting into a matrix which is the fault free shift matrix(SM). Now consider all the combinations of the single input bridging faults for a given circuit. For each bridging fault, find the faulty unitary matrix(UM_f) and covert it into identity matrix by right circular shifting of '1' 'in each row. Count the number of right circular shifting of '1' in each row of the faulty unitary matrix for each bridging fault and represent those shifting as a faulty column matrix which is termed as faulty shift matrix(SM_f). A test vector '01' or '10' can detect the bridging fault between 'x& y', which has different row shifting as compared to fault free shift matrix, whereas rest of the rows will have the same shifting values. The test vectors for each bridging fault can be obtained by comparing the faulty shift matrix with the fault free shift matrix.

4.1 Proposed Test Pattern Generation Algorithm

Step1 : Consider a fault free unitary matrix UM[2^n] [2^n] of order (2^n*2^n) of a n-qbits reversible circuit as an input.

Step2 : for i = 1 to 2^n -1
For j= i to 2^n -1
{
Right circular shifting (rcs) ← 0;
Right circular shifting(rcs)) ← right circular shifting + 1 (Until the unitary matrix UM[i] [j] is converted to identity matrix by right circular shifting of '1' in each row)
fault free shift matrix SM[0][i]= rcs;
}

Step3: Consider all the combinations of bridging faults.

Step 4: For each bridging fault consider a faulty unitary matrix UM_f[2^n] [2^n] of order (2^n*2^n) of a n-qbits reversible circuit

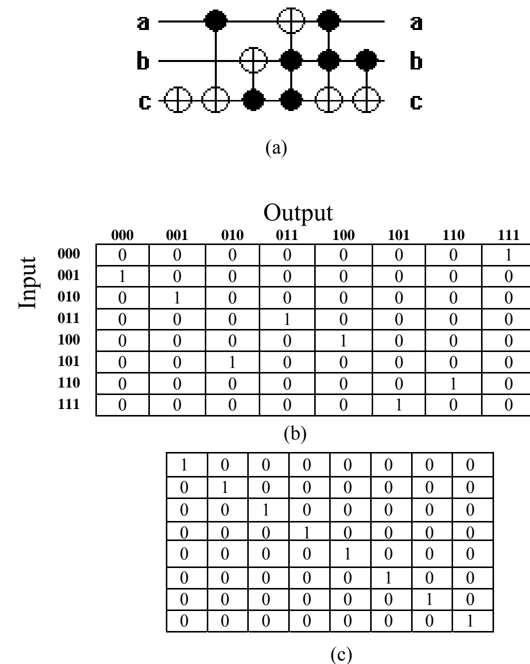
Step5 : for i = 1 to 2^n –1
for j= i to 2^n -1
{
Right circular shifting (rcs_f) ← 0;
Right circular shifting(rcs_f) ← right circular shifting + 1(until the unitary matrix UM_f[i] [j] is converted to identity matrix by right circular shifting of '1' in each row).
faulty shift matrix SM_f[0][i] = rcs_f;
}

Step7: If SM [0][i] ≠ SM_f[0][i] then the ith test vector can detect the bridging fault.
Count the # of bridging faults can be detected by each test vector.

Step8: Find the first test vector from the top that can detect more than half of all possible bridging faults.
This test vector will be the initial test vector T(0).

Step 9 : do
T(i+1) = T(i) << 1.
Return T(i).
While(the bit next to MSB is 0).

Example: Consider a three inputs benchmark reversible circuit shown in fig 6



(a)

Output

Input	000	001	010	011	100	101	110	111
000	0	0	0	0	0	0	0	1
001	1	0	0	0	0	0	0	0
010	0	1	0	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	1	0	0	0	0	0
110	0	0	0	0	0	0	1	0
111	0	0	0	0	0	1	0	0

(b)

1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1

(c)

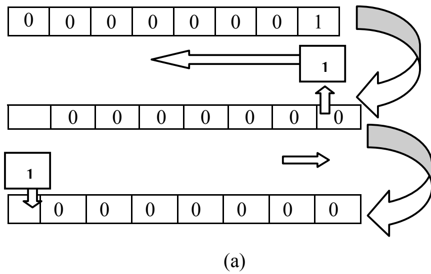
Fig 6:(a)3_17tc_tfc benchmark circuit, (b)The Output Unitary Matrix (UM) (c) The Identity Matrix (UI)

Consider the 1st row of the output Unitary Matrix of the above circuit and is represented by a row matrix(UM_1).

UM_1=	0	0	0	0	0	0	0	1

One bit right circular shifting is required here to convert it into the first row of the Identity Matrix (UI_1). So the value in

the first row of the Shifting Matrix(SM_1) is 1. Tree representation of the proposed method shown in Fig 7.



(a)

UI_1 =

1	0	0	0	0	0	0	0

(b)

SM_1 = 1

(c)

SM=
1
1
1
0
0
3
0
2

(d)

Fig 7: (a) Tree Representation (b) First row of the Input Identity Matrix (UI) of the above circuit (c) First row of the shift Matrix. (d) Fault free Shift Matrix(SM)

Similar shifting method has been applied for the rest of the rows of the Unitary Matrix (UM) and obtained input identity matrix (UI). The number of shifting needed store into a shift matrix (SM) as shown in fig 7(d). Now consider all the combinations of bridging faults.

Case 1: 'ab' bridging fault

For 'ab' bridging fault find the faulty unitary matrix. Now convert it into the identity matrix by right circular shifting of '1' in each row. Store these numbers of shifting into a shift matrix, which is the faulty shift matrix of 'ab' bridging fault as shown in Fig 8.

$UM_{f=ab}$ =

0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0
0	0	0	0	0	1	0	0

(a)

$SM_{f=ab}$ =
1
1
3
3
5
5
0
2

(b)

Fig 8. (a)The Output Unitary Matrix of faulty circuit (b) Faulty Shift Matrix for 'ab' bridging fault.

Case 2: 'bc' bridging fault

For 'bc' bridging fault find the faulty unitary matrix. Now convert it into the identity matrix by right circular shifting of '1' in each row. Store these numbers of shifting into a shift matrix, which is the faulty shift matrix of 'bc' bridging fault as shown in Fig.9.

$UM_{f=bc}$ =

0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0

(a)

$SM_{f=bc}$ =
1
2
3
0
0
1
2
2

(b)

Fig 9. (a)The Output Unitary Matrix of faulty circuit (b) Faulty Shift Matrix for 'bc' bridging fault.

Case 3: 'ac' bridging fault

Similarly for 'ac'bridging fault, the Output Unitary Matrix of the faulty circuit and the corresponding Shift Matrix is shown in Fig.10 .

$UM_{f=ac}$ =

0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	1	0	0

(a)

$SM_{f=ac}$ =
1
2
1
2
5
3
5
2

(b)

Fig 10. (a)The Output Unitary Matrix of faulty circuit (b) Faulty Shift Matrix for 'ac' bridging fault.

Now we compare the number of shifting of the fault free matrix (Fig 7d) with each of the faulty matrix (fig 8b, fig 9b, fig 10b) to find the number of bridging faults detected by each test vectors as shown in Table 1.

Table 1: Number of shifting of the fault free and the faulty matrix

Input abc	#Shifting in the fault free matrix	#shifing for ab bridging fault	#shifting for ac bridging fault	#Shifting for bc bridging fault	# faults detected
000	1	1	1	1	0
001	1	1	2	2	2
010	1	3	1	3	2
011	0	3	2	0	2
100	0	5	5	0	2
101	3	5	3	1	2
110	0	0	5	2	2
111	2	2	2	2	0

We find that the input test vector (T_0= 001) can detect more than half of all possible bridging faults, which is the initial test vector. The next test vector (T_1) is obtained by one bit left shifting of the initial test vector until the bit next to MSB is 0 that is (010 ← 001). These two test vectors (001,010) cover all the input bridging faults (four) of the above circuits as shown in Table 2. Hence the universal test set is, T_U={T(0),T(1)} = {001,010}.

Table 2: Detection of Bridging fault of 3_17tc_tfc benchmark circuit

Bridging Between	Test set applied	Desired output	Output in case of OR-bridging	Output in case of AND bridging
	a b c	a b c	a b c	a b c
a ◄► b	1 0 1	0 1 0	**1 0 1**	0 **0** 0
a ◄► c	1 1 0	1 1 0	1 **0 1**	**0 0 1**
b ◄► c	1 1 0	1 1 0	1 **0 1**	1 **0 0**
a ◄►b◄► c	1 1 0	1 1 0	1 **0 1**	1 1 **1**

Theorem 1: Only $\lceil n/2 \rceil$ numbers of universal test vectors are sufficient for detecting all single and multiple input bridging faults of a n-input reversible circuit.

Proof : Let us consider the test patterns are T(0),T(1)…T($\lceil n/2 \rceil$) where first test vector T(0) = 0…0($\lceil n/2 \rceil$ times)1…1((n-$\lceil n/2 \rceil$)times), contains $\lceil n/2 \rceil$ number of 0s and (n-$\lceil n/2 \rceil$) number of 1s, which can detect bridging faults between the first $\lceil n/2 \rceil$ number of inputs and the rest of the inputs. Second test vector T(1) = 0…0(($\lceil n/2 \rceil$ -1)times)1…1((n- $\lceil n/2 \rceil$)times) 0, contains ($\lceil n/2 \rceil$-1) number of 0s, followed by (n-$\lceil n/2 \rceil$) number of 1s, followed by single zero, which can detect bridging faults between the first ($\lceil n/2 \rceil$-1)number of inputs, last input and the other inputs. Similarly the $\lceil n/2 \rceil$ th test vector T($\lceil n/2 \rceil$) = 01…1((n-$\lceil n/2 \rceil$)times) 0…0(($\lceil n/2 \rceil$-1)times) , which can detect bridging faults between the first input, last input and the other remaining inputs. All of the above test vectors are also sufficient to detect all multiple input bridging faults. We conclude that O($\lceil n/2 \rceil$) numbers of generated test vectors are sufficient for detecting all single and multiple input bridging faults of the reversible circuit.

5. Experimental Results

Proposed algorithm has been applied to reversible benchmark circuits (6sym, 9sym, hwb6, hwb7, hwb8, rd73, rd84, ham7, ham15, Mod1024adder, Mod1048576adder). Column 1,2,3 and 4 of Table 3, represent circuit's name, input / output, number of stuck-at faults and bridging faults (Single, Multiple) respectively. Column 5 represents number of universal test vectors of the earlier methods [20], Column 6 and 7 represents the number of universal test vectors as per our method and % of reduction of the test vectors. Experimental results show that the proposed method reduces approximately 95% of test vectors as compared to the earlier methods [20].

6. Conclusions

This paper introduces a new novel method of generating the minimum universal test vectors for detecting all single and multiple input bridging faults. The proposed method reduces the number of test vectors significantly as compared to the earlier methods. The Results on benchmark circuits show that only $\lceil n/2 \rceil$ numbers of universal test vectors are sufficient to detect all single and multiple input bridging faults. Detection of faults in reversible circuits using Genetic algorithm may be the future research of interest.

References

[1] H. Bahrman, J. Tromp and P. Vitanyi, "Time and Space Bounds for Reversible Simulation", *Journal of Physics A: Mathematical and General*, vol.-34, pp. 6821-6830, September 2001.

[2] M. Li, J. Tromp, and P. Vitanyi, "Reversible Simulation of Irreversible Computation", *Physica D*, pp. 168-176, September 1998.

[3] B. Desoete and A. De Vos, "A Reversible Carry-Look Ahead Adder Using Control Gates", *Integration, The VLSI Journal*, vol.-33, pp. 89-104, 2002.

[4] M. A. Nielsen and I. L. Chuang, "Quantum Computation and Quantum Information", Cambridge University Press, 2000.

[5] A. De Vos, "Towards reversible digital computers", *Proceedings of European Conference on Circuit Theory and Design, Budapest*, pp.923-931, 1997.

[6] B. Desoete, A. De Vos, M. Sibinski and T. Widerski, "Feynman's reversible logic gates implemented in silicon", *Proceedings of 6th International Conference MIXDES*, pp.496-502, 1999.

[7] P. Picton, "Optoelectronic, multi-valued, conservative logic", *International Journal of Optical Computing,* 2, pp.19-29, 1991.

[8] P. Picton, "A universal architecture for multiplevalued reversible logic", *MVL Journal*, pp.27-37, 2000.

[9] J. A. Smolin, and D. P. DiVincenzo, "Five two-bit quantum gates are sufficient to implement the quantum Fredkin gate", *Physical Review A*, 53, pp.2855-2856, 1996.

[10] A. Peres, "Reversible Logic and quantum Computers", *Physical Review A*, 32, pp.3266-3276, 1985.

[11] R. C. Markle, "Two types of mechanical reversible logic", *Nanotechnology*, 4, pp.114-131, 1993.

[12] Patel K.N., Hayes J.P., and Markov I.L., "Fault testing for reversible circuits", *VLSI test Symposium.* pp. 410-416, 2003.

[13] Hayes J.P., Polian I., and Becker B., "Testing for missing-gate faults in reversible circuits" *Asian Test Symposium*, pp. 100-105, 2004.

[14] Polian I., Hayes J.P., Fiehn T., Becker B., "A Family of Logical Fault Models for Reversible Circuits", *Asian Test Symposium*, pp. 422-427, December 2005.

[15] J. Zhong, J.C. Muzio, " Analyzing fault models for reversible logic circuits", *IEEE congress on Evol. Computation*, pp. 2422-2427, 2006.

[16] M. P. Frank, "Introduction to Reversible Computing: Motivation, Progress and Challenges", *Computing Frontiers*, pp.285-390, 2005.

[17] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, " Synthesis of Reversible Logic circuits", *TCAD,* Vol. 22(6), pp.710-722, 2003.

[18] M. Saeedi, M. Saheb Zamani, M. Sedigh, "On the Behavior of Substitution – Based Reversible Circuits Synthesis Algorithms: Investigating and Improvement", *ISVLSI*, 2007.

[19] M. Saeedi, M. Sedighi, M. Saheb Zamani, " A Novel Synthesis Algorithm for Reversible Circuits", *ACM International Conference on CAD*, 2007.

[20] H. Rahaman, D.K. Kole, D.K. Das, B.B. Bhattacharya, "Optimum Test Set for Bridging Fault Detection in Reversible Circuits", 16th Asian Test Symposium , pp. 125-128, 2007.

Table 3 : Detection of Single and Multiple Bridging Faults

Benchmark Circuits	# Input/ Output	Total Faults		Test Vectors as[20] for Reversible Circuits	Proposed universal test vectors	% of reduction
		# Stuck-at Faults	# Bridging Faults(Single, Multiple)	# Test Vectors	# Test Vectors	
6sym	6/1	12	90	60	3	95%
9sym	9/1	18	352	87	5	94.24%
hwb7	7/7	14	152	375	4	98.93%
hwb8	8/8	16	238	NA	4	NA
hwb6	6/6	12	90	375	3	99.2
rd73	7/3	14	152	63	4	93.7
rd84	8/4	16	238	84	4	95.29
ham7	7/7	14	152	89	4	95.51
ham15	15/15	30	1848	524	8	98.48
mod1024 adder	20/20	40	4598	252	10	96.1
mod 10485 76 adder	40/40	80	21222	NA	20	NA

On The Design of Self-Recovering Systems

Yang Lin and Mark Zwolinski
School of Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ, UK
Email:{yl5g09,mz}@ecs.soton.ac.uk

***Abstract*—Conventional fault tolerance techniques require at least 100% hardware overhead. We propose a self-recovery method that checks the status flags from datapath registers. We demonstrate its effectiveness by simulating a number of errors in a modified processor architecture. The hardware overhead is significantly reduced, but there is still a performance overhead.**

***Index Terms*—Fault tolerance, on-line test, self-checking, self-repair.**

I. Introduction

With shrinking feature sizes and increasing levels of integration, the reliability of CMOS integrated circuits is becoming a cause for concern. In traditional safety-critical systems, techniques such as Triple Modular Redundancy (TMR) have been employed, but the overhead is far too great for consumer electronics. Very regular structures, such as memory, can include error detection and correction with much lower overheads, but such methods are not applicable to general logic.

The motivation for this work is to find methods for error detection and correction that impose a relatively low overhead (perhaps 10%) in terms of area and performance. We are interested in general-purpose digital systems that can be characterised as having two parts – a datapath, that does the processing; and a controller that ensures that operations in the datapath happen in the correct sequence. We assume that the datapath is significantly larger than the controller. Hence, we also assume that an error in the controller needs to be detected and corrected immediately, but that an error in the datapath may not need immediate action. In other words, detecting an error some time after it occurs may be sufficient and may require a smaller hardware overhead.

This paper is organised as follows. The next section describes the background to this work. In section III, we describe the error correction and correction technique. In section IV, we show how the method can verified by simulating a number of faults. Finally, we conclude the paper in section V.

II. Background and Previous Work

The work described here is an extension to that published in 2006 [1]. For convenience, we summarize that work here. A general model of a digital system is given in Fig. 1. The controller is a state machine that generates control signals to enable functional units within the datapath. In turn, the datapath generates condition signals that drive the next state logic of the controller.

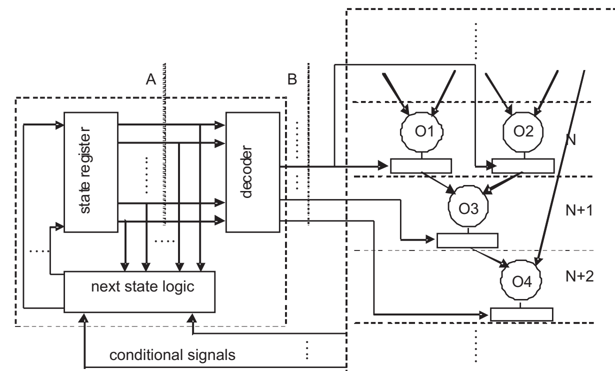


Fig. 1. Controller/Datapath Interaction.

In previous research concerning self-checking controllers, the state signals were encoded according to some error-detecting and/or correcting scheme, such as parity, single-error correcting Hamming code, constant Hamming distance, or even physical duplication. All self-checking takes place at the state register outputs (point “A” of Fig. 1). Any possible faults in the decoder are not considered, and would therefore corrupt the decoded control signals. Consequently, if robust reliability properties are to be maintained, it is highly desirable that controller testing take place *after* the decoding operation, that is on the raw control signals (at point “B” of Fig. 1).

To enable self-checking in the datapath, there needs to be redundancy. One method, proposed in other work of ours [2], is to duplicate operations on different instances of hardware. Ideally the duplication would occur in different clock cycles, using hardware that would otherwise be idle. Thus, if there is sufficient hardware capacity, the cost of duplication would be zero, although there remains the cost of the comparators and checkers. In practice, there is usually insufficient spare resource in the system for the cost to be that low.

We proposed a number of different schemes for making the controller self-checking. Not all of these were totally self-checking. A typical result is shown in Table I. The original design needs nearly 4000 gates and runs at nearly 40MHz. Adding temporal duplication and self-checking to the datapath incurs an overhead of 67% in terms of area. While the clock speed remains the same, the number of cycles needed to complete one operation increases by nearly 40%. Adding a

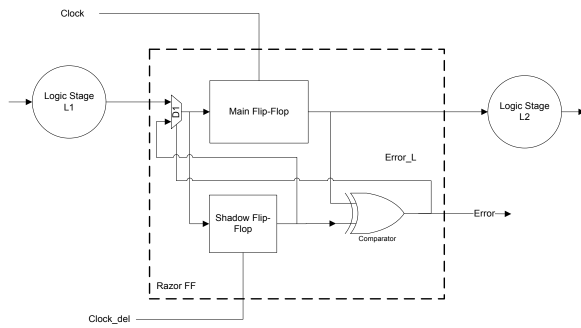


Fig. 2. Razor Flip-Flop Structure.

Totally Self-Checking (TSC) controller slows the clock speed down a little, but the *marginal* increase in the area overhead is only 4.4% (164 gates). While this example may be small, it does suggest that focussing on the controller is likely to be more beneficial than a more general approach.

Hence, the basic idea here is to extend the concept of a TSC controller to also consider the status of the condition signals fed back from the datapath to the controller. As it is not possible to determine the error status of a signal from a single bit, the condition signals need to be dual-rail encoded. Thus, for example, 01 and 10 might be valid states for one condition flag, while 00 and 11 would indicate an error.

To allow dual-rail encoding of selected condition registers, we have adapted the shadow flip-flop model of Razor [3], Fig. 2. In Razor, a copy of a signal is stored in the shadow flip-flop, but the copy is made some time after the primary value is stored – perhaps by using the succeeding "inactive" clock edge. This protects both against timing delays and against errors in the main flip-flop (or indeed, in the shadow flip-flop). The values in the main and shadow flip-flop now form a two-bit code word that can be fed back into the controller.

III. Self-recovering processor

In order to determine whether extending a TSC controller to include self-checking on the condition signals is sufficient to give us a self-recovering design, we have used the SAYEH processor [4] as a test exemplar. To illustrate the technique, we have chosen two flag signals generated by the datapath: the z_flag generated by the ALU to indicate a zero-valued result; and the c_flag, generated by the CPU to indicate a carry. The function of the two flags is slightly different – both are used to indicate a status to the controller, but the c_flag is additionally used in subsequent calculations.

The Verilog code in the SAYEH processor is modified to include shadow registers for the z_flag and the c_flag, both of which are triggered by the negative edge of the clock. We also include a backup register for the c_flag, which will be explained below.

Because we require the system to be self-correcting, it must be able to recover when an error is detected. As stated above, we assume that it is sufficient to detect an error eventually, but not necessarily as soon as it occurs. Hence the effect of an error (in the form of inconsistent values for the z_flag or the c_flag) may be observed several cycles later and hence the results of a computation may be incorrect. Thus, to recover from an error, it is necessary to back track to a known good state and to repeat the computation. (Of course, if the error is permanent, this may be futile, but for the purposes of this study, we assume that any error is transient.)

We therefore need to make further modifications to the system. First, data registers need to be backed up. For the purposes of this study, we are only considering two status flags, thus it is sufficient to back up the register files associated with the ALU and, as noted, the c_flag because it is used in subsequent computations. Second, the controller functionality needs to be modified to act on errors in the z_flag and the c_flag. An example of this is shown in Fig. 3. Additional states are added to force a re-execution in the event of a mismatch in values in a status flag. In this example, the behaviour of different instructions may be classified into three types, depending on the effect on different registers. Thus, three different forms of this modification are needed, although it is possible to optimise the resulting state machines and to reduce the number of extra states. Finally, the controller itself needs to be made TSC, as described in [1]. For the purposes of this exercise, we limited these modifications to the control signals from the controller to the datapath. 48 control signals had 6 parity bits added to make them self-correcting.

Notice that, although we use the Razor flip-flop structure, this technique works at a different level to Razor. Within a Razor-enabled processor, faults are detected and corrected within the datapath (in particular within the pipeline of a processor). The controller is not aware of any fault detection or correction. Here, the controller is responsible for fault detection and correction. Hence, the two methods are not directly comparable and may indeed be complementary. This is a topic for further research.

IV. Verification

The self-recovery features of the modified design were verified by simulating an RTL model. In addition to the modifications described above, fault injection mechanisms were included. Three different types of fault were modelled:

- Transient faults in the datapath, leading to errors in the flags.
- Timing errors in the datapath, such that a value is delayed by more than one clock cycle.
- Transient faults in the control signals.

Some examples are given here. Fig. 4 shows the effect of a transient fault in the datapath while executing an scf (set carry flag) instruction. The error is injected at 440ns for one clock cycle. This leads to an error in the c_flag on the next rising clock edge, causing the instruction to be fetched and executed again.

Fig. 5 shows a timing fault injected during the execution of a sub instruction. At 5700ns, the data misses the rising clock edge and is not latched. The system goes into recovery mode.

TABLE I
DIFFEQ BENCHMARK SYNTHESIS RESULTS (ALCATEL CMOS 0.35 VLSI)

data path testing	control path testing	area (gates)	speed (cycles)	maximum frequency (MHz)	hardware overhead (gates %)	speed penalty (cycles %)
-	-	3679	16	39.6	N/A	N/A
present	-	6143	22	39.6	67.0	37.5
present	method 6	6307	22	37.6	71.4	37.5

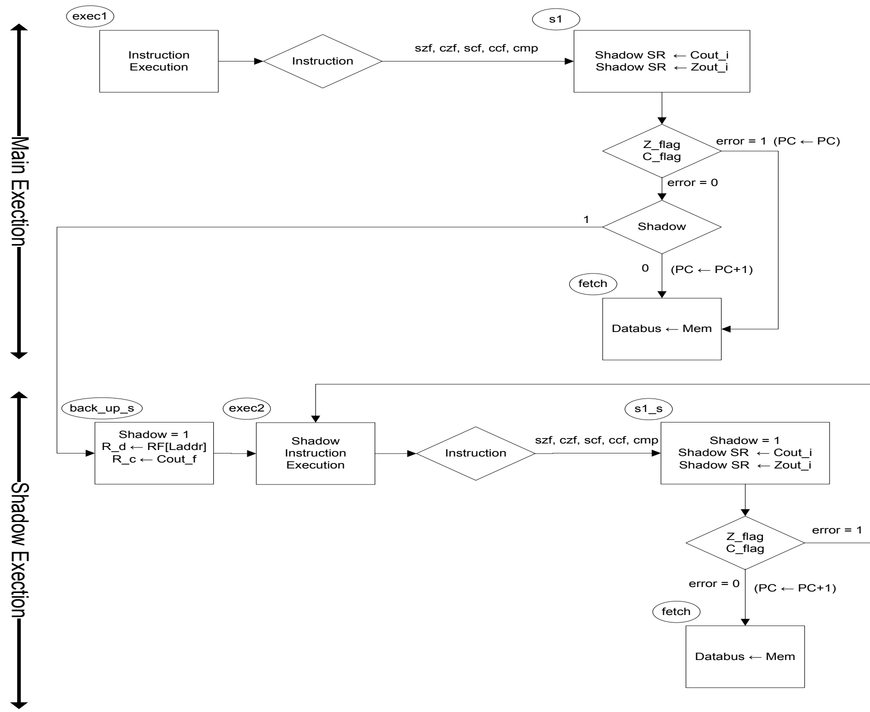


Fig. 3. Modified Controller.

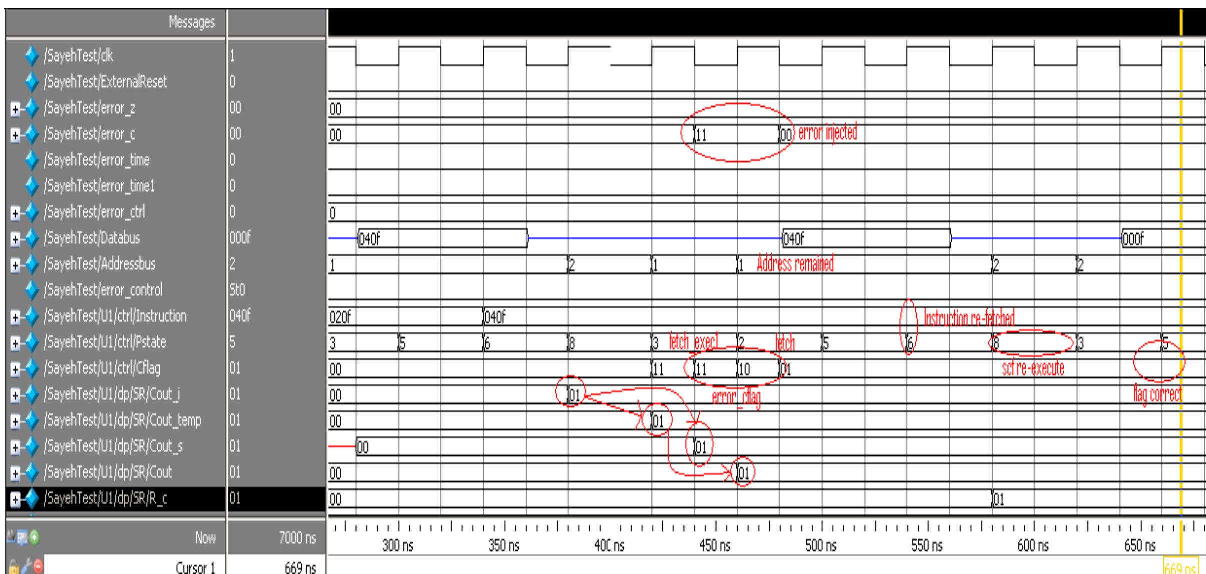


Fig. 4. Operation of instruction scf with transient fault.

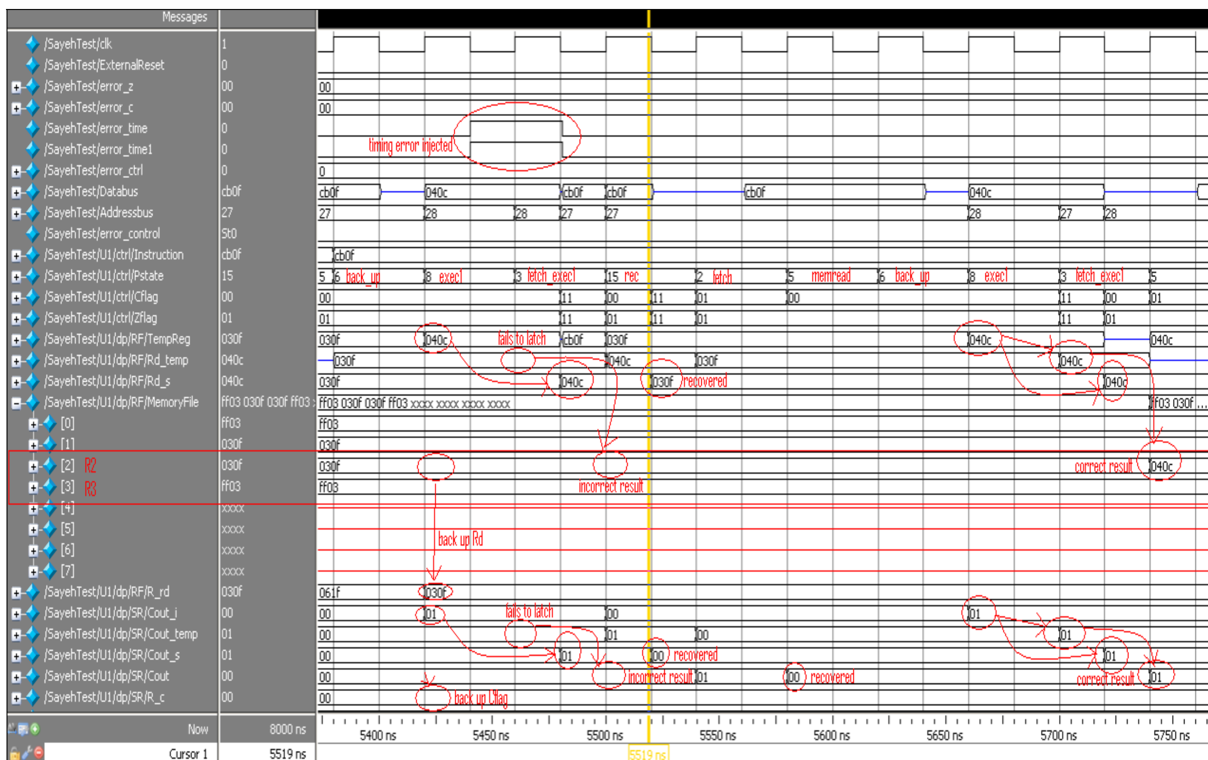


Fig. 5. Operation of sub with timing fault.

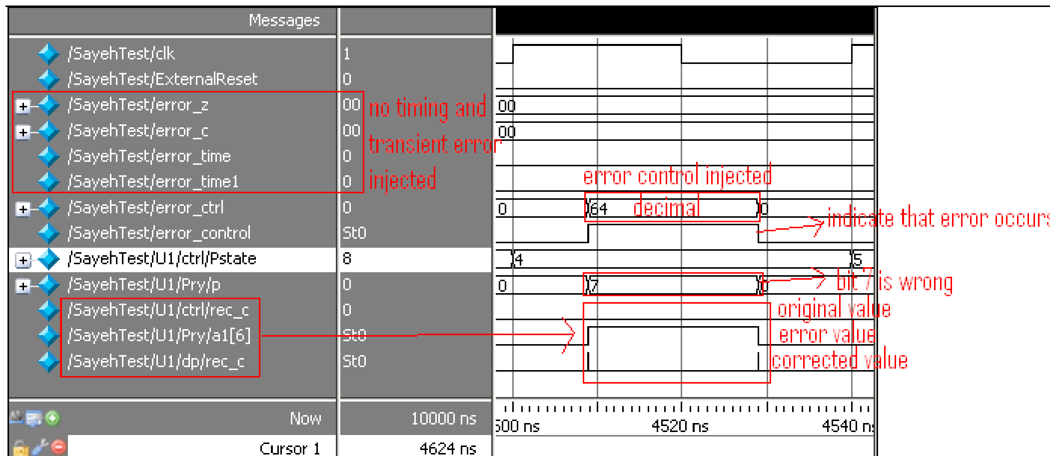


Fig. 6. System operation with faults in controller.

Five clock cycles later, at 5900ns, the correct value is latched and the system carries on.

Finally, Fig. 6 shows an error in the control signals. Because this is simply a case of parity correction, the error is easily detected and corrected.

The hardware overhead for the datapath – two shadow registers, two temp registers, two back-up registers, two comparators and a parity checker and corrector – is much less than required in conventional duplication (or triplication). There is also a small overhead caused by the dual-rail encoding of status flags. The modification to the controller also adds speed penalties to the system. It can be observed in the simulation waveforms that one extra clock cycle is needed for the back up state. Moreover, the modification means the main flip-flop catches the result one clock cycle later than normal, but this does not affect the system operation. The SAYEH processor also includes the capability for so-called *shadow* instructions – in other words for limited pipelining. We have implemented fault recovery mechanisms for these shadow instructions, which are as effective as those for the main instructions, but with a speed penalty of three clock cycles.

V. Conclusions

We have shown that limiting redundancy to the status flags of a datapath is sufficient to detect a number of transient and timing errors within that datapath, when combined with a totally self-checking controller. By including back up states, the system can repeat its operation in the presence of transient and timing faults. The hardware overhead is significantly less than for conventional duplication.

The technique is largely complementary to Razor. Thus, future work will investigate how voltage scaling and the Razor methodology might be combined with the proposed technique. The example used here is relatively simple; we will investigate the application of the technique to a processor with much greater pipelining.

References

[1] P. Oikonomakos and M. Zwolinski, "On the design of self-checking controllers with datapath interactions," *Computers, IEEE Transactions on*, vol. 55, no. 11, pp. 1423 –1434, nov. 2006.

[2] ——, "An integrated high-level on-line test synthesis tool," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 25, no. 11, pp. 2479 –2491, nov. 2006.

[3] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: A low-power pipeline based on circuit-level timing speculation," in *MICRO 36: Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*. Washington, DC, USA: IEEE Computer Society, 2003, p. 7.

[4] "SAYEH processor," www.opencores.org.

Approximate and Bottleneck High Performance Routing for Self-healing VLSI Circuits

Achira Pal[1], Tarak N. Mandal[2], Alak K. Datta[3], Rajat K. Pal[4], and Atal Chaudhuri[5]

[1]Harinavi Subhasini Balika Sikshalaya, PO: Harinavi, 24 Parganas (South) – 743 359, West Bengal, India
[2] PMI Service Center Europe Sp. z o.o., al. Jana Pawla II 196, 31-982 Krakow, Poland
[3]Department of Computer and System Sciences, Visva-Bharati, Santiniketan, Birbhum – 731 235, West Bengal, India
[4]Department of Information Technology, School of Technology, Assam University, Silchar, Cachar – 788 011, Assam, India
[5]Department of Computer Science and Engineering, Jadavpur University, Kolkata – 700 032, West Bengal, India

Abstract—**Crosstalk minimization is one of the most important aspects in interconnecting VLSI circuits. With the advancement of fabrication technology, devices and interconnecting wires are placed in closer proximity and circuits operate at higher frequencies. This results in crosstalk between wire segments. In this paper, we show that the crosstalk minimization problem in the reserved two-layer Manhattan routing model is NP-complete, even if channels are free from any vertical constraint. In addition, we introduce the problems of minimizing bottleneck crosstalk and approximating crosstalk minimization, and prove that these problems are also NP-complete. We further show that all these results hold even if doglegging is allowed.**

Keywords- channel routing; NP-hardness; crosstalk minimization; high performance routing; bottleneck crosstalk; approximation algorithm; doglegging; green computing.

I. INTRODUCTION

In VLSI layout design it is required to realize a specified interconnection of a set of terminals present in different modules, primarily using minimum possible area. This is known as the *routing problem*. There exist several routing strategies for efficient interconnection among different modules. One of the most important types of routing strategies is *channel routing* [9, 16].

A *channel* is a rectangular routing region that has two open ends, the left and right sides of the rectangle. The other two sides, viz., the upper and lower side of the rectangle have two rows of fixed points, called *terminals*. The terminals are aligned vertically in *columns* that are usually equispaced along the length of the channel. A set of terminals that need to be electrically connected together is called a *net*. In Figure 1, two columns having the same number (other than zero) uniquely define a *two-terminal net*.

A *vertical wire segment* is a wire that lies in a column whereas a *horizontal wire segment* is a wire that lies in a track. *Tracks* are horizontal lines that are usually equispaced along the height of the channel, parallel to the two rows of (fixed) terminals.

A *route* for a net is a collection of horizontal and vertical wire segments spread across different layers connecting all the terminals of the net. A *legal wiring* of a channel is a set of routes that satisfy all the prespecified conditions where, no two wire segments used to connect different nets overlap on the same conducting layer. A legal wiring is also called a *feasible routing solution*.

The *channel routing problem (CRP)* is specified by two *m* element vectors *TOP* and *BOTTOM*, and a number *t*; objective is to find a feasible routing solution for the channel using no more than *t* tracks, if it exists. An instance of the CRP is shown in Figure 1, where we have an assignment of intervals of the nets present in this channel to four tracks only. Let L_i (R_i) be the leftmost (rightmost) column position of net n_i, then $I_i = (L_i,R_i)$ is known as the *interval* (or *span*) of the net.

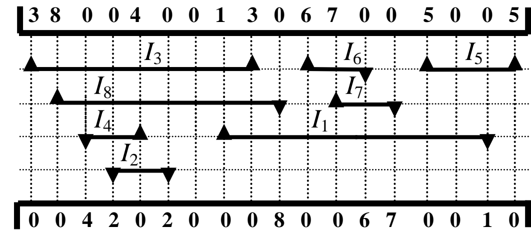


Figure 1: An example channel of eight nets. Intervals of the nets are placed in four different tracks. Terminals are vertically aligned along the columns of the channel. The length of the channel (i.e., the number of columns) is 18. Arrows indicate that the terminals to be connected, either on the top or at the bottom, to complete the required interconnection of all nets present in the channel.

As fabrication technology advances and feature size reduces, devices are placed in closer to each other and interconnecting wire segments are assigned with narrower pitch, whereas the circuits' operations are realized at higher frequencies. As a result, electrical hazards, viz., *crosstalk* between wire segments are evolved. Crosstalk between wire segments is proportional to the coupling capacitance, which is in turn proportional to the coupling length; the total length of overlap between wire segments of two different nets. Crosstalk is also proportional to the frequency of operation and inversely proportional to the separating distance between wires.

More crosstalk means more signal delay and reduced circuit performance. Therefore, it is desirable to develop channel routing algorithms that not only reduce the number of tracks (i.e., channel area) but also crosstalk. Work on routing channels with reduced crosstalk is very important from high performance requirement point of view [3, 9].

We define that the amount of crosstalk between the horizontal wire segments (i.e., intervals) of two different nets assigned to two adjacent tracks in a given routing solution is proportional to the amount of overlap of their horizontal spans. If two intervals do not overlap, there is no horizontal constraint between the nets. That is, if there is an overlap of the horizontal wire segments of a pair of nets, there is a possibility of having *accountable* crosstalk (other than negligible or no crosstalk) between them.

We measure crosstalk in terms of number of units of overlap between a pair of nets on adjacent tracks in a feasible routing solution. We assume that the crosstalk between wire segments of two different nets (that may or may not overlap) assigned to two nonadjacent tracks is negligibly small (in comparison to the amount of crosstalk between overlapping wire segments of two different nets on adjacent tracks), and hence can be ignored. In fact technology is indeed responsible to make the amount of this crosstalk within a permissible range of noise margin.

We assume that the amount of crosstalk between vertical wire segments of two different nets placed in two adjacent columns to be very small, and hence can be neglected. It is a matter of technology to

keep safe separation between two adjacent columns of a channel so that crosstalk evolved due to vertical wire segments is always within some limit of tolerance even if longest possible adjacency of vertical wire segments of two different nets, along the length of the channel.

II. Problems of Crosstalk Minimization in Channel Routing

In computing a routing solution, our prime intention is to compute a solution that uses minimum channel area. In addition to computing a routing solution with reduced area, in high performance routing our interest is also to obtain a routing solution with less electrical hazards (i.e., crosstalk), less signal propagation delay, less power consumption, less or no hot spot formation, and so and so forth.

The CRP of area minimization is polynomial time computable if the instances are free from any vertical constraint, and there are algorithms for computing exactly density, d_{max}-track routing solutions for such instances [6, 9]. Since the problem of minimizing area for instances of routing channel with only horizontal constraints is polynomial time solvable (using exactly d_{max} tracks), we define such instances as the *simple* instances of channel routing. (We define a channel specification as *general*, if both the constraints are present in it.) Though a routing solution of only d_{max} tracks is guaranteed for the simple instances of channel specifications (in the stated routing model), it may not be a *good* routing solution from the resulting crosstalk point of view (see Figure 1).

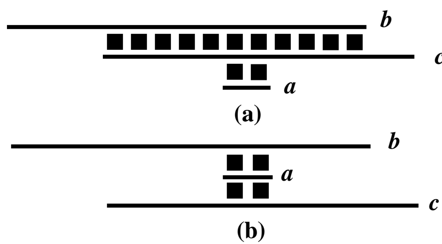


Figure 2: Crosstalk minimization problem in two-layer VH channel routing, in the absence of vertical constraints. **(a)** A feasible three-track routing solution with three intervals of three different nets *a*, *b*, and *c* that are overlapping to each other. Nets *b* and *c* share 11 units of horizontal span in the channel (as they are assigned to two adjacent tracks), and nets *c* and *a* share 2 units, amounting a total of 13 units' cross coupling length. **(b)** Another feasible three-track routing solution for the same channel instance, with a total net sharing of 4 units of horizontal span; hence a minimized crosstalk routing solution is obtained just by reassigning the nets to tracks.

Now let us consider a smaller simple channel instance of only three nets and illustrate the presence of crosstalk between nets (or intervals), when these are assigned to tracks in a two-layer VH channel routing (see Figure 2). Since all the three nets *a*, *b*, and *c* overlap, we are compelled to assign them to three different tracks on the same horizontal layer in any feasible routing solution. Interestingly, the fact to be noticed that just by reassigning the nets to tracks, the amount of crosstalk in Figure 2(b) is reduced to 30.77% to that of in Figure 2(a). Hence we have the following observation.

Observation 1: *The amount of crosstalk is mostly reduced if a net (or interval) of smaller span is sandwiched by two nets (or intervals) of larger spans.*

Now we pose the decision version of the problem *VHP*.

Problem: *VHP* (Crosstalk minimization in two-terminal no-dogleg two-layer VH channel routing, given a˜priori partition of nets.)

Instance: A simple channel specification of two-terminal nets, a partition *P* of nets into classes of non-overlapping intervals, and a positive integer *K*.

Question: Does there exist a minimum area (i.e., |*P*|-track) no-dogleg two-layer VH routing solution of the given channel specification so that (i) all the nets of the same class in the given partition *P* are assigned to the same track, and (ii) the total crosstalk is *K* or less?

In Section III, we prove that this problem is NP-complete. Now we formally pose the problems of crosstalk minimization, *VHS* for simple instances, and *VHG* for general instances of channel specifications in the absence of any partition *P* of nets, as follows.

Problem: *VHS* (Crosstalk minimization in two-terminal no-dogleg two-layer VH channel routing for simple instances of channel specification.)

Instance: A simple channel specification of two-terminal nets with density d_{max}, and a positive integer *K*.

Question: Does there exist a d_{max}-track no-dogleg two-layer VH routing solution of the given channel specification so that the total crosstalk is *K* or less?

In posing *VHG*, we consider general instances of channel specification, and here d_{max} is replaced by an integer *t* as the number of tracks required in computing a no-dogleg two-layer VH routing solution.

In Section IV, we prove that these problems are NP-complete. Now we pose the *bottleneck* crosstalk minimization problem as follows.

Problem: *BVHP* (Bottleneck crosstalk minimization in two-terminal no-dogleg two-layer VH channel routing, given a˜priori partition of nets.)

Instance: A simple channel specification with two-terminal nets, a partition *P* of nets into classes of non-overlapping intervals, and a positive integer *B*.

Question: Does there exist a minimum area (i.e., |*P*|-track) no-dogleg two-layer VH routing solution of the given channel specification so that (i) all the nets of the same class in the given partition *P* are assigned to the same track, and (ii) the amount of crosstalk between wire segments of two different nets on adjacent tracks is *B* or less?

In Section VI, we have proved that this problem is also NP-complete.

III. Hardness of Crosstalk Minimization in the Absence of Vertical Constraints

In this section we show that *VHP* is NP-complete by reducing a variant of the *Hamiltonian path (HP)* problem to *VHP*. The problem *HP* is the following [1, 4, 12].

Instance: An undirected graph $G = (V,E)$.

Question: Does *G* contain a Hamiltonian path?

Before showing that *VHP* is NP-complete, we need to show that the following variant *HP** of problem *HP* is also NP-complete.

Problem: *HP* (Weighted Hamiltonian path)*

Instance: An undirected weighted complete graph $G^* = (V,E^*)$, with weight $w(e) = 1$ or 2 for each edge $e \in E^*$.

Question: Does *G* contain a Hamiltonian path of weight $n-1$, where $n = |V|$?

We show that the problem HP^* is NP-hard by reducing the problem HP to it. We know that the problem HP is identified to be NP-complete [1, 4, 12]. We first show that HP^* belongs to NP. Given an instance of the problem, we use as a certificate the sequence of $n = |V|$ distinct vertices in the path. The verification algorithm checks that this sequence contains each vertex exactly once, sums up the edge weights, and checks whether the sum is exactly $n-1$. This algorithm can certainly be performed in polynomial time. Therefore, $HP^* \in$ NP.

To prove that HP^* is NP-hard, we show that $HP \leq_P HP^*$. Let $G = (V,E)$ be any instance of HP. We construct a corresponding instance of HP^* as follows. We compute a complete graph $G^* = (V,E^*)$, where $E^* = \{(v_i,v_j) \mid v_i, v_j \in V\}$, and we assign weight $w(e) = 1$, if $(v_i,v_j) \in E$; otherwise, we assign weight $w(e) = 2$, if $(v_i,v_j) \notin E$. The instance of HP^* is then obtained in polynomial time. The construction has been explained in Figure 3.

We now show that graph G has a Hamiltonian path if and only if graph G^* has a weighted Hamiltonian path HP_P^* of weight $n-1$, where $n = |V|$. Suppose that graph G has a Hamiltonian path HP_P. Each edge in HP_P belongs to E and thus HP_P^* has weight $n-1$ in G^*. Thus if HP_P is a Hamiltonian path in G, then HP_P^* is a weighted Hamiltonian path in G^* with weight $n-1$.

Conversely, suppose that graph G^* has a path HP_P^* of weight $n-1$. Since the weights of the edges in E^* are either 1 or 2, the weight of the path HP_P^* is $n-1$ indicates that the path HP_P^* contains only edges in E. We conclude that HP_P is a Hamiltonian path in graph G. This completes the proof.

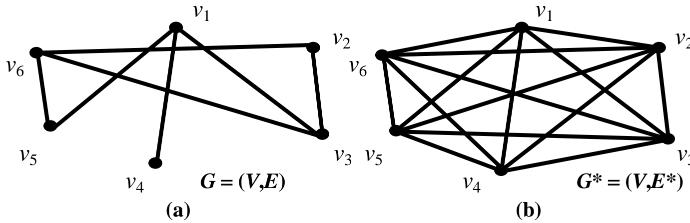


Figure 3: **(a)** A graph instance $G = (V,E)$ of problem HP. **(b)** The graph $G^* = (V,E^*)$ of the corresponding instance of problem HP^*, that is computed from G in (a), with weight $w(e) = 1$, if $e = \{(v_i,v_j) \in E \mid v_i, v_j \in V\}$; otherwise, $w(e) = 2$.

We summarize this result in the following theorem.

Theorem 1: *HP* is NP-complete.*

Now we show that VHP is NP-complete by reducing an instance I of problem HP^* to it. We show that $VHP \in$ NP. Given a feasible $|P|$-track no-dogleg two-layer VH routing solution for any instance I' of VHP, we can verify in polynomial time whether (i) all the nets of the same class in the given partition P are assigned to the same track, and (ii) the total crosstalk is K or less. Therefore, $VHP \in$ NP.

To show that VHP is NP-complete, we consider the following transformation from problem HP^* to VHP. We construct an instance (I',P,K) of problem VHP from any instance I of problem HP^* by a polynomial time transformation as follows.

Let the number of vertices of the graph in I be n. For every vertex $v_i \in V$, $1 \leq i \leq n$, we introduce $n-1$ two-terminal nets $n_{i1}, n_{i2}, \ldots, n_{i(n-1)}$; one for each edge (v_i,v_j), $j \neq i$, where v_j is a vertex adjacent to v_i in G^* (as G^* is a complete graph). We place the two terminals of net n_{ij} at positions $5((2n-i)(i-1)/2+(j-i-1))+1$ and $5((2n-i)(i-1)/2+(j-i-1))+5$ of the top row of terminals of the channel specification in I', if $i < j$. On the other hand, if $i > j$, we place the two terminals of net n_{ij} at positions $5((2n-j)(j-1)/2+(i-j-1))+2$ and $5((2n-j)(j-1)/2+(i-j-1))+2+w(v_i,v_j)$ of the bottom row of terminals of the channel specification in I'. We further consider a partition P of the set of nets produced into n ($= |P|$) subsets $\{P_1, P_2, \ldots, P_n\}$, where P_i is a class of nets n_{ij}, $1 \leq j \leq n$ such that $j \neq i$. All the remaining terminals of the constructed channel instance I' are vacant terminals (i.e., the terminals that are not to be connected). Hence, the construction of the channel instance I' is completed that contains $2e = n(n-1)$ nets and the length of the channel in I' is $5e$. The construction of such a channel instance in I' from a graph instance in I has been explained in Figure 4.

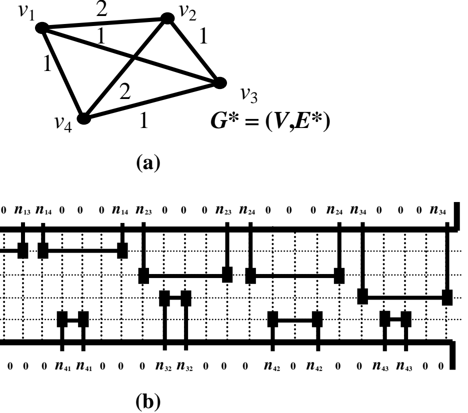


Figure 4: **(a)** A complete graph $G^* = (V,E^*)$ of instance I of problem HP^*, with weight 1 or 2 on the edges of the graph. **(b)** The corresponding channel instance I' of the crosstalk minimization problem VHP, where nets n_{ij}, $1 \leq i, j \leq n$ but $j \neq i$, are introduced into the channel corresponding to vertex v_i in G^*, and forming class[i] of non-overlapping nets in P. In addition, intervals I_{ij} and I_{ji}, of nets n_{ij} and n_{ji}, respectively, $1 \leq i, j \leq n$, $j \neq i$, of the constructed channel instance I' overlap of an amount 1 (2) unit(s), if $w(e) = 1$ (2), where $e = \{(v_i,v_j) \in E^* \mid v_i, v_j \in V\}$. Incidentally, for I', $n = |V| = |P|$.

Following the construction of I', we observe that the density of the channel is 2, as stated in the following lemma.

Lemma 1: *The channel density d_{max} of the constructed channel instance I' is 2.*

To complete the proof that VHP is NP-complete, we now establish the following lemma.

Lemma 2: *I has a weighted Hamiltonian path HP_P^* of weight $n-1$, if and only if I' has a feasible n-track no-dogleg two-layer VH routing solution with (i) all the nets in the same class in the given partition P, where $|P| = n$, are assigned to the same track, and (ii) the total crosstalk is $n-1$.*

Proof: Suppose that there is a weighted Hamiltonian path $HP_P^* = \langle v_1', v_2', \ldots, v_n' \rangle$ in I of weight $n-1$. We show that there is a feasible $|P|$-track no-dogleg two-layer VH routing solution S for I' with (i) all the nets of the same class in the given partition P are assigned to the same track, and (ii) the total crosstalk is $n-1$. Observe that all the $n-1$ nets n_{ij}', $1 \leq j \leq n$ such that $j \neq i$, corresponding to vertex v_i' in HP_P^* are in a single class. We assign all these nets to the ith track from the top of the channel. In addition, no two different sets of nets are assignable to the same track as at least two intervals belonging to two different classes of nets overlap each other. So, the only nets n_{ij}' corresponding to vertex v_i' in HP_P^* are assigned to the ith track of the channel.

Now, since the weight of the path HP_P* in I is $n-1$ and HP_P* is a path of exactly n vertices, weight of each edge in the path is 1. Corresponding to this path HP_P* of $G*$ in I, we obtain a no-dogleg two-layer VH routing solution S, as stated earlier, where in the topmost track we assign the class of nets n_{1j}'. For two successive vertices v_i' and v_{i+1}' in HP_P*, $1 \le i \le n-1$, we assign the corresponding classes of nets to the ith and $(i+1)$th track in S from top to bottom, where only two nets of these two classes amount to overlap only 1 unit of horizontal span (as $w(v_i',v_{i+1}') = 1$), following the construction of I'. So we have a $|P|$-track no-dogleg two-layer VH routing solution S in I' with (i) all the nets of the same class in the given partition P are assigned to the same track, and (ii) the total crosstalk is $n-1$.

Now suppose there is a $|P|$-track no-dogleg two-layer VH routing solution S in I' with (i) all the nets of the same class in the given partition P are assigned to the same track, and (ii) the total crosstalk is $n-1$. We show that there is a weighted Hamiltonian path HP_P* in I whose weight is $n-1$. According to the constructed channel instance I' and the solution S of n tracks (as $|P| = n$), note the following: (i) In each track i, $1 \le i \le n$, we have a set of $n-1$ non-overlapping intervals I_{ij}' corresponding to the nets n_{ij}', $1 \le j \le n$ such that $j \ne i$, in a class of P, and (ii) exactly one pair of nets belonging to any two classes of partition P of nets in I', overlap amounting at most of 2 units.

Observe that no two classes of nets assigned to two adjacent tracks in S overlap 2 units of horizontal span; otherwise, the amount of total crosstalk is more than $n-1$. In other words, the amount of total crosstalk is $n-1$ indicates that each pair of classes of nets assigned to two adjacent tracks in S overlap 1 unit of horizontal span only. Therefore, we construct a sequence of vertices for the path HP_P* in I, where v_i' is at the ith position, $1 \le i \le n$ $(= |V|)$, if the non-overlapping intervals I_{ij}' corresponding to the nets n_{ij}', $1 \le j \le n$ such that $j \ne i$, in a class of P are assigned to the ith track from top in S. So, for sets of nets n_{ij}' and $n_{(i+1)j}'$ assigned to two adjacent tracks i and $i+1$, respectively, we have two successive vertices v_i' and v_{i+1}' in HP_P*, $1 \le i \le n-1$, so that $w(v_i',v_{i+1}') = 1$ in $G*$. Hence we have a weighted Hamiltonian path HP_P* in I whose weight is $n-1$. ♦

We summarize the result in the following theorem.

Theorem 2: *VHP is NP-complete.*

The result of crosstalk minimization problem with partition of nets established in this section equally holds for the general instances of channel specifications (where partition is provided in such a way that no cyclic vertical constraint is formed). This is because the set of simple instances of channel specifications is a proper subset of the set of general instances of channel specifications considering both the constraints present in it.

IV. Hardness of Other Crosstalk Minimization Problems

In this section, we consider the problems *VHS* and *VHG*, separately, as defined in Section II, of crosstalk minimization in two-layer channel routing of minimum area, without any restriction on partition of nets to tracks. In other words, we want to compute a minimum area no-dogleg two-layer VH routing solution, without any partition of nets (so that the restriction on a set of nets to be assigned to the same track is no longer stated), in which crosstalk is also minimized. In that case a natural question is: *What is the minimum amount of sum crosstalk that a fixed area no-dogleg two-layer VH routing solution may have?*

Let us first consider the problem *VHS*. There are polynomial time algorithms for computing d_{max}-track routing solutions and these are the minimum area routing solutions for the simple instances of channel specifications [6, 9, 10], as stated above. So, if we could compute all possible d_{max}-track routing solutions of such an instance, we may compute a minimum crosstalk routing solution among them. This may take exponential amount of time.

Nevertheless, for any such combination of d_{max} sets of non-overlapping intervals, we have the nets in the form of classes of a partition of nets. We may assume that the size of this partition P is same as d_{max} in order to compute a minimum area feasible routing solution for the given channel instance. Now we could reassign the intervals trackwise, so that this reassignment of tracks may provide a d_{max}-track minimum crosstalk routing solution. This requirement for an instance with partition P, where $|P| = d_{max}$ of crosstalk minimization problem *VHS* and the objective for a similar instance of crosstalk minimization problem *VHP* are exactly identical. Hence we can conclude that these two problems are having the same computational complexity. We formally state the result in the following theorem.

Theorem 3: *VHS is NP-complete.*

Now let us consider the problem *VHG*. In this problem, we want to compute a no-dogleg two-layer VH routing solution of total crosstalk K or less, if there exists such a solution of t tracks for the given (general) channel specification. Since the set of simple instances of channel specifications is completely contained in the domain of general instances of channel specifications, so in order to prove *VHG* NP-complete, we restrict the instances of this problem by the instances of *VHS*. In addition, we set $t = d_{max}$. Hence, as a result we conclude the following.

Theorem 4: *VHG is NP-complete.*

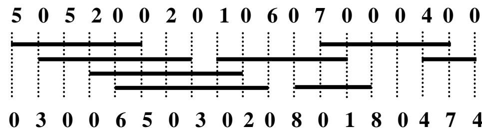
V. Hardness of Approximating Crosstalk Minimization

In previous two sections, we have shown that the problem of two-layer crosstalk minimization is NP-complete for simple as well as general instances of channel specifications. A natural question arises: *Whether there is any polynomial time approximation algorithm with guaranteed error bound for these problems.* In this section, we prove that the problem of developing such approximation scheme is impossible.

In fact, we will show that it is impossible to design an approximation algorithm with ratio error bound $2-\varepsilon$, unless P = NP, even for the simple instances of the problem when a partition is given, so that the nets in a class of the partition are to be assigned to the same track.

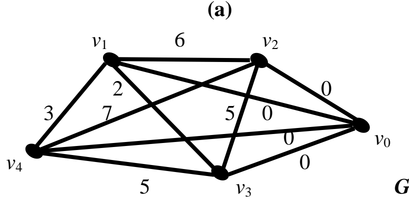
To establish this result we formulate crosstalk minimization problem with partition as a general *traveling salesman problem (TSP)*. The formulation is based on constructing a weighted undirected complete graph G, as described below.

For every set i of nets (i.e., the nets in class[i]), $1 \le i \le t$, of the crosstalk minimization problem with partition P, where $|P| = t$, we introduce a vertex v_i into graph G. For a pair of tracks we introduce an edge between the corresponding vertices in the graph with weight same as the total amount of overlapping between the nets in these two tracks. So, we assign weight w to the edge joining vertex pair v_p and v_q, if and only if the total amount of overlapping of nets belonging to tracks p and q (obeying classes in P) is w. We now introduce one more vertex v_0 and t edges $\{v_0,v_i\}$ into the graph, where $1 \le i \le t$. So, now G is a complete graph of order $t+1$. For each of these newly introduced edges to v_0, we assign weight zero. The construction has been illustrated in Figure 5.



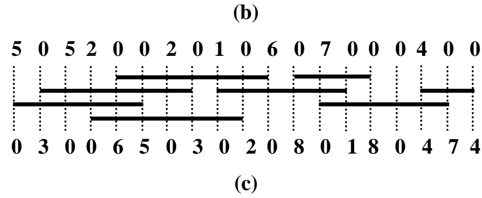
Partition $P = \{C_1, C_2, C_3, C_4\}$,
where $C_1 = \{5, 7\}$, $C_2 = \{3, 1, 4\}$, $C_3 = \{2\}$, and $C_4 = \{6, 8\}$.

(a)



A tour $T = \langle v_0, v_4, v_2, v_1, v_3, v_0\rangle$

(b)



(c)

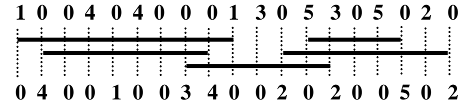
Figure 5: **(a)** A channel specification, and a partition P with four classes of nets C_1 through C_4 so that the nets in a class are non-overlapping to each other and to be assigned to the same track. Note that this is only an input to the problem, not a routing solution. **(b)** The corresponding constructed graph instance G for the *TSP* problem, where each of the weights assigned to edges is same as the amount of overlapping between the nets belong to the associated classes in P. A tour T for the *TSP* problem is assumed as $\langle v_0, v_4, v_2, v_1, v_3, v_0\rangle$. Cost of the tour is 15 units. Cost of the path $\langle v_4, v_2, v_1, v_3\rangle$ is also 15 units, as the weights of the edges incident to v_0 are all zero. **(c)** The assignment of nets, following the path of the tour (ignoring v_0), from top to bottom along the height of the channel, as the nets are there in different classes in P. This assignment of nets results a routing solution with exactly 15 units of total crosstalk.

In this example, we have considered a channel instance and a partition P of nets, as shown in Figure 5(a). The corresponding graph G for the *TSP* problem is shown in Figure 5(b). A tour T is also assumed there, and the cost of the tour $c(T)$ is 15 units. Following the tour (ignoring v_0) of the *TSP* problem, we assign the nets in different classes in P of the channel instance to tracks along the height of the channel, as shown in Figure 5(c). The total amount of crosstalk of this assignment of nets as a routing solution is 15 units, which is same as the cost of the tour.

Now it is interesting to note the following lemma.

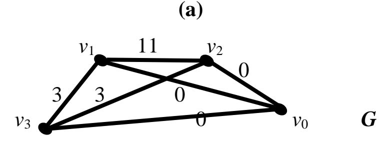
Lemma 3: *For every tour, starting with v_0, of the traveling salesman problem there is a unique $|P|$-track no-dogleg two-layer VH routing solution, and the amount of crosstalk in this routing solution is same as the cost of the tour.*

Proof: Let $T = \langle v_0, v_1', v_2', \ldots, v_t', v_0\rangle$ be a tour for an instance of the *traveling salesman problem (TSP)* obtained with cost $c(T)$. From tour T, if we delete vertex v_0 and its adjacent edges then the cost of the path from v_1' through v_t' remains same as $c(T)$. This is because, from the construction of graph G for a *TSP* instance, the weights of the edges (v_0,v_1') and (v_t',v_0) are zero. Accordingly, we could assign the sets of nets from top to bottom along the height of the channel, obeying t classes of partition P, so that the nets in a class are to be assigned to the same track. In this assignment, nets corresponding to v_i' are assigned to track i, $1 \le i \le t$, from the top of the channel. So, the nets corresponding to v_t' are assigned to the bottommost track, where $t = |P|$. Hence, a $|P|$-track no-dogleg two-layer VH routing solution is obtained.



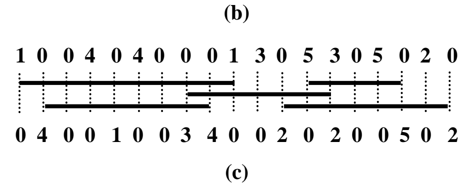
Partition $P = \{C_1, C_2, C_3\}$,
where $C_1 = \{1, 5\}$, $C_2 = \{4, 2\}$, and $C_3 = \{3\}$.

(a)



A tour $T = \langle v_0, v_2, v_3, v_1, v_0\rangle$

(b)



(c)

Figure 6: **(a)** A channel specification, and a partition P with three classes of nets C_1, C_2, C_3, similar to the channel instance in Figure 5(a). **(b)** The associated graph instance G for the general *TSP* problem, where the triangle inequality is not maintained. This is because the cost of edge (v_2,v_3) (i.e., 3 units) plus the cost of edge (v_3,v_1) (i.e., 3 units) is less than the cost of edge (v_1,v_2) (i.e., 11 units). So, if a tour T for the general *TSP* problem is assumed as $\langle v_0, v_2, v_3, v_1, v_0\rangle$, then the cost of the tour becomes 6 units only, which is same as the cost of the path $\langle v_2, v_3, v_1\rangle$. **(c)** In fact, there is an assignment of nets, following the path of the tour (ignoring v_0), from top to bottom of the channel, maintaining the classes of nets in P, and this assignment results a routing solution with exactly 6 units of total crosstalk.

Now we prove that whether the amount of crosstalk of this routing solution is same as $c(T)$. Here we have just stated how the nets are assigned to tracks following their classes in P, along the height of the channel. So, for two consecutive vertices v_i' and v_{i+1}' in T (ignoring v_0), $1 \le i \le t-1$, the corresponding sets of nets are assigned to two successive tracks i and i+1, respectively, from the top of the channel. According to the construction of the graph, the weight of edge (v_i',v_{i+1}') is same as the total amount of overlapping between the nets in the corresponding classes in P. Hence, the sum crosstalk of the routing solution is same as the cost of the path consisting vertices v_1' through v_t', which is same as $c(T)$, as the weights of the edges (v_0,v_1') and (v_t',v_0) are zero. ♦

It turns out from the above lemma that the crosstalk minimization problem can be formulated as a general *TSP* problem. Observe that an instance of the *TSP* problem thus obtained may not satisfy the triangle inequality, as explained in Figure 6. Also we know that there is no approximation algorithm for the general *TSP* problem, with ratio error $2-\varepsilon$, unless P = NP [1, 4, 12]. We, therefore, claim the following theorem.

Theorem 5: *Unless P = NP, it is impossible to design an approximation algorithm for no-dogleg two-layer VH channel routing problem of crosstalk minimization with partition of nets for simple instances of channel specifications, with ratio error $2-\varepsilon$.*

As the problem of crosstalk minimization with partition in two-layer channel routing for simple instances of channel specifications is a

special case of the general two-layer channel routing problem of crosstalk minimization, we claim the following theorem.

Theorem 6: *Unless P = NP, it is impossible to design an approximation algorithm for no-dogleg two-layer VH channel routing problem of crosstalk minimization without any partition of nets for general instances of channel specifications, with ratio error* $2 - \varepsilon$.

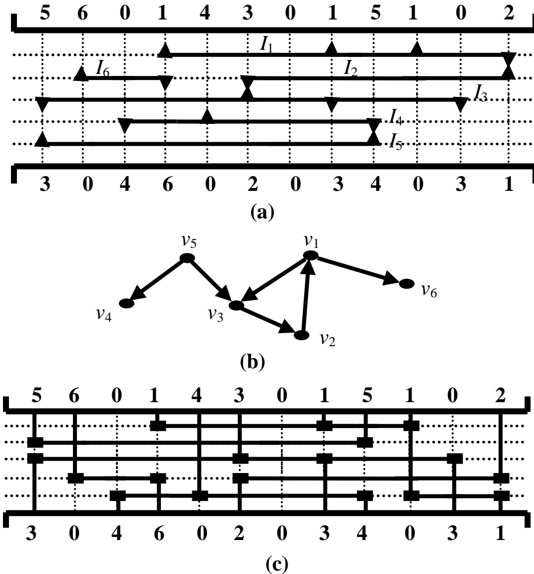


Figure 7: **(a)** A channel instance. **(b)** The VCG of the channel instance that contains a cycle. **(c)** A restricted dogleg routing solution for the channel instance in (a), where net 1 is doglegged and its horizontal subsegments are assigned to the first track and the fifth track of the channel, from top to bottom. Via holes are also shown in this figure, where two orthogonal wire segments of the same net intersect; these are used for changing layers of interconnect.

VI. Hardness of Bottleneck Crosstalk Minimization

Bottleneck crosstalk minimization is especially important when we are interested in computing a routing solution with every (individual) crosstalk between a pair of nets, due to overlapping of nets assigned to two adjacent tracks. This is interesting as a problem of combinatorial optimization, and equally important from high performance routing requirement point of view. The sum crosstalk minimization problem is important but lacks this point. As a result, the performance of an overall optimized crosstalk routing solution may not be *good* enough in propagating all the necessary signals in synchronization.

Now we prove that the bottleneck crosstalk minimization problem *BVHP* is NP-complete, following the proof of the crosstalk minimization problem *VHP*. We show that the problem *BVHP* is NP-complete for channels with routing having as low as an upper bound of two units of overlapping of the nets on adjacent tracks. This is straightway obtained by following the construction and the proof technique used to establish the problem *VHP* as NP-complete. In that construction, the net groupings among the classes of the partition have entirely defined. The amount of overlapping between a pair of nets, if they in fact overlap, in two different classes of nets is either 1 unit or 2 units only. So, we fix the value of $B = 2$, and the result is summarized in the following theorem.

Theorem 7: *BVHP is NP-complete.*

Now it is needless to mention that the bottleneck crosstalk minimization problem is NP-complete with and without any partition of nets (whether the sets of non-overlapping intervals are to be defined in classes of the partition for their assignment to tracks), for both simple as well as general instances of channel specifications.

So far in this paper, we have considered several crosstalk minimization problems in computing a routing solution for two-layer VH channel routing, using the shape of no-dogleg routes only. No-dogleg routing is simple but each instance of the CRP may not have a feasible routing solution using only the shapes of no-dogleg routes. As for example, consider the channel instance in Figure 7(a). The VCG of this channel contains a cycle (see Figure 7(b)). Hence, we consider channel routing with restricted doglegging [2, 9] for the instances with multi-terminal nets (see Figure 7(c)), and extend the NP-completeness results proved so far to this model by considering no-dogleg routing of instances restricted to have only two-terminal nets.

VII. Conclusion

In this paper we have considered the crosstalk minimization problem in two-terminal no-dogleg two-layer VH channel routing, and proved that the problem is NP-complete for simple as well as general instances of channel specifications. In addition, we consider the problem of existence of polynomial time approximation scheme to solve the stated CRP and proved that it is unattainable to design such an approximation scheme. The bottleneck crosstalk minimization problem has also been considered, and proved its NP-completeness. In addition, all these problems are proved NP-hard, even if restricted doglegging is allowed.

References

[1] Cormen T. H., C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*, Prentice-Hall of India Pvt. Ltd., New Delhi, 2001.

[2] Deutsch D. N., A Dogleg Channel Router, *Proc. of 13th ACM/IEEE Design Automation Conf.*, pp. 425-433, 1976.

[3] Gao T. and C. L. Liu, Minimum Crosstalk Channel Routing, *Proc. of IEEE Int. Conf. on Computer-Aided Design*, pp. 692-696, 1993.

[4] Garey M. R. and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, New York, 1979.

[5] Golumbic M. C., Algorithmic Graph Theory and Perfect Graphs, Academic Press, New York, 1980.

[6] Hashimoto A. and J. Stevens, Wire Routing by Optimizing Channel Assignment within Large Apertures, *Proc. of 8th ACM Design Automation Workshop*, pp. 155-169, 1971.

[7] Ho T.-T., S. S. Iyengar and S.-Q. Zheng, A General Greedy Channel Routing Algorithm, *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 10, pp. 204-211, 1991.

[8] LaPaugh A. S., *Algorithms for Integrated Circuit Layout: An Analytic Approach*, Ph.D. thesis, Lab. for Computer Sc., MIT, Cambridge, 1980.

[9] Pal R. K., *Multi-Layer Channel Routing: Complexity and Algorithms*, Narosa Publishing House, New Delhi (Also published from CRC Press, Boca Raton, USA and Alpha Science International Ltd., UK), 2000.

[10] Pal R. K., A. K. Datta, S. P. Pal and A. Pal, Resolving Horizontal Constraints and Minimizing Net Wire Length for Multi-Layer Channel Routing, *Proc. of IEEE Region 10's 8th Annual Int. Conf. on Computer, Communication, Control and Engineering*, vol. 1, pp. 569-573, 1993.

[11] Pal R. K., A. K. Datta, S. P. Pal, M. M. Das and A. Pal, A General Graph Theoretic Framework for Multi-Layer Channel Routing, Proc. of Eighth VSI/IEEE International Conference on VLSI Design, New Delhi, India, pp. 202-207, Jan. 4-7, 1995.

[12] Papadimitriou C. H., *Computational Complexity*, Addison-Wesley Publishing Co., Reading, Massachusetts, 1995.

[13] Rivest R. L. and C. M. Fiduccia, A 'Greedy' Channel Router, *Proc. of 19th ACM/IEEE Design Automation Conf.*, pp. 418-424, 1982.

[14] Schaper G. A., *Multi-Layer Channel Routing*, Ph.D. thesis, Dept. of Computer Sc., Univ. of Central Florida, Orlando, 1989.

[15] Szymanski T. G., Dogleg Channel Routing is NP-Complete, *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 4, pp. 31-41, 1985.

[16] Yoshimura T. and E. S. Kuh, Efficient Algorithms for Channel Routing, *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 1, pp. 25-35, 1982.

A Scalable Heuristic for Incremental High-Level Synthesis and Its Application to Reliable Computing

Shohei Ono[1] Hiroaki Yoshida[2,3] Masahiro Fujita[2,3]

1. Dept. of Electrical Engg. & Information Systems, the University of Tokyo
2. VLSI Design and Education Center (VDEC), the University of Tokyo
3. CREST, Japan Science and Technology Agency

Abstract

Due to extremely high non-recurring-engineering costs in ASIC development, incremental synthesis techniques have been becoming increasingly important. Although incremental logic synthesis, placement and routing tools have been available for a while, incremental high-level synthesis is still a challenging problem. We also show that incremental synthesis plays an important role in Concurrent Error Detection & Diagnosis (CEDD) technique, which is known as an effective technique for reliable computing. In this paper, we propose a practical incremental high-level synthesis method and present the details of a tool implementation and a preliminary result.

1 Introduction

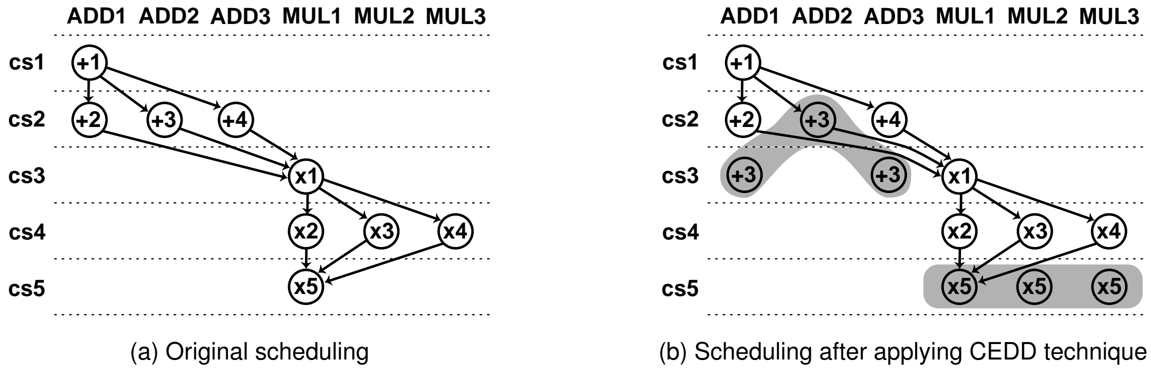
With the increase of VLSI complexity, high-level synthesis which can improve the design productivity plays an important role. High-level synthesis is a technology which converts a design described in a high-level language such as C language to a Register Transfer Level (RTL) description. Since the design and verification are performed in a higher level of abstraction using this technology, the time required for the design and verification can be shorten from the conventional design methodology based on RTL descriptions. Varieties of techniques for more reliable designs, including various checker insertion, duplicated computation and others, can be relatively easily incorporated in high level designs.

High-level synthesis consists of three phases: allocation, scheduling and binding. In the allocation phase, the number and type of functional units are determined. The scheduling phase determines when and which functional unit executes each operation. The result of the scheduling phase may vary dramatically depending on the design constraints provided by a designer. The binding phase determines the functional units and the registers used by each operation.

After generating the RTL description, logic synthesis generates a netlist from the RTL description. A netlist describes logic circuits and the interconnections of them. Then, the mask layout is generated by placement and routing. Generally, these processes take a long time. However, sometimes a design change is required late in these processes due to design errors and/or specification changes. After such a design change, logic synthesis, placement and routing need to be performed again even though these processes take a long time. To reduce the re-spin time, incremental logic synthesis and incremental placement and routing which update only the changed part are typically used. When the RTL description is changed, incremental logic synthesis and incremental placement and routing can be used, and these processes take shorter time than a general non-incremental synthesis. However, when the design at the higher level is changed, a conventional high-level synthesis results in too large changes on the RTL description. It takes a long time although we use incremental logic synthesis, placement and routing. By changing the RTL description manually to satisfy the design change at the higher level, the RTL description which is changed little can be made, but such change of the design on the RTL description needs a functional verification on the RTL description which also takes a long time.

Changes of designs and specifications may happen due to the requirement of reliable computing. For example, some duplicated computation turns to be too costly in late design stages such as a logic and layout designs. Then, other reliable computing method must replace those computations.

The increase of the design cost due to such design changes has led to a high requirement for incremental high-level synthesis which can generates the RTL description very close to the pre-change RTL description after the design change at the higher level. When there are small changes on the high-level description, the incremental high-level synthesis results in small changes on the RTL descrip-



(a) Original scheduling (b) Scheduling after applying CEDD technique

Figure 1. An example of Concurrent Error Detection & Diagnosis (CEDD) technique.

tion and reduce the process time because incremental logic synthesis, incremental placement and routing can be performed effectively.

Although an incremental high-level synthesis is known as a key technology to reducing the time of designing a VLSI, very little work has been performed except a recent work [1]. This paper proposes a scalable heuristic for incremental high-level synthesis. We also present the details of a tool implementation and a preliminary result.

Incremental high-level synthesis technique also plays an essential role in Concurrent Error Detection & Diagnosis (CEDD) technique. This technique is one of the widely-used techniques for reliability enhancement. In particular, a register-transfer-level CEDD technique can be realized as a high-level synthesis of a duplicated system [2]. We explain the CEDD technique using an example shown in Figure 1. Assuming that three adders and three multipliers are avaiable, the fastest schedule is given in Figure 1 (a). For example, operation +1 is performed on the adder ADD1 while operation x3 is performed on the multipler MUL2. It is clear that all functional units such as adders and multiplers are not being used in every clock cycle. A basic idea of the CEDD technique is to utilize such *spare computation cycles* to perform error detection and diagnosis. For example, suppose that we replicate operation +3 and execute each of them on a distinct adder, as illustrated by a shaded area in Figure 1 (b). Three replicated operations are executed on ADD2 in control step cs2 and ADD1 and ADD3 in control step cs3. By comparing the outputs of the adders pairwise, a faulty function unit can be identified. If (ADD1, ADD2) and (ADD2, ADD3) disagree, ADD2 can be identified as a fault adder. Thus, error diagnosis can be performed using spare computation cycles. The latency of diagnosis can be reduced if there is enough spare capacity. For example, three replicated operations x5 in 1 (b) are executed in the same cycle without increasing the latency.

A state-of-the-art RT-level CEDD technique employs an iterative improvement method [3]. Their ILP formulation uses a simplified model of hardware costs such as area, delay and power. Such a model-based optimization suffers from an inaccuracy of estimated costs. Incremental synthesis technique may improve the CEDD synthesis quality because accurate hardware costs can be obtained by using the technique.

2 Motivating Example

Figure 2 compares the results of a conventional high-level synthesis and an incremental high-level synthesis. The upper part of the figure shows the pre-change design and the result generated by high-level synthesis. The bottom part of the figure shows the changed design and the result generated by conventional high-level synthesis and incremental high-level synthesis.

The symbol $=$ in the designs means an assignment. There are four additions in the pre-change design. This means that these additions are calculated in the order. The result of scheduling shows which operation is calculated on which FU at which step. An arrow in the figure shows the dependency of the data used at the operation. In the pre-changed example, $h + i$ is calculated on the adder 1 at the step 1, and $j + k$ is calculated on the adder 2 at the step 1. The datapath shows the interconnection between the registers and the functional units. The operations in the design are calculated on this datapath with the control signal based on the result of scheduling. The control signal controls the multiplexers and the FUs in the datapath.

For the changed design, there are the results which are generated by conventional high-level synthesis and incremental high-level synthesis. $a = f + g$ is inserted before $b = h + i$, there are five additions in the changed design. As a result, the result of scheduling and the datapath are changed from the result before change. In the result of scheduling by conventional high-level synthesis, $a = f + g$ is bound to the adder 1 at the step 1 although $b = h + i$ was bound to this position in the result before change. This is because conventional high-level synthesis does not focus on the difference between the original result and the

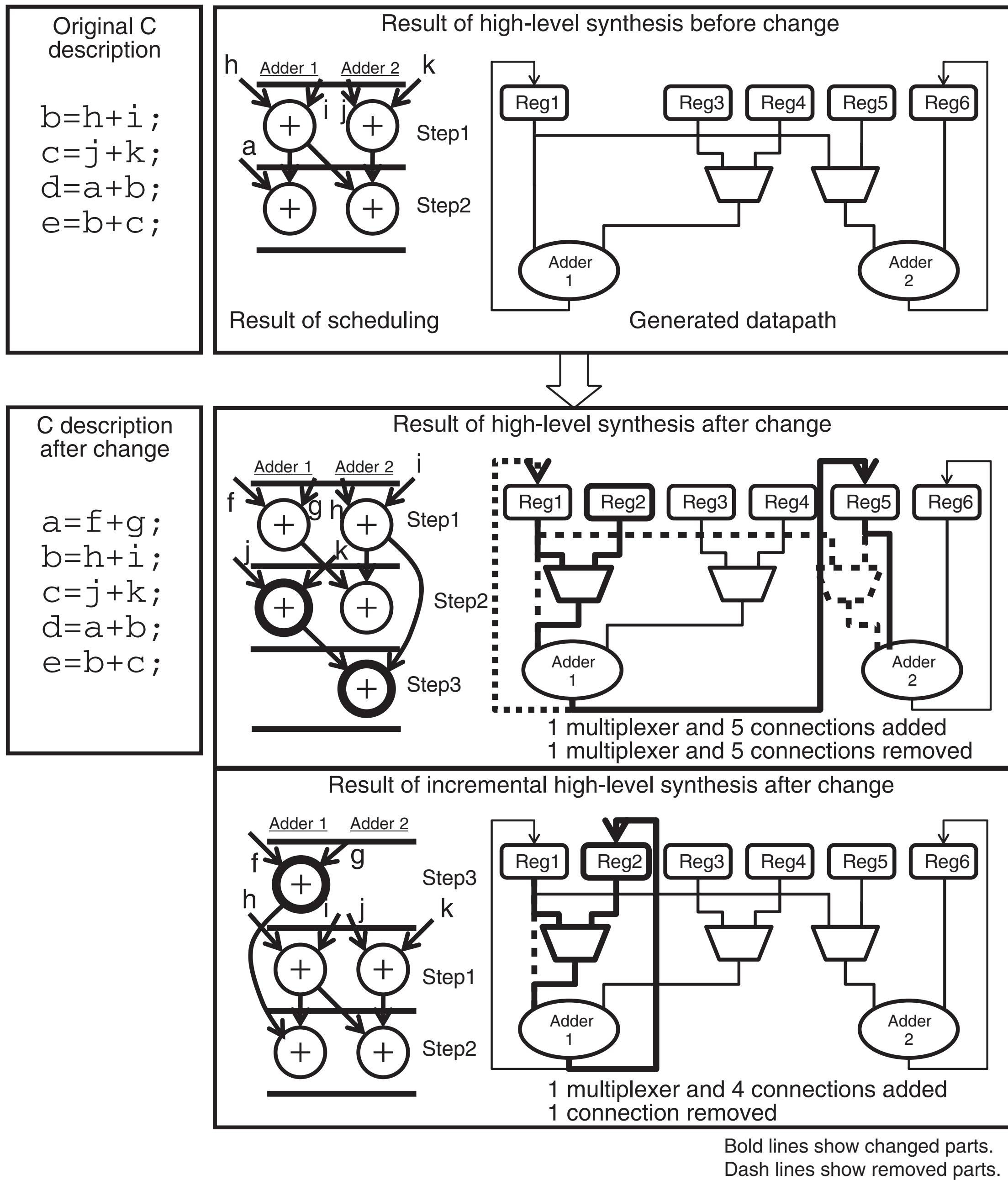


Figure 2. An example of incremental high-level synthesis.

result after change. And, the adder and the step which the other four additions are bound are also changed. The nodes which are shown by the bold line require the change of hardware. Then, the datapath generated by conventional high-level synthesis is changed. In the datapath, the five interconnections and the one multiplexer are added and the five interconnections and the one multiplexer are removed. In the incremental high-level synthesis, $a = f + g$ is bound to the step 3 which is newly inserted and the other four operations are bound to the same position with the pre-changed one. In the result by incremental high-level synthesis, only the node which is added by the design change require the change of hardware. Thus, the datapath has only smaller changes, the four interconnections and the one multiplexer are added and the one interconnection is removed. Like this, incremental high-level synthesis make only small changes on the RTL description. Hence, by using incremental high-level synthesis, incremental logic synthesis and incremental placement and routing can perform effectively and can reduce the processing time.

3 Proposed Method

In this section, we explain the proposed scalable heuristic for incremental high-level synthesis.

3.1 Problem Formulation

The input design is represented in a control data flow graph (CDFG) form. A CDFG is a combination of a control flow graph (CFG) and a data flow graph (DFG). A CFG is a directed graph which describes a control flow, and a DFG is a directed graph which describes dependence relationships of data used by operations. In this paper, we focus only on changes in a DFG to simplify the problem.

Given the CDFG of the original design and the changed design and the result of high-level synthesis for the original design, the steps and the FUs and the registers for each of the operations which are changed in the design are determined. In other words, the result of scheduling and binding of the changed design is generated. The behavior of this result must meet the behavior described in the changed design. In addition, it must satisfy the constraint condition provided by a designer and makes as small difference between the result of the original design and the changed design as possible. The goal is to find the datapath and control signals which is based on the scheduling and binding result described above.

3.2 Target Architecture

A template of the target architecture is shown in figure 3. It consists of functional units, a controller, a local store and registers. A functional unit (FU) is a computing unit which executes operations. Typical FU types are an ALU, a shifter and a multiplier. The input of an FU is connected to the output of the other FU or the register and the output of an FU is connected to the input of the other FU or the register. The FU has multiplexers on its input, and it will be controlled by a controller. A controller (CTRL) send control signals to all multiplexers and FUs based on the present state to execute arbitray operations. The controller has an input which determines the next state. A register (REG) is a memory to save local variables. The data which is calculated by FUs is saved to the registers on every step. A local store (LS) is a memory to save global variables. It is connected to outside, and it can share the contents of this memory with external hardware components. The number and type of these FUs, registers are variable.

3.3 Incremental High-Level Synthesis

Based on the synthesis result of the original design, the datapath and the control circuit corresponding to the changed part is generated incrementally. The proposed method consists of the allocation phase, the scheduling phase and the binding phase. At first, it enters to the allocation phase, then it enters alternately to the scheduling phase and the binding phase.

```
schedule_and_bind(dfg) {
  unbind(removed_nodes(dfg))
  dfg = SMS_sort(dfg)
  // Process in the determined order.
  for node in added_nodes(dfg) {
    schedule_and_bind_node(node)
  }
}

schedule_and_bind_node(node) {
  steps = available_step(node)
  for s in steps {
    success = FU_bind(node, s)
    if (success) break
  }
  if (!success) {
    s = insert_new_step()
    FU_bind(node, s)
  }
}
```

Figure 4. A pseudo code of scheduling procedure.

In the allocation phase, the number and type of the FUs are determined. These are same with the one which is used in the pre-change result.

In the scheduling phase, which step to execute the operations which is added by the design change is determined based on the synthesis result of the original design. The pseudo code of the scheduling phase is shown in figure 4. In this phase, we utilize the node ordering algorithm of the software pipelining approach which generates schedules that are near optimal in terms of an initiation interval, a register count and a stage count, called Swing Modulo Scheduling (SMS) [5]. First of all, the binding of the node which is removed by the design change is cancelled. Next, the list of available step for the nodes added by the design change is calculated in the order which is determined by the SMS. Binding the nodes to the step in the list is tried in the order. If the binding fails, a new step is inserted and the step is bound to it.

In binding phase, the nodes are bound to the FUs. The pseudo code of the binding phase is shown in figure 5. First, the FUs which is not bound at the target step and capable of the target operation are found. Next, the FU which has the least amount of hardware changes when the target operation is bound to it is calculated. Then, the FU is bound to the target operation, the inputs and the outputs of the FU are

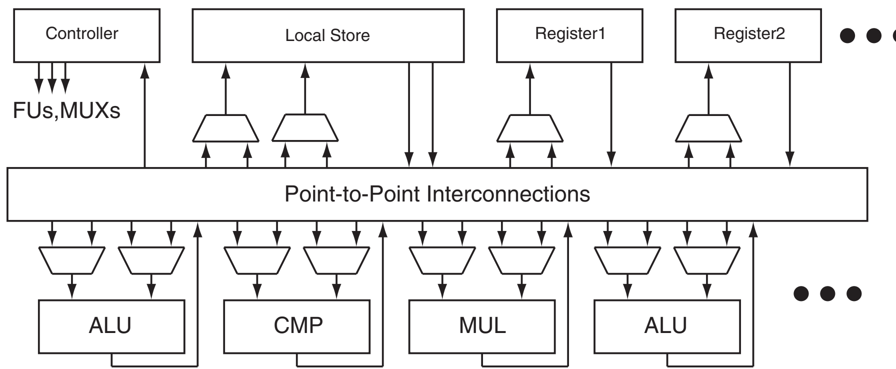


Figure 3. Our target architecture template.

```
FU_bind(operation, step) {
  FUs = get_available_FU(operation, step)
  FU = minimum_cost(FUs)
  bind(FU, operation, step)
  for input in operation.inputs {
    connect_register(input)
  }

  for output in operation.outputs {
    connect_register(output)
  }
  return true
}
```

Figure 5. A pseudo code of binding procedure.

```
/* This global variable is stored in a
 * local store and used to communicate
 * with the external components.
 */
int a[16];

void fibonacci(void) {
  int i;
  for (i=2; i<16; i++) {
    a[i] = a[i-1] + a[i-2];
  }
}
```

Figure 6. The design used to test the implementation

connected to the register. If there are no available FUs, the binding for the target operation fails.

After the scheduling phase and the binding phase, the registers which the variables use is determined. We use an optimal register assignment algorithm [4] which guarantees its optimality if the underlying form is an static single assignment (SSA) form to this procedure.

Then, the datapath and the control signals are generated as a register transfer level description.

The example of incremental high-level synthesis which is shown in figure 2 has the same result with the proposed method. In this example, $a = f + g$ is added and no operation is removed. The proposed method performs the procedure to only $a = f + g$. It calculates the available step and tries the FU binding procedure to the step 1 and the step 2, however, the FU binding procedure fails because there are no available FUs. Then, the step 3 is inserted before the step 1 and $a = f + g$ is bound to the adder 1 at the step 3.

4 Implementation and A Preliminary Result

We have implemented the proposed method partially in the SORA synthesis framework which we have developed. Our implementation can interpret the input design description written in the C language and build a CDFG in SSA form which is representing the input design description. We used the LLVM compiler infrastructure [6] which is a framework of compilers to implement this function. After a CDFG is created, we apply the proposed method to the CDFG. And, the result of the proposed method is generated as a Verilog HDL description. This output description has all information which is required to synthesize the design. Thus, we can synthesize this Verilog HDL description easily by using generic synthesis tools.

We have verified that our implementation synthesizes properly the input design by synthesizing a small benchmark since we have not implemented the function that calculates the difference of the CDFG. Figure 6 shows the benchmark which we used. The benchmark is written in the C language, and converted to the CDFG which has the

27 nodes. The global variables in the C language is to communicate with the external and is bound to a local store. We have generated the Verilog HDL description which has the 1119 lines from the CDFG by using our implementation. This process have finished in a few seconds. We have simulated the output Verilog HDL description and verified that the result of the simulation is correct.

5 Summary and Future Work

In this paper, we proposed a scalable heuristics for incremental high-level synthesis which causes only small changes on register transfer level description if the design on high-level language is changed. And, we implemented this method in our SORA framework, and tested using small simple designs.

In the future, we plan to demonstrate the advantage of our proposed method through evaluating the quality of the synthesized circuit and the runtime. Since the proposed method is a heuristic, it is not guaranteed that the result generated by our proposed method is optimal. We will implement an optimal method which generates an optimal result and compare our proposed method and the optimal method in terms of the amount of the changed hardware and the efficiency of the results. Then, we will show that our proposed method can synthesize practical designs by evaluating the runtime of our proposed method. Besides, we will demonstrate that the proposed incremental technique is useful for the CEDD technique.

References

[1] L. Lavagno, A. Kondratyev, Y. Watanabe, Q. Zhu, M. Fujii, M. Tatesawa, and N. Nakayama, “Incremental High-Level Synthesis,” in *Proc. Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan. 2010, pp. 701-706.

[2] R. Karri and B. Iyer, ”Introspection: A register transfer level technique for cocurrent error detection and diagnosis in data dominated designs,” *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 6, no. 4, pp. 501–515, Oct. 2001.

[3] Y. Liu and K. Wu, ”Towards Cool and Reliable Digital Systems: RT Level CED Techniques with Runtime Adaptability,” in *Proc. IEEE Int. Conf. Computer Design*, Oct. 2010.

[4] P. Brisk, F. Dabiri, R. Jafari, and M. Sarrafzadeh, “Optimal Register Sharing for High-Level Synthesis of SSA Form Programs,” *IEEE Trans. Computer-Aided Design*, vol. 25, no. 5, pp. 772-779, May 2006.

[5] J. Llosa, A. Gonzalez, E. Ayguade, and M. Valero, “Swing Modulo Scheduling: A Lifetime-Sensitive Approach,” in *Proc. IEEE Int. Conf. on Parallel Architecture and Compilation Techniques (PACT)*, Oct. 1996, pp. 80-87.

[6] C. Lattner and V. Adve, ”LLVM: A compilation framework for lifelong program analysis transformation,” in *Proc. IEEE/ACM Int. Symp. on Code Generation and Optimization (CGO)*, May 2004, p. 75.

O-51/R73

A study of failure mechanisms in CMOS & BJT ICs and their effect on device reliability.

M.G.Rajesh[1], Gopika Vinod[2], D.Das[1], V.Bhatnagar[1], C.K.Pithawa[1]

1. Electronics Division, BARC 2.Reactor Safety Division, BARC

Aditya Thaduri[3], A.K.Verma[3]

3. Department of Electrical Engineering, IIT Bombay

Abstract – **The reliability of electronic systems, used in nuclear power plants, is traditionally estimated with empirical databases such as MIL-HDBK-217, PRISM etc. These methods assign a constant failure rate to electronic devices, during their useful life. Currently, electronic reliability prediction is moving towards applying the Physics of Failure approach which considers information on process, technology, fabrication techniques, materials used, etc. The constant failure rate assumption stems from treating failures as random events. Electronics division of BARC is engaged in design & fabrication of CMOS and BJT ASICs for nuclear pulse processing. These new microelectronic devices often exhibit infant mortality and wear-out phenomena while in operation. It points to competing degradation mechanisms-electro migration, hot carrier injection, dielectric breakdown etc.-that make a device's useful life different from that predicted by empirical methods. Understanding the dominant mechanisms that lead to device failure –Physics of Failure– is a more realistic approach to reliability prediction. This paper describes common failure mechanisms- encountered in CMOS and BJT ICs and the efforts being taken to quantify these effects in an ASIC -4N36- which forms a part of the isolation in neutron flux monitoring systems.**

Keywords: Reliability; Optocoupler; Failure Mechanism; Design of experiments; Physics of failure.

I. INTRODUCTION

Industries employ different technologies like CMOS, BJT and BICMOS for various applications. The possibility of chance of failure at interdependencies of materials, processes, and characteristics under operating conditions is the major concern which affects the performance of the devices. They are characterized by several failure mechanisms and hence failure models of these devices should consider them at various stages such as wafer level, interconnection, etc. For this, the dominant failure mechanisms and stress parameters needs to be identified.

Design of experiments is an efficient and prominent methodology for finding the reliability of the item, as the experiment provides a proof for the hypothesis under consideration. One of the important techniques involved is Taugachi method which employs for finding the prominent failure mechanisms in semiconductor devices. By physics of failure approach, the factors that are affecting the performance on both environmental and electrical parameters with stress levels are identified. By constructing taugachi array with these parameters where output parameters decides the effect of top two dominant failure mechanisms and their extent of chance of failure can be predicted. This analysis helps us in making the appropriate modifications considering both the failure mechanisms for the reliability growth of these devices. This paper highlights the application of design of experiments for finding the dominant failure mechanisms towards using physics of failure approach in electronic reliability prediction.

II. FAILURE MECHANISMS

The most prominent failure mechanisms for CMOS and BJT technology are Hot Carrier Injection (HCI), Electromigration, Temperature dependence dielectric breakdown and Negative Bias Temperature Instability [1],[2],[3] and [4]. Other mechanisms include Stress Migration, Radiation effects, corrosion and thermal fatigue.

A. Electro Migration (EM)

Failure occurs mainly due to the blocking (or voids) of interconnects due to transport momentum at conductor-metal interface. Sometimes atoms of one conductor pile up to another conductor to cause short-circuit (Hillock Failure of whisker failure). Mostly happens at higher current density (>10^5 A/cm2) and at higher temperatures. Ea = 0.5 – 0.8 eV. Causes due to Grain boundary diffusion on Al wires and surface diffusion in Cu wires and Thermal Effects: high power collide scattering joule heating. The Black's Equation is given below.

$$MTTF = A(J^{-n})e^{\left(\frac{E_g}{KT}\right)} \quad (1)$$

Where J is current density in the conductor, K is Boltzman's constant, T is temperature in Kelvin. Aluminium (Ea = 0.6+/- 0.1eV) has good conductivity, good ohmic contacts and adherence to substrate. Copper(Pure) is more robust (Jcu = 5JAl). Ea Increases and mobility increases by adding 1% palladium. EM is proportional to current density & smaller grain boundaries. For a Bamoo structure; if, width is proportional to average grain size then the Electromigration decreases. For large magnitude currents, slotted wires are used to meet power requirements. Blech Length: Lower limit of length of interconnect to allow EM. Solder joints (SnPb, SnAgCu) occur at lower current densities [4].

B. Temperature Dependence Die-electric Breakdown

Failure occurred due to continuously applying stress to Gate oxide film which makes Di-electric falling shorting Anode and Cathode by increased Electric field. Time to failure increases with increasing electric field and Temperature. But as Electric field decreases, activation energy increases and thus stresses increases. For high fields (<10 MV/cm), field enhanced thermal bond breakage. Decrease in the activation energy leads to electron reaction rate [2].

$$\text{E-Model: } MTTF = Ax10^{-\beta E} xe^{\left(\frac{Ea}{KT}\right)} \quad (2)$$

An E on oxide film causes injection of holes on anode side causing traps. As traps increases in oxide, current produces by SILC (stress induced leakage current) due to tunneling. And traps between Gate and Silicon substrate, leakage current increases leads to gate oxide to break down.

$$\text{1/E Model: } TF = \tau_0(T)e^{\left(\frac{G(T)}{Eox}\right)} \quad (3)$$

1/E Model: For Low Electric fields, current by Fowler Nordheim conduction. Electrons experience impact ionization that damages the di-electric, which accelerated. Accelerated electrons reach anode produces hot holes which tunnel back to dielectric (hot hole injection mechanism).

For ultra-thin oxides, Temperature is non-Arrhenius,

$$MTTF = T_{BD0}(V)e\left(\frac{a(V)}{T} + \frac{b(V)}{T^2}\right) \quad (4)$$

C. Hot Carrier Injection

Carriers in increased Electric field, accelerated to gain Energy. Some charges have Hot Energy to overcome Potential between Gate and Substrate. These carriers injected to Gate (some are trapped), form a space charge, change in Vt & gm. Injected carriers which are not trapped as gate current and others as substrate current [3].

i. Drain Avalanche Hot Carrier DAHC injection

   Electrons from Source lead to impact ionization because of high electric field at Drain, which generates electron-hole pairs and which has higher Energy injected to Gate. Vgs = ½ Vds. The greatest factor is at normal temperatures.

ii. Channel Hot Electron CHE injection (Vgs = Vds, lucky electrons which are not energy dissipation)

iii. Secondary generated hot electron SGHE injection

iv. Subsrate hot electron SHE injection

$t = CxIsub^{-m}$ and $t = Axe^{-B/Vds}$ are the time dependent models.

HCI is prominent at lower temperatures [1]. Thermal vibrations increases and collisions decreases, means have higher probability for mean free path of electrons to absorb more energy. Higher electric field injects carriers and probability of injecting increases. Impact ionization increases and thus increases the secondary electrons. As Source Voltage decreases, impact ionization mode has been changing.

Degradation by HCI is $\Delta P = At^n$, Where P= parameter gm, Vth, i_{sat}.

$$\text{n- Channel, Eyring Model } MTTF = B(I_{sub})^{-N} e^{(Ea/KT)} \quad (5)$$

$$\text{P-channel } MTTF = B(I_{gate})^{-M} e^{(Eg/KT)} \quad (6)$$

Substrate current Vs Voltage in p-channel is substrate doubles for each 0.5V increase in Vsd. Acceleration factor is defined in Eq. (7)

$$AF = e^{(B(1/Vdd - 1/Vdd,max))} \quad (7)$$

D. Negative Bias Temperature Instability (Slow Trap)

Shift in Vt. Holes trapped between Si/SiO2 interfaces, especially in PMOS. Holes are thermally activated and gain sufficient Energy to disassociate Si/SiO2 defects near LDD. Concentration of holes is directly proportional to temperature. NBTI decreases Idsat, decrease in gm & Ioff increases and Vth increases. (T = 100-25°C, E = 6MV/cm).

$$\Delta Vth = Af_1(t)f_2(Vg)e^{(-Ea/KT)} \quad (8)$$

Silicon dangling bond on interface inactivated by H, Si-H, stress (High Temperature), increase in bias, holes gives to electro-thermal reaction frees H. Silicon dangling bond becomes interface state and H diffuses in oxide film. Some diffusing H joins with defects to form traps. Increase in interface state and charge resulting from traps in oxide for degrading Vth. Recovery by removing stress bias and applying reverse bias. NBTI is important in circuits in which DC stress is applied.

III. STUDY, OPERATION AND TESTING OF 4N36 OPTO-COUPLERS

A. Introduction

An Optocoupler (in Fig: 1) or optoisolator is a cool little device that allows you to completely separate sections of an electric circuit. An optocoupler or sometimes refer to as opto-isolator allows two circuits to exchange signals yet remain electrically isolated. It consists of and LED at the input and Photo-transistor at the output and the isolation is implemented by light medium [5].

Due to the degradation of Opto-couplers, Reliability plays important role.

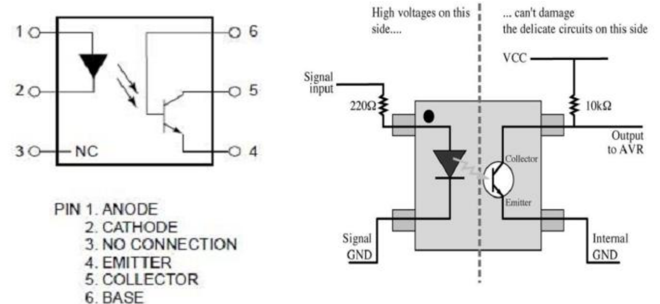


Fig: 1 Block Diagram of 4N36

Current transfer ratio (CTR) is the main characteristic for operation of Optocouplers. For fixed V_{ce}, CTR as in Eq. (9)

$$CTR=\frac{I_{Collector}}{I_{Diode}} \quad (9)$$

The current transfer ratio (CTR) is the amount of output current derived from the amount of input current. CTR is normally expressed as a percent. CTR is affected by a variety of influences, including LED output power, hFE of the transistor, temperature, diode current and device geometry. If all these factors remain constant, the principle cause of CTR degradation is the degradation of the input LED. Other characteristics include Id Vs Vd, transmission speed and operating temperature range.

B. Optocoupler Input (LED) [8]

The area of greatest concern in optocoupler reliability has been the infrared LED. The decrease in LED light output power over current flow affects the performance. Companies are focused on the infrared LED to improve CTR degradation and consequently achieved a significant improvement in coupler reliability. The improvements have included die geometry to improve coupling efficiency, metallization techniques to increase die shear strength and to increase yields while reducing user cost, and junction coating techniques to protect against mechanical stresses, thus stabilizing long-term output.

C. Optocoupler Output (Photo-Transistor)

There are varieties of outputs available like bipolar transistors, MOS, SCR with different ratings to suit particular applications. The slow change in the electrical parameters over time when voltage is applied which is termed as electric field. This is extreme at 100C with high voltage 1KV. This is due to the release of charge carriers which results in change of gain, reverse current and reverse voltage where direction of field is important. To improve characteristics, pn junctions are protected by transparent ion screen.

D. Experimental Setup

4N36 is 6-pin Dip. The pins on the left side refer to LED inputs and on the right side refer to phototransistor output.

Ageing Tests: There were using several accelerated wear-out tests for ageing tests of optocouplers. Many parameters include LED ageing and ambient temperature on photo-detector is suspected.

Circuit Diagram for 4N36: Two types of tests are carried out; temperature and input current in Fig 3. In order to highlight the ageing tests of optocouplers, first measure their funcional characteristics after that ageing tests should be implemented.

Optocoupler Parameter Drift: Ageing tests are carried out in two batchs: LED side and photo-transistor side. Two batches of 20 4N36 ICs with ambient temperature as 30C and junction temperature as 105C using Fig 2.. By studying the variation of CTR on the input measurement current on both the batches, degradation of components stressed in photo-transistor is insignificant event after 1000 hours. Significant degradation exists in LED aging where for smaller currents it is more rapid up to 100 hours and after that rate decreases with time.

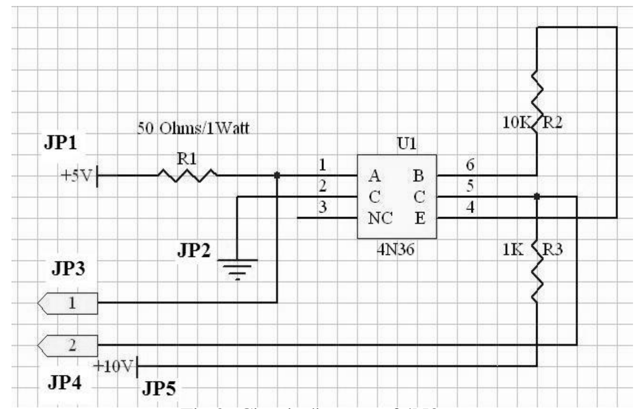


Fig 2: Circuit diagram of 4N36

Ageing Parameters: From the studies, LED ageing was prominent role for degradation. We need to select the stress parameter which degrades CTR; temperature or input current. A batch with high LED current under ambient temperature and another with high temperature value with low LED current.

Results: The effect of response times is negligible on both the tests. But CTR degradation is prominent with high current stress test. So input current is ageing parameter in CTR degradation.

Variation of CTR with current and time:

From the above results, drift in CTR depends on I and t. Failure is considered as if CTR reaches 50% lesser than original value.

$$D(t)=\frac{CTR(0)-CTR(t)}{CTR(0)} \quad (10)$$

Modeling of Optocouplers ageing [5]:

Bajenesco proposed a model for optocouplers ageing in terms of life time as Eq. (11), junction temperature and current across LED.

$$t_1=\frac{A}{I_d e^{-\frac{E_a}{KT_j}}} \quad (11)$$

Where t1 is lifetime, Id is ageing (LED) Current, Ea is Activation Energy, Ea = 0.15eV, K is Boltzman Constant, Tj is Junction Temperature and A is Time factor A.s

The main cause for CTR degradation is the reduction in efficiency of the LED in the optocoupler. Its quantum

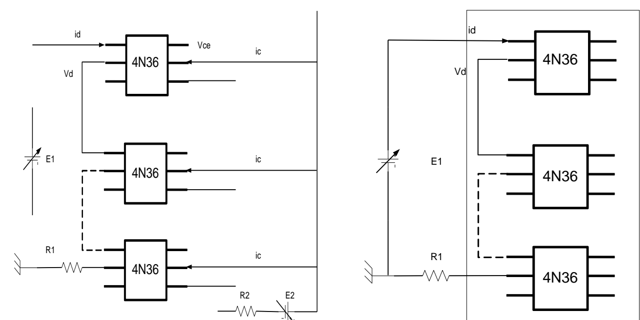


Fig 3: (a) Photo detector ageing (b) LED ageing

efficiency (total photons per electron of input current) decreases with time at a constant current. The LED current in Eq (12) consists of two components diffusion current and a space-charge recombination current.

$$If(Vf)=Ae^{qVf/KT}+Be^{qVf/KT} \quad (12)$$

For constant current, if recombination current increases due to B, then diffusion current, the radiative component will decrease. This reduction is due to both current density and junction temperature. In general, emitter current density is a function of current, junction geometry, resistivity of both the regions of diode. Junction temperature is a function of coupler packaging, power dissipation and temperature. As with current density, high Tj will increase rapidly in proportion with recombination current. From the block diagram of abstract *optocoupler [7] and CTR expression is given as in Eq. (13),*

$$CTR=I0/If(100\%)=KR\eta(If,t)\beta(Ip,t) \quad (13)$$

Where K is Transmission factor, R is Resistivity of photodetector, η is Quantum efficiency with function of input current and time and β is Gain of output amplifier with function of photo current and time

Temperature variation affects the efficiency and gain. The normalized CTR is given as in Eq. (14)

$$\frac{\Delta CTR}{CTR}=\frac{\Delta\eta}{\eta}+\frac{\Delta\eta}{\eta}\left(\frac{\delta\ln\beta}{\delta\ln Ip}\right)+\frac{\Delta\beta}{\beta} \quad (14)$$

1st term: Major contribution to normalized CTR. In general, it is negative over time.

2nd term: Second order effect of shift in Q point of amplifier as efficiency changes.

3rd term: Negligible effect with change in gain over time.

Lindquist Model [6]:

Lindquist also suggested a model describing the relative (CTR) degradation in terms of ageing current and measurement current as Eq. (15).

$$D(\%)=\frac{j(0)-j(t)}{j(0)}=1-e^{q\Delta V/KT} \quad (15)$$

Since the transistor life is more comparing to LED, the optocoupler ageing mainly depends on degradation of LED light output which is flux.

$$\frac{\Delta CTR}{CTR}=\frac{\Delta j}{j} \quad (16)$$

Where output flux as in Eq. (17) is a function of efficiency of opt coupler and diffusion current.

$$j=\eta.I_{diff}=\eta.\alpha.e^{qV/KT} \quad (17)$$

By the above model, ΔV increases with decreasing current means degradation increases as measurement current increases. For direct band gap emitters, the degradation is due to non-radiative component at which flux is measured. Current density J(V) in Eq. 18 is combination of radiation and non-radiation current densities

$$J(V)=\frac{I(V)}{A}=\alpha e^{q\Delta V/KT}+\gamma e^{q\Delta V/2KT} \quad (18)$$

Where α denotes the coefficient of diffusion (Radiation) and γ denotes the coefficient of recombination current (Non-radiation). Taking the boundary values into consideration in current density equation, by solving, change in gamma coefficient with respect to initial value can be found as Eq.19

$$\frac{\Delta\gamma}{\gamma(0)}=-2\xi_0\sinh\left(\frac{q\Delta V}{2KT}\right)+e^{-q\Delta V/2KT}-1 \quad (19)$$

Where ξ, the ratio of diffusion current and recombination current and is given as in Eq. 20

$$\xi=\frac{\alpha}{\gamma}e^{qV/2KT} \quad (20)$$

Substituting this value in the degradation mechanism, the final equation is in Eq. 21

$$\frac{D(\%)}{100}=1-\frac{\left(1+\frac{\Delta\gamma}{\gamma(0)}\right)^2}{4\xi_0^2}\left\{\sqrt{1+\frac{4\xi 0(\xi 0+1)}{\left(1+\frac{\Delta\gamma}{\gamma(0)}\right)^2}}-1\right\}^2 \quad (21)$$

$where: \Delta\gamma=\gamma(t)-\gamma(0)$ and ξ, for the values of γ(0) and V(0). Suppose, change in γ is proportional to current, $d\gamma/dt = b.I_s$, then the drift model of CTR is in Eq (22)

$$\frac{\Delta CTR(t,I_s,I_m)}{CTR}=1-\frac{\left(1+\frac{b.I_s.t}{\gamma(0)}\right)^2}{4.C^2.I_m}\left\{\sqrt{1+\frac{4.C.\sqrt{I_m}(C.\sqrt{I_m}+1)}{\left(1+\frac{b.I_s.t}{\gamma(0)}\right)^2}}-1\right\}^2 \quad (22)$$

Where t = time in hours, I_s = ageing current in A, I_m = measurement current in A, γ (0) = 10-12(A), C = 80(A-1/2) and b = constant related to optocoupler

Cause of CTR Degradation:

Total electron flux emitted by LED degrades slightly over operating time of the device. At higher stress currents, change of light output increased over time. Main causes include reduction in emitter efficiency, decrease in transmission ratio, and reduction in responsiveness of photodetector or change in gain of amplifier which all are dependent on time. The critical cause is the result of electrical and thermal stressing of PN junction. Assuming degradation mechanism establishes a resistive shunt parallel to active PN Junction. At low values of input current, resistance path exhibits appreciable impact on the performance which offers low resistance. As current increases further, junction experiences low resistance which draws more current.

Reliability of optocouplers:

Important area of investigation is the light output test of LED, assembly area in die attach and wire bond. Temperature cycle is a more effective screen than stabilization

bake. Temperature coefficient of expansion and low glass transition temperature of unfilled, clear plastics is much greater than that of the other components. To maintain reasonable device integrity requires temperature range of operation and stronger mechanical construction; some clear plastics build up mechanical stress on the encapsulated parts during curing. This stress has been likened to rapid, inconsistent degradation of IRED light output.

Although a filled plastic would stop these phenomena, the filler also spoils the light transmission properties of the plastic. The decrease in quantum efficiency of LEDs is the main reason for CTR degradation of optocouplers. Other less important causes of CTR degradation are a decrease in the transmission of the transparent epoxy, a change in sensitivity of the photodetector and a change in gain of the output amplifier. It is now known that the rate of CTR degradation is influenced by the materials and processing parameters used to manufacture the LED, and the junction temperature of the LED in addition to the current density through the LED.

IV. DESIGN OF EXPERIMENTS

From the above study, input current to LED is dominant for the degradation of CTR for optocouplers which interm is dominant stress parameter. Temperature is second dominant parameter for optocoupler which also degrades CTR. Other stress parameters inculdes radiation effects [9] and humidity were negligible. Statistically, the effect of both the stress parameters is not quantified. To define this, the prominent Design of experiments by Taguachi method is implemented here which also involves screening of stress parameters. It gives the statistical measure of amount of S/N generated by specific parameter at specific level. This also helps in choosing the design parameter for extented MTTF of the item.

Extended Taguachi Method [10]:

Item: 4N36 (I_f: 10 mA, 25C, CTR = 100) I_f: (0, 100mA) and T (-55, 150C)

Failure Mechanisms: LED ageing Stress Parameters: Input LED current and Temperature.

Levels: I_f (H: 70mA; L: 10mA) and T (H: 100C, L: 30C)

Samples: For each run, n = 5 sampels are selected.

Measurement Paremeter: CTR is calculated by measuring currents on both input and output. Bigger is better (B-Type Statistics).

Using the above two level parameters, L4 Array is constructed as in Table 1.

Table 1. L4 Array

Run No:	A(I_f)	B (T)	AXB
1	1	1	1
2	1	2	2
3	2	1	2
4	2	2	1

The average of CTR is

$$\bar{Y} = \frac{Y1 + Y2 + Y3 + Y4 + Y5}{n} \tag{23}$$

Signal to Noise Ratio is defined as S/N = -10log MSD where

$$\text{MSD}=\frac{1/Y1^2+1/Y2^2+1/Y3^2+1/Y4^2+1/Y5^2}{n} \tag{24}$$

Where Y_Y = Average of Y_{AVG} value for that level.

By calculating results table and response table, the effect of the both the parameters are quantified and will maximize the CTR for the respective selection of parameters.

V. CONCLUSION

In this paper, the most dominant failure mechanisms that are responsible for the degradation of the performance are illustrated. A detailed study, operation and modelling of an optocoupler 4N36 is selected and reliability analysis is carried out. LED aging is responsbile for the degradation of CTR of optocoupler and mainly due to increase in the input current and temperature. Statiscal analysis using design of experiments is implemeted here for the screening of the stress parameter and also the selection of the design parameter level for increase in the CTR thus by improving the reliability. The dominant failure mechanism which affects the performance is found to be degradation of LED by input.

VI. REFERENCES

[1] Semiconductor Device Reliability Failure Models SemaTech, AMD, 2000

[2] Electronic circuit reliability modeling Joseph, Moshe, Univ of Maryland, 2006, Elsevier, ScienceDirect

[3] Sony Quality and Reliability Handbook.

[4] Renesas Semiconductor Reliability.

[5] Study and modelling of optocouplers Ageing, J.B.H. Slama, H.Helali, A.Lahyani, K.Louati, P.Venet and G.Rojat, Automation and Systems Engineering

[6] A New model for light output degradation of direct band gap emitters. P.F.Lindquist, HP, 1980

[7] Ageing Problem of optocouplers. I.Bajenesco. IEEE 1944

[8] Manufacturing and Reliability of optocouplers, Vishay Semiconductors, 2006.

[9] Radiation Characterization and Test Methodology Study of Optocouplers Devices for Space Applications, Mangeret, Bonara, 2002

[10] Reliability improvement with design of experiments, 2nd edn, Lloyd W. Condra

Test Data Compression Technique for IP Core Based SoC using Artificial Intelligence

Usha S. Mehta*, Kankar S. Dasgupta**, Nirnjan M. Devashrayee*, Harikrishna Parmar***

** Institute of Technology, Nirma University, Ahmedabad* *** Space Application Center, ISRO, Ahmedabad*
***C. K. Pithawala College of Engineering and Technology, Surat

***Abstract*:** Test power and test time have been the major issues for current scenario of VLSI testing. The hidden structure of IP cores in SoC has further exacerbated these problems. The test data compression is the well known method used to reduce the test time. The test vector reordering method can be used for effective reduction in scan power. In this paper, in beginning, the A* algorithm for from Artificial Intelligence is used to reorder the test vector. The quality parameter used for reordering is Weighted Transition Matrix (AWTM) considering both, scan-in and scan-out vectors. The The experimental results on ISCAS benchmark circuit proves that the proposed method gives better power reduction.

***Index Terms*— Scan-in Power, Test Vector reordering, WTM, Artificial Intelligence.**

I. Introduction

Advancements in semiconductor fabrication technology has helped the design engineers to accommodate more number of transistors in a VLSI chip. With the proliferation of mobile battery-powered devices, reduction of power in the embedded VLSI chips has become an active area of research. During the last decade, power reduction techniques have been proposed at all levels of the design hierarchy - from system to device levels. For the development of complex, high performance, low power devices implemented in deep submicron technology, power management is a critical parameter and it cannot be ignored even during testing. With the increase in the density of the chips, the problem of testing has also increased manifold.

A related problem is to achieve power reduction during the actual testing of a chip [10]. Power consumption in test mode is considerably higher than the normal functional mode of a chip. The reason is that test patterns cause as many nodes switching as possible, while a power saving system mode only activates a few modules at a time. Thus, during testing switching activity in all the internal lines of a chip is often several times higher than during normal operation. Sometimes parallel testing is used in SoCs to reduce test application time, which results in excessive power dissipation. Again, successive functional input vectors applied to a given circuit during system mode have a significant correlation, while the correlation between consecutive test patterns can be very low. Usually, there is no definite correlation between the successive test patterns generated by an ATPG (for external testing) or by an LFSR (for BIST) for testing of a circuit. This can cause significantly larger switching activity in the circuit during testing than that during its normal operation. Abnormal power consumption during testing leads to adverse effects on the chip and the testing process such as, (a) this may give rise to severe hazards to the circuit reliability and lead to long or short-term malfunction, (b) it can cause chip destruction due to excessive heat in absence of proper heat dissipation mechanism, (c) it can increase the packaging and cooling costs, (d) it can cause the chip to falsely fail the test due to noise problems such as, *IR* and *Ldi/ dt* drops, which may be a source of yield loss and increase in production cost, (e) it may make it difficult to obtain a carefully tested bare die to be used in multichip modules (MCM) or what is called the Known Good Die problem (KGD) and (f) it can dramatically shorten the battery life when on-line testing is considered. For all these reasons, various techniques have been proposed to reduce the impact of high power consumption during test application. Low power dissipation during test application is becoming an equally important figure of merit in today's VLSI circuits design with BIST and is expected to become one of the major objectives in the near future.

In this paper an AI-based approach is proposed to reorder the test vectors such that the switching activity and hence the power dissipation during testing is reduced. The paper is organized as follows: preliminaries and prior works for low power testing techniques and test vector reordering is covered in Section 2. In Section 3, the motivation of the work and the problem formulation is described. Section 4 explains the AI-based approach of test vector reordering in detail. Empirical results and performance comparison is presented in Section 5 followed by concluding remarks in Section 6.

II. Preliminaries and Prior Work

A. Power dissipation in CMOS technology

The Power dissipation in CMOS technology [14,15] can be classified into static and dynamic. Static dissipation is due to leakage current that has small magnitude in digital CMOS circuits. Hence, for such circuits, the dynamic dissipation is the dominant term. Dynamic dissipation occurs at a node when it switches from one logic level to another logic level. Dynamic dissipation is divided into two components caused by short circuit current and charge/discharge current. The former is caused by a current pulse that flows from power supply to ground when both n- and p- transistors are simultaneously ON during small interval of switching period and is negligible in high speed circuits where n- and p-transistors are simultaneously ON for very short periods. The later is caused by switching activity of transistors during 0 1 or from 1 0 transitions. The charge/discharge current is the current that charges and discharges the capacitive load on the output of a gate during 0 1 or from 1 0 transitions and in

general, dominates dynamic power dissipation [15,16]. The dynamic power dissipation (PD) in the circuit is given by (1)

$$PD= 1/2\ \Sigma j^{n}\ CL(j)\ s(j) Vdd^{2} \quad (1)$$

where CL(j)is the load capacitance at line j of the Circuit Under Test (CUT), s(j) is the frequency of switching of the line j, and Vdd is the power Supply voltage. Other quantities being constant for a given circuit, test vector set generated to minimize the frequency of switching at circuit lines during test application will minimize the heat dissipation during testing.

B. *Prior work*

A survey on low power testing of VLSI circuits has been given in [5]. There are several techniques for low power testing of VLSI circuits such as, test vector reordering, scan chain ordering, power-constrained test scheduling, use of multiple scan chains, low power test pattern generation, vector compaction, etc. Test vector reordering is a very well-known technique to reduce dynamic power dissipation during combinational circuit testing through switching activity minimization in the circuit. Test vector reordering is an essential task in testing VLSI systems because it affects from two perspectives: power consumption and correlation among data used for test data compression. The problem of test vector reordering can be mapped into finding Hamiltonian cycle in a complete weighted graph, which is known to be NP-hard. So, there is no polynomial time solvable algorithm for the problem. Therefore, it is essential to find a good heuristic-based solution for the problem. Existing approaches have used several heuristics for solving this problem. In [11], the problem of test vector reordering has been mapped into finding the Hamiltonian path in a fully connected weighted graph which is similar to the traveling salesman problem (TSP). As there exists no polynomial time algorithm for TSP, approximation methods of solution have been used. Solutions for three cases have been given: (i) reordering for maximal or minimal activity, (ii) reordering of test vectors with a desired circuit activity across the VLSI chip while achieving a high coverage for stuck-at faults, and (iii) reordering for localized switching activities to maximize it in one part and minimize at other part of the circuit. In another work [6], proposed greedy algorithm has guaranteed decrease in power consumption without modifying the initial fault coverage. A second technique based on Simulated Annealing (SA) has been proposed in which the greedy solution is used as initial solution and it shows a considerable average power reduction during test application Here, only Hamming distance between test vectors has been used to avoid simulation of the circuit and providing a solution (an Hamiltonian path of minimum cost in the Hamming distance graph) in a short computation time. It has been shown that there is a correlation between the Hamming distance and the transition activity. But, if signal transitions in the internal line be considered, then obviously optimal solution can be found. Another work [4] has also considered the Hamming distance minimization between adjacent vectors to reduce the dynamic power dissipation during testing. In [3], reduction of power dissipation during test application has been studied both for scan designs and for combinational circuits tested using built-in self-test (BIST). They have shown that heuristics with good performance bounds can be derived for combinational circuits tested using BIST and a post- ATPG phase has been proposed for reducing power dissipation during test application in full-scan circuits and for pure combinational circuits. They have shown that scan-latch ordering along with test-vector reordering can give considerable improvement in power dissipation and considerable savings can be obtained by repeating some of the test vectors. In [8], an evaluation of different heuristic approaches has been done in terms of execution time and quality. Here, it has been shown that the Multi-Fragment heuristic performs better than Christofides and Lin-Kernighan heuristics in terms of time. It also outperforms the Christofides heuristic in terms of quality and achieves performance very close to Lin-Kernighan. They recommended reordering algorithms to use the Multi-Fragment heuristic for near minimal ordered sets of vectors that result in both reduced power consumption and enhanced data compression ratio. Recently, some works have formulated test vector reordering problem as TSP and Genetic Algorithm (GA) has been used to generate low power test patterns.

Chattapadhyay and Choudhary have shown proposed a GA-based formulation to solve the problem of generating a test pattern set such that it has high fault coverage and low power consumption has been proposed in [2]. They have shown a method of selecting a subset of test vectors generated by an ATPG tool to reduce power dissipation by sacrificing a small amount of fault coverage. In [13], authors have studied two well known search methods (2-opt heuristic and a GA-based approach) with reduction in fault coverage. They have also combined those two methods for power reduction.

Sudip Roy et.al has proposed a test vector reordering technique with Artificial Intelligence approach. They have shown the method only for combinational circuits C17 and they achieved almost around 22% reduction in switching activity.[17]

III. MOTIVATION OF THE WORK AND PROBLEM FORMULATION

A. *Motivation of the work*

Most popular techniques for test power minimization orders the deterministic test patterns and several approaches have been followed for test vector reordering such as, finding minimum cost Hamiltonian path after mapping the problem into TSP instance, finding optimal solution by applying GA or SA. Although the dynamic power minimization problem by test vector reordering during VLSI testing is an old problem, here new approach is proposed for solving it using Artificial Intelligence (AI). This problem can again be viewed as finding optimal path from start to goal node in a search space by applying informed search methods of AI, where start node is the node when no test vector is selected and the goal node is the node when only one test vector is remaining for selection. A* search algorithm is a very well-known informed search method used in AI. It takes advantages of both efficiency of greedy search and optimality of uniform-cost search by simply summing the two evaluation functions. Thus, it is optimally efficient algorithm for finding optimal solution in an informed search space. This has motivated us to apply A* search technique for test vector reordering problem for dynamic power reduction during testing.

Also Previous work of Sudip Roy et al. motivated us to implement Artificial Intelligence algorithm for test vector reordering to sequential circuits in order to reduce scan in scan out power.[17]

B. *Problem formulation*

Consider a test set for a combinational circuit is given by T = *{t*1,t2*, ..tk} and its output response is given by O={o1,o2,….ok}* with a predefined fault coverage, where */T/ = k*. Each test vector is formed by a fixed ordered set of bits *bj* i.e., t*i* =< *b*1,*b*2*, ...bl* >, where *l* =length of the test vectors or the number of primary inputs (PIs) of the circuit. Assume Π be the initial ordering of test vectors *T*.

The problem of scan in scan out power minimization by test vector reordering is to compute an optimal vector ordering Π_ of T such that total scan in scan out power dissipation in the circuit during testing is minimized. The problem of reducing the peak power dissipation is not considered here. Only the average power reduction has been considered. Since, the power dissipation is directly proportional to switching activity, the problem can be restated as to find out an optimal path or optimal ordering of vertices Π_ from the search space of having all possible orderings of vectors *V* such that total switching activity in the circuit is minimized.

IV. AN A*-BASED METHOD FOR DYNAMIC POWER MINIMIZATION BY TEST VECTOR REORDERING

A. *Basic underlying principle for A* algorithm*

The A* algorithm combines features of uniform-cost search and pure heuristic search to efficiently compute optimal solutions. A* is a best-first search in which the cost associated with a node is given by the evaluation function $f(n)$.

$$f(n) = g(n)+h(n) \qquad (2)$$

where $g(n)$ is the cost of the path from the initial state to node n, and $h(n)$ is the heuristic estimate of the cost of a path from node n to a goal node. Thus, $f(n)$ estimates the lowest total cost of any solution path going through node n. At each point, a node with lowest f -value is chosen for expansion. Ties among nodes of equal f -value is broken in favor of nodes with lower h-values. The algorithm terminates when a goal node is chosen for expansion. For a given node, the sum [current cost + heuristic value] is an estimation of the cost of reaching the ending node from the starting node, passing by the current one. This value is used to continuously choose the most promising path. In practice, the algorithm maintains two lists of nodes that are filled and modified during the search: an **OPEN** list and a **CLOSED** list. **OPEN** list is a priority queue, contains the tracks leading to nodes that can be explored in increasing order of the evaluation function $f(n)$. Initially, there is only the starting node and at each step, the best node of **OPEN** list is taken out. Then, the best successor of this node (according to the heuristic) is added to the list as a new track. The **CLOSED** list stores the tracks leading to nodes that have already been explored.

B. *Cost function g(n)*

A complete weighted graph X(T,*O,E,W*) is constructed where T is the test set, O is the output response, *E* is the set of edges

t1: 010	O1: 100
t2: 001	O2: 110
t3: 110	O3: 100
t4: 111	O4: 111
t5: 010	O5: 000
t6: 010	O6: 011
t7: 100	O7: 000
t8: 100	O8: 111

Figure 1. a.) Test vector b.) Output Response

Weighted Transition Matrix(WTM) is used here to reorder the test patterns. Here different test patterns passed through the output response and hence the switching will occur and from that WTM is made as shown in figure 1 c. The corresponding equation is also shown below.

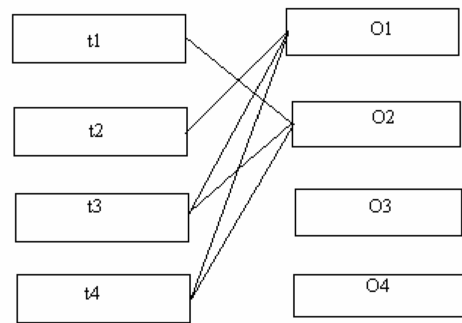


Figure 1 c.) Test vector passing through output response

$$WTM = \sum_{j=1}^{n} \sum_{i=1}^{n} t(j,i) \text{ XOR } O(j,i)*(n-i) \qquad (3)$$

$$\begin{vmatrix} 0 & 2 & 3 & 2 & 4 & 4 & 4 & 4 \\ 5 & 0 & 2 & 1 & 5 & 5 & 1 & 3 \\ 4 & 2 & 0 & 2 & 4 & 4 & 4 & 1 \\ 6 & 4 & 1 & 0 & 6 & 1 & 2 & 2 \\ 3 & 1 & 4 & 3 & 0 & 3 & 5 & 5 \\ 5 & 3 & 2 & 1 & 5 & 0 & 3 & 3 \\ 3 & 1 & 4 & 3 & 3 & 3 & 0 & 5 \\ 6 & 4 & 1 & 0 & 6 & 6 & 2 & 0 \end{vmatrix}$$

Figure 2. Weighted Transition Matrix

and *W* represents the set of weights associated with each edge indicating the total number of signal transitions (i.e., total switching activity) in the whole circuit considering internal lines for consecutive application of two adjacent test vectors. So, WTM (Weighted transition matrix) represents a matrix of total transitions and it is called as Transition graph or *T*-graph. For example, Fig. 4 represents the *T*-graph for the test vectors shown in Fig. 1(a),(b) for S27 benchmark circuit of ISCAS85.

The cost function $g(ni)$ of a successor n_i of the node n is calculated by $g(ni) = g(n)+W(n,ni)$, where W-value is taken from the T-graph.

C. Computing lower bound of switching activity : Heuristic function h(n)

In the A*-based algorithm, lower bound of switching activity among the remaining test vectors is taken as the heuristic function $h(n)$ for a node n. Lower bound of signal transition is computed by bit-wise scanning the test vectors to be selected to reach the *Goal* node. If for *ith* bit of all the test vectors are 1 or 0, then the lower bound of switching activity for *ith* bit is 0; otherwise, it is taken as 1. So, the maximum value of the lower bound is equal to the length of the test vectors, l.

Theorem 1. The lower bound (lb) of switching activity for the remaining test vectors is given by the sum of the lower bounds for all the bits.

An heuristic h is called an *admissible heuristic*, if it never overestimates the cost to reach the goal. The above lower bound of signal transition is computed using a procedure *f ind lower bound*() and it never overestimates the actual cost of reaching the *Goal* node. Thus, it is an *admissible heuristic*. Some examples of calculating h() value has been shown in Fig. 3 for the test vectors shown in Fig. 1(a).

t2 : 001
t3 : 110
t4 : 111
t5 : 010
t6 : 010
t7 : 100
t8 : 100

Lb(bitwise) 111 =3

a.) For the remaining test vector <2,3,4,5,6,7,8> the lower bound h()=3

t3 : 110
t4 : 111
t5 : 010
t6 : 010
t7 : 100
t8 : 100

Lb(bitwise) 111 =3

b.) For the remaining test vector <3,4,5,6,7,8> the lower bound h()=3

t3 : 110
t4 : 111

Lb(bitwise) 001 =1

c.) For the remaining test vector <3,4> the lower bound h()=1

t5 : 010
t6 : 010
t7 : 100
t8 : 100

Lb(bitwise) 110 =2

d.) For the remaining test vector <5,6,7,8> the lower bound h()=2

Figure 3. Examples of calculating lower bound by h(n)

D. The A-based algorithm : AITVR*

Here an approach is propose of test vector reordering for power minimization using the of concept of A*-search of AI and we call it as AI-based Test Vector Reordering (*AITVR*). The dynamic power minimization problem by test vector reordering involves selection of test vectors one by one. Consider a search tree, with *source node* for the case when no test vector is selected, and the goal node for the case when all the test vectors are selected. A node at level p represents the sequence of p test vectors chosen so far. To obtain a sequence/ Ordering with minimum dynamic power dissipation, we find the minimum-cost path from *Source* to *Goal*, with appropriately defined cost. Proposed algorithm *AITVR* finds the *optimal solution path*. Each node of the tree has a cost = g() obtained from so far due to selection of test vectors + h() of the lower bound of switching activity among the remaining test vectors. This implies the following result. *Lemma 1.* The lower bound of switching activity h() is an admissible heuristic.

Theorem 2 [12]. A* returns an optimal solution with an admissible heuristic.

Lemma 1 and Theorem 2 leads to the following result. *Corollary.* For the initial number of test vectors, the algorithm *AITVR* terminates, and always produces a solution with minimum switching activity.

In the following description of the algorithm - *OPEN* is a list containing the un-expanded nodes, *CLOSED* is a list containing the already-expanded nodes, $g(n)$ is switching activity from the *Source* to node n, $h(n)$ is the estimated switching activity from n to *Goal*, and $f(n) = g(n)+h(n)$ is the total estimated switching activity from *Source* to *Goal* passing through node n. The pseudo code for the algorithm is given as *Algorithm AITVR*.

Algorithm 1 AITVR()

---

1. Find the T matrix for the test set V
2. G(Source)=0;
3. H(source)=0;
4. F(source)=0;
5. Put Source in OPEN; found=False
6. while OPEN not empty and not(Found) do
7. Select a node n from OPEN with minimum f -value
8. Remove n from OPEN and put n in CLOSED
9. if n is a Goal node then
10. Found = True

11. else
12. Get the list of test vectors selected so far from
13. Source to the node n. Let, this is Vs
14. Get the list (V_s) of test vectors not selected so far from Source to the node n
15. Expand n (* consider each successor test vector to the next level in V_s *)
16. for each immediate successor ni of n do
17. g(ni) = g(n)+W(n,ni)
18. Find V_s for ni i.e., V_s (ni)
19. h(ni) = f ind lower bound(V_s (ni)
20. f (ni) = g(ni)+h(ni)
21. Put ni in OPEN
22. Direct backward pointer from ni to n
23. end for
24. end if
25. end while
26. Find the remaining test vector v_{rem} to be selected; v_{last} =last test vector selected in the Goal node n
27. Find $SW_{last} = W(v_{last}, v_{rem})$
28. Calculate total switching activity SW = f (n)+SWlast
29. Output SW and solution path Π_ (* sequence of test vectors selected *) by tracing back pointers

---

The proposed algorithms finds the optimal ordering of test vectors from the search space as shown by the path <5,2,7,1,3,8,4,6 > from **Start node** to **Goal node** in Fig. 4.

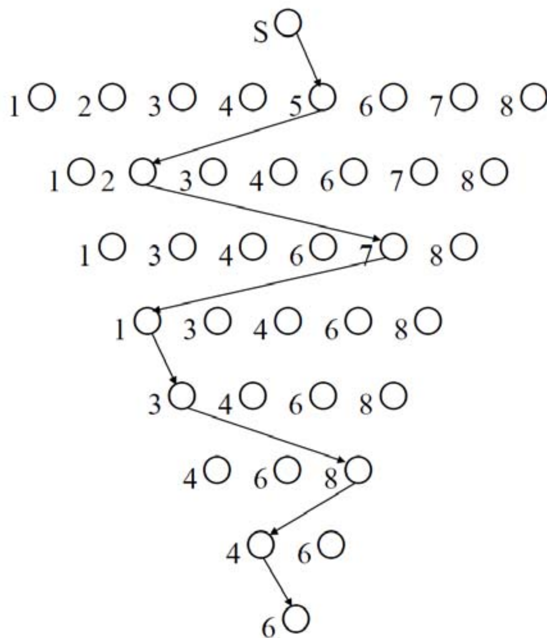


Figure 4. Finding optimal path in the informed search space

E. Time complexity of AITVR

Complexity of A*-based algorithm is obtained from its empirical performance. However, time complexities can be estimated for the heuristic computation, and node expansion. The worst-case complexity for computing *g*() for each node is $O(n^2)$. If l be the length of the test vectors, then the time complexity for calculating *h*()-value for each node expanded is $O(ln)$, in worst case. Thus, a single node generation in *AITVR* requires time $O(n^2)+O(ln)$.

V. Empirical Observation

The proposed algorithm *AITVR* was implemented in MATLAB on an Windows platform with an Intel Pentium IV proces- sor of 1 GHz clock speed and 1 GB RAM. For evaluation of the proposed algorithm, S27 benchmark circuits were considered. First test patterns were generated by the Mentor Graphics. DFT Advisor and Fast Scan are used from Mentor Graphics to generate patterns.

Mentor Graphics ATPG tool with 100% fault coverage and those were taken as initial ordering of the test set. In this work, we have reduced dynamic power consumption during test application without losing stuck-at fault coverage. A simulation-based technique has been used to obtain the actual switching activity at the internal nodes of a circuit using unit-delay model and the total Weighted Transition Matrix was formed. The comparision of the switching activity is shown in in Table 1.

Experimental result shows that the dynamic power of a given combinational circuit can be reduced by about 56.52% after applying the test vectors in the order given by *AITVR* over the test vectors order given by Mentor Graphics, where as Hamming distance method provides 17.39% reduction in switching activity on an average for all the benchmarks.

VI. Conclusion

Test power has become a serious problem with scan-based testing. It can lead to prohibitive test power in the process of test application. In this paper we have proposed an AI-based algorithm for dynamic power minimization through test vector reordering. The proposed algorithm *AITVR* gives optimal solution for the benchmark circuits. The switching activity for the proposed algorithm has been compared with that obtained by Hamming distance techniques. The results are quite encouraging. This has been proved with sequential benchmark circuits.

References

[1] Concorde's TSP solver, http://www.tsp.gatech.edu/concorde/index.html. 2005.

[2] S. Chattopadhyay and N. Choudhary. " Genetic Algorithm based Approach for Low Power Combinational Circuit Testing." In *Proc. of the VLSID*, pages 552–559, 2003.

[3] V. Dabholkar, S. Chakravarty, I. Pomeranz, and S. Reddy. "Techniques for Minimizing Power Dissipation in Scan and Combinational Circuits During Test Application." *IEEE TCAD*, 17(12):1325–1333, 1998.

[4] P. Flores, J. Costa, H. Neto, J. Monteiro, and J. Marques-Silva. "Assignment and Reordering of Incompletely Specified Pattern Sequences Targeting Minimum Power Dissipation." In *Proc. of the VLSID*, pages 37–41, 1999.

[5] P. Girard. " Survey of Low-Power Testing of VLSI Circuits." *IEEE Design and Test*, 19(3):82–92, 2002.

[6] P. Girard, C. Landrault, S. Pravossoudovitch, and D. Severac. "Reducing Power Consumption During Test Application by Test Vector Ordering." In *Proc. of the ISCAS*, pages 296–299, 1998.

[7] H. Hashempour and F. Lombardi. "ATE-Amenable Test Data Compression with No Cyclic Scan." In *Proc. of the IEEE ISDFT*, page 151, Washington, DC, USA, November 2003.

[8] H. Hashempour and F. Lombardi. "Evaluation of Heuristic Techniques for Test Vector Ordering." In *Proc. of the GLSVLSI*, pages 96–99, 2004.

[9] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys. "*The Traveling Salesman Problem*." John Wiley, Chichester, 1985.

[10] N. Nicola and B. M. Al-Hashimi. "*Power-Constrained Testing of VLSI Circuits*." Kluwer Academic Publishers, 2003.

[11] K. Roy, R. K. Roy, and C. A. "Stress Testing of Combinational VLSI Circuits Using Existing Test Sets." In *Proc. Of the ISVLSI*, pages 93–98, 1995.

[12] S. J. Russell and N. Peter. "*Artificial Intelligence: A Modern Approach"*. Pearson Education, 2003.

[13] A. Sokolov, A. Sanyal, D. Whitley, and Y. Malaiya. " Dynamic Power Minimization During Combinational Circuit Testing as a Traveling Salesman Problem." In Proc. of the IEEE Congress on Evolutionary Computation, volume 2, pages 1088–1095, 2005.

[14] Seongmoon Wang, Sandeep k.Gupta, Feb. "ATPG for HEAT Dissipation Minimization During Test Application" *IEEE Trans. On computer* Vol: 47, No. 2, pp 256-262, 1998.

[15] Z Luo et. All, "Test Power Optimization Techniques for CMOS Circuits" *Proceedings of the 11th Asian Test Symposium,* 2002.

[16] K.Royand S.Prasad, "Low Power CMOS VLSI Circuit Design" *Wiely Inc*. 2000.

[17] S Roy, S Gupta, A Pal "Artificial intelligence approach to test vector reordering for dynamic power reduction during VLSI testing" *Proceedings of IEEE Region 10 Conference, TENCON 2008,* pp.1- 6

TABLE 1

COMPARISON OF SWITCHING ACTIVITY

Sr. No	ISCAS circuit	Switching in case of Ready Test Data	Switching in Hamming Distance Based Reordering	Switching in Artificial Intelligence Based Reordering	Improvement in % reduction
1	S27	23	19	10	56.52
2	S298	1595	1631	1328	16.73
3	S344	1475	1517	1244	15.66
4	S349	1602	1615	1312	18.10
5	S382	3951	4008	3595	9.01
6	S386	402	389	250	37.8
7	S400	4124	4139	3439	16.61
8	S420	3239	3241	2642	18.43
9	S444	4218	4187	3585	15.00
10	S510	754	697	536	28.9
11	S526	4523	4510	3969	12.24
12	S641	3181	3185	2674	15.93
13	S713	2590	2626	2165	16.40
14	S820	857	794	616	28.12
15	S832	886	815	653	26.29
					Avg=22.11

A design methodology for specification and performances evaluation of Network On Chip

Djamel Adrouche*, Rabah Sadoun*, Sébastien Pillement†
* Ecole Nationale Polytechnique d'Alger, Algeria
Djamel.Adrouche@gmail.com, rabah.sadoun@mail.enp.edu.dz
† University of Rennes 1, CAIRN IRISA/INRIA, France
Sebastien.Pillement@irisa.fr

***Abstract*—In modern SoC the bottleneck is the interconnection structure. Networks on Chips are good solutions in order to offer high flexibility, scalability, while maintaining performances requirements.**

Designing a NoC based circuit is difficult due to the lack in methodology including NoC parameters in the design-cycle life. Some development techniques are used to model NoC at a high level of abstraction for validation and functional verification. However, these tools rarely support simulation for performances evaluation of NoC, in terms of throughput and latencies.

In this paper, we propose a design methodology for NoC design. The NoC architecture behavior is described by using the SDL language. We then generate *SDL Agent* to evaluate the performances of the NoC in the NS2 simulator. As a proof of concept we implement and evaluate a simple protocol in SDL.

I. Introduction

IC designers faces challenges to propose an interconnection structures that are scalable, reusable and efficient. These communications platforms needs to support the integration of an increasing number of heterogeneous cores on a single chip. The Network-on-Chip (NoC) paradigm seems particularly adapted. It consists to use the non-integrated network concepts such as communication protocols and topologies, to offer flexible and distributed integrated communication structures.

Nowadays the communication structures is the bottleneck of modern System-on-Chip (SoC). This will requires that designing frameworks must address all the properties of NoC. As a NoC offer flexibility, designers must take into account the implementation costs of such a structure (area of routers, power consumption, ...). Due to the design complexity of NoC systems, the frameworks must allow early simulation and performance estimation prior to system implementation.

To make early estimation of network performances, early works proposed to reuse the design approaches and tools widely used in non-integrated networks domain. The NoC is modeled at a high level of abstraction in order to perform bandwidth and latency evaluations. unfortunately, these tools must be extended in order to be adapted to the NoC characteristics (area overhead estimation, power consumption, ...).

Formal methods are a particular kind of mathematically-based techniques for the specification, development and verification of software and hardware systems. The use of formal methods for system design is motivated by the expectation that, it is possible to formally verify essential design properties to improve the reliability and robustness of a design. However, using formal methods comes with a high timing cost and high complexity descriptions. Furthermore, some works have proposed to use formal development techniques to model the NoC, but they do not permit to integrate simulators for performances evaluation.

In this work, we propose a complete NoC design methodology (Fig. 1) that supports the specification, the implementation and the validation/simulation steps. The entry point of our methodology is a formal language, in order to produce specifications and check at a high-level any inconsistencies in it. Also, the high level model can be extended for NoC design exploration, by estimations of specific NoC characteristics. The formal specification is then interfaced with the NS2 simulator [1] for performance evaluation purpose. The NS2 tool is widely used in the non-integrated networks domain, to simulate communication protocols and to study the interoperability of heterogeneous protocols.

In this paper, we presents the NoC specification and the link with the performances evaluation. The remainder of this paper is as follows: in the next section we draw a state of the art of NoC design frameworks. In section III we present the NoC design methodology. Finally, after presentation of results in section V, some conclusions and future work are discussed.

II. Related Works

With the emergence of the NoC concept [2], researchers have realized the need in design methodologies which fulfill the NoC design style. For topology and performances evaluations, some existing network simulator such as Network simulator (NS2) was used to simulate and observe the behavior of a NoC [3] and [4]. But in this works the proposed simulation frameworks used predefined NS2 component, such as UDP protocol, which are not suited for NoC implementation. The main problem coming from the definition of NoC protocol under NS2.

Among all the proposed approach in NoC design, [5] propose a design flow that allows to automatically determine the parameters of a NOC giving application constraints. The design environment tool μSpider was developed, it consists of tools for decision and synthesizable VHDL code generation.

This design approach is dedicated to a specific NoC architecture based on TDMA bandwidth reservation technique. The ATLAS environment [6] was developed to automates several design steps. The design flow supports: NoC generation, simulation and performance evaluation based on various traffic generation. This environment is dedicated for the HERMES network a 2D mesh NoC.

Formal description techniques were studied for NoC functional verification and simulation purpose. In [7], the author gives some indications where formal methods can be used in the NoC research field. SDL has been used in [8] to create a simulator tool for performance evaluation. They have used SDL as it has the features to represent block, concurrent process, dynamic generation of process, communication channels and timers. The simulator is created to evaluate architecture options buffer size in switches and their effect on delay and packet loss. In SDL, the NoC performances evaluation, in terms of latencies and throughput is a complex task.

A number of works have proposed a designing frameworks for NoC architecture. These frameworks permits to evaluate the NoC performances throughout the design cycle to suit application's needs. The main problem is that most of these design frameworks are specific to particular NoC or implement only part of the overall framework.

III. NoC DESIGN FRAMEWORK

We propose a complete methodology (Fig. 1) independent of a particular NoC, implementing the overall design process (i.e. from specification to implementation via simulation). Our methodology is based on an SDL (Specification and Description Language [9]) specification of the NoC. SDL was chosen because it was designed for specification and validation of communication systems. All the constructs required for a NoC representation are presents in the language. The use of this formal language will enable us to make high level specification checking, to evaluate the correctness of the specification. As said above, it is quite difficult to make performance evaluations at the SDL level to obtain accurate results. We then use the NS2 simulator for this purpose.

The NS2 simulator should be extended to support the SDL specification. So, a new module (*SDL Agent*) was developed in order to define new NoC protocol in the NS2 simulator. Based on the redefinition of communication mechanism of NS2 this module permits to create in the interpreter a simulation model based on the SDL specification of the NoC.

The network infrastructure is generated from the SDL specification to represents the network's nodes in the NS2 environment. These modification permits to efficiently use the different environment. The SDL specification describe the overall NoC protocol layers (transport , network and data link layers), the communication parameters (routing protocol, communication protocol, switching mode) and the flow control. While the simulator components provide the network parameters such as topology, buffers size and links.

As NS2, do not support NoC parameters evaluation, we will introduce in the specification some models for area and power evaluation of a specific implementation of the NoC. Finally a classic SDL-to-VHDL code generator is used for implementation of the resulting network.

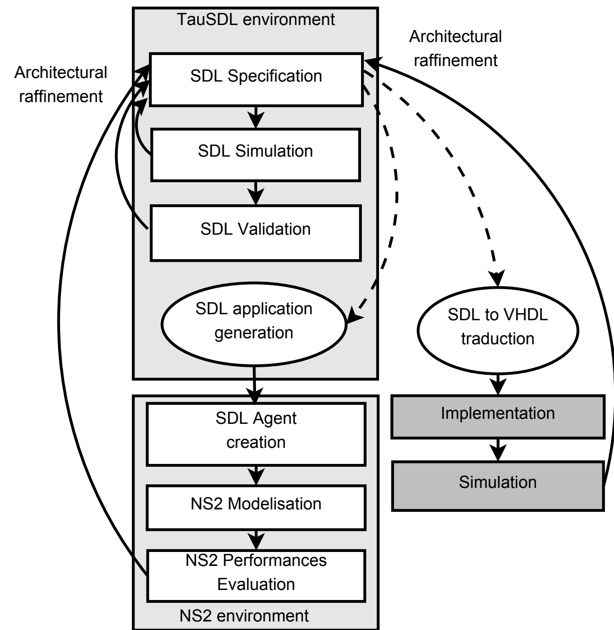


Fig. 1. Proposed NoC design flow. After the formal description of the NoC, a *SDL Agent* can be generated in order to integrate the SDL description in the NS2 environment for simulation purpose. After validation and simulation a VHDL generator is used for implementation of the NoC.

In this paper, we present the SDL specification, the functional simulation and the integration of SDL specification within the simulator to achieve some performances evaluation. The formal verification of the NoC protocol, the validation and the implementation on a target circuit are under the scope of this paper, since they are ongoing works.

A. SDL specification

In the first step of the methodology, we model the NoC at a high level of abstraction. The SDL language is well suited for describing reactive discrete systems [10], which is a good model for NoC description. For instance, a NoC is composed of switchs, that reacts on a discrete event: a packet arrival. After that it takes a decision and perform some actions in order to route the packet across the network. Due to the high level of description, it is relatively simple to build such a complex system model.

In SDL, the behavior of the system is specified using concurrent processes. Each process contains an extended finite state machine wich communicate by exchanging signals through channels (or signal routes).

A NoC protocol is constructed as several layers (transport, network, data link, etc.). Each layer providing services to higher layers of the protocol. Therefore, the NoC protocol can be represented by an SDL system which is composed of blocks interconnected by channels, as depicted in Figure 2. At the higher level of hierarchy, we find the NoC protocol

layers (transport, network, etc.). Each layer is build over blocks which implement services of the layer. Each block is composed of processes, interconnected by signals, and which describe the behavior of this block. For instance, the network layer is represented by the "network block" which contains a "routing process". This process achieve the routing algorithm.

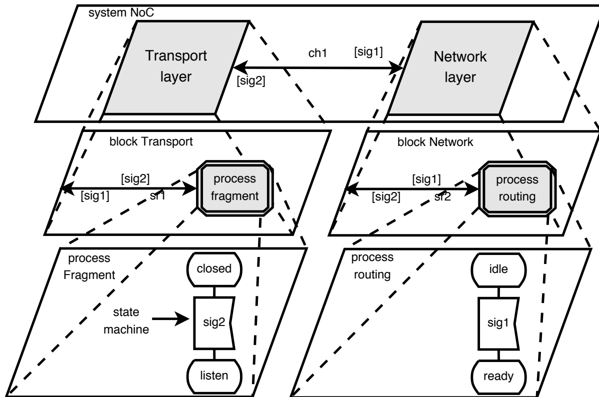


Fig. 2. In SDL, the NoC protocol is modeled as a system containing blocks. Each block may contain either blocks or processes. Each process contains an extended finite state machine. State machines communicate by exchanging signals through channels (or signal routes).

The TauSDL [11] editor is used to describe functional representation of a system hierarchy using blocks and processes. To check and debug the syntactic errors of the description, the SDL compiler is used.

B. SDL simulation

SDL simulators provide high caliber debugging features, from symbol by symbol stepping to automatic simulations using various strategies (random, exhaustive, bit-state, supertrace etc.). For the functional simulation of NoC protocol, we use the MSC (Message Sequence Charts) diagrams which are describing the requirements on the dynamic behavior. We can notice that currently only functional simulation is supported.

IV. Integration of SDL application within NS2 environment

In order to obtain precise evaluation of network performance, we needs to interface the NoC specification with the NS2 environment. This is done by the mean of generating an *SDL Agent* which represents one node of the network. The topology of the network is obtained by the mean of interconnecting several ns node, using ns links.

A. Structure of the NoC architecture in NS2

Figure 3 shows the structure of the NoC architecture which should be implemented on NS2. Each node of the network represents an *SDL Agent* which is composed of an *SDL application* and *SDL interface*. The *SDL application* is generated from the SDL specification thanks to the the Cadvanced SDL to C compiler. An *SDL application* (Fig. 3) is:

- the SDL system: which describes the SDL system behavior. This block implement the NoC protocol. It is implemented by the way of independent C processes communicating through the SDL kernel.
- the SDL Kernel: this system layer support the scheduling of SDL processes and the SDL system communications.
- the environment functions: This layer implement the interface of communication between the SDL system and the external environment, in our case, the NS2 simulator.

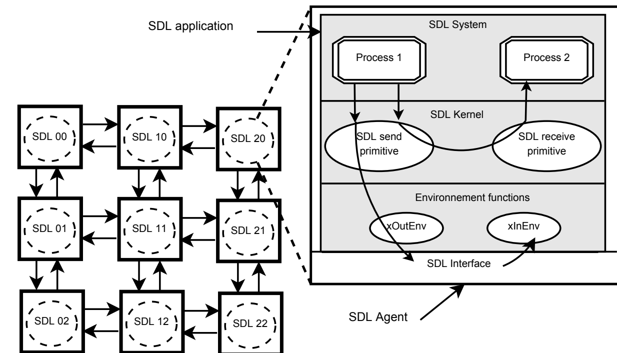


Fig. 3. Structure of a Noc architecture in NS2. Each ns node implements an *SDL Agent* which is composed of an *SDL application* and an *SDL interface*.

B. Operation of the NoC architecture in NS2

In NS2, the SDL network is described in a script file which instantiate the *SDL application* and permits to monitor the progress of the simulation by the ns scheduler.

The ns and the SDL schedulers operate according to the concept of globally-asynchronous locally-synchronous. Indeed, the two schedulers are independent. The ns scheduler ensures the progress of the overall simulation while the SDL schedule the application operations. By construction, in the *SDL Agent*, the *SDL interface* and the *SDL application* needs to operate in synchronous mode. To achieve this operation mode, we do some changes in the SDL scheduler.

As we will see the data carried out by ns components are provided to the *SDL application* through the xInEnv() function. The SDL scheduler implements an infinite loop which calls the xInEnv() function in order to extract signals sent by the environment to the *SDL applications*. We modify this behavior so that the function is activated only if a signal is sent to the *SDL application*. This activation is controlled by the *SDL interface* which notifies to the *SDL application* when it receives a packet from an ns components.

Similarly, a timer is implemented in the *SDL interface* to allow to the *SDL application* to notify that data provided through the xOutEnv() function should be transmitted through the network.

The data area must suitable to represented the signals treated by the components of the network (*SDL application*, *SDL interface*, *ns links*, etc.). Figure 4 shows the structure of data exchanged through the network. Inside an *SDL application*, SDL signals ensure transfer of data exchanged between different processes.

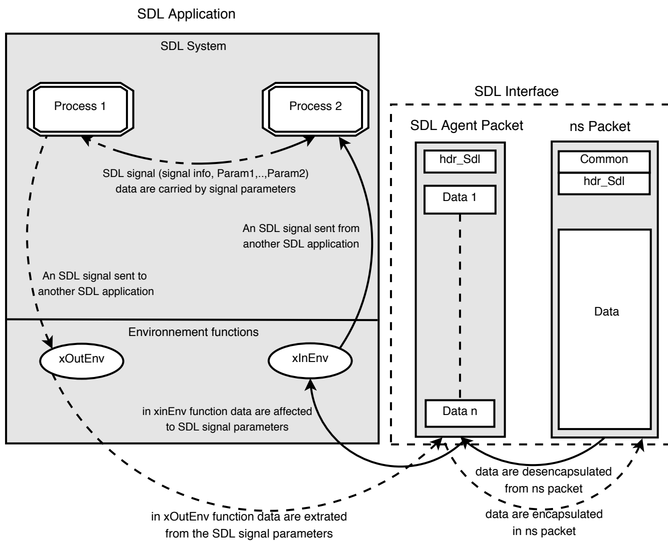


Fig. 4. Structure of data exchanged between the *SDL application* components and the *SDL interface*.

When signals are exchanged between two *SDL applications*, data are extracted from the SDL signal parameters thanks to the xOutEnv() function. Data are then passed to the *SDL interface* through temporary buffers. *SDL interface* encapsulate the data in ns packets to be transmitted over the network. Upon a packet is received by the destination node, the *SDL interface* extract data and store them in a buffer. Data are then affected to SDL signals parameters through the xInEnv() function. These functions are described in section IV-C2.

To allow the network components to identify the data type to be treated, an identifier is associated to the data exchanged. For example, the packets exchanged on ns links are composed of two fields. The first is used to carry the data. The second field identifies the data type carried by the packets.

Also, an addressing system is used to identify the nodes on the network. In fact, each node have two address. The first address identifies the ns nodes in the network. This address is provided to each ns node through the simulation script. The second address is used to identify the *SDL applications* over the SDL network. These addresses are expressed in XY, where X represents the horizontal position and Y the vertical position. Each time an *SDL application* instance is created in the simulator it received it's address through the SDL kernel. A correspondence between these two system needs to be made (an *SDL application* is mapped on a *ns node*).

All these adaptations are supported in the *SDL Agent*.

C. The SDL Agent

The *SDL Agent* (*SDL application+SDL interface*) represents the new protocol added to the NS2 environment. The key point in integrating *SDL Agent* in the simulator, rely on the correct definition of the *SDL interface*.

1) SDL interface: The *SDL interface* implements mainly the functions which allowed to insert the SDL protocol in the simulator. This interface is developed to:

- ensure a synchronous data transmission between *SDL applications* and ns components,
- allow data transfer between SDL signals and ns packets,
- locate the *SDL application* in the ns architecture.

The *SDL Agent* (representing the NoC protocol) is declared in the ns architecture as a subclass of the Agent class. This permit to inherit the functions that allow to send and to receive data exchanged between the *SDL applications*. The *SDL Agent* contains and mainly implements the *command(), send() and recv() functions* .

```
//NoC protocol packet header
struct hdr_sdl {
   unsigned int id;
   unsigned int payload;
//required method for the PacketHeaderManager to access
//NoC protocol
   packet header
   static int offset_;
   inline static int& offset() {
      return offset_;
   }
   inline static hdr_sdl* access(const Packet* p) {
   return (hdr_sdl*) p->access(offset_);
   }
};
```

Listing 1. SDL header packet declaration.

The first thinks to do, is to add the definition of SDL packet header by the way of the PacketHeaderManager to the ns packets headers. As said before in NS2, the packet header have a data structure composed of two fields. The first one called "payload" is used to transport data exchanged between the *SDL applications*. The second field called "id" is used to distinguish between different types of data. Listing 1 shows the declaration of the SDL packet header and the link with the ns packet structure.

In order to take into account *SDL Agent* a link between the simulator and the objects interpreter is required. So, when an object is created in the interpreter, an equivalent object is created in the simulator. Listing 2 shows the association of the SDL packet which is defined by the static variable class_sdlhdr of the *SdlHeaderClass*. It also shows the link of the *SdlAgentClass* object with the interpreter *TclClass* class.

```
int hdr_sdl::offset_;
static class SdlHeaderClass: public PacketHeaderClass {
   public:
   SdlHeaderClass():PacketHeaderClass("PacketHeader/Sdl",
   sizeof(hdr_sdl)) {
      bind_offset(&hdr_sdl::offset_);
   }
} class_sdlhdr;

static class SdlAgentClass : public TclClass {
   public:
   SdlClass() : TclClass("Agent/Sdl") {}
   TclObject* create(int argc, const char*const* argv) {
      return (new SdlAgent(nsaddr_t));
   }
} class_sdl;
```

Listing 2. Association between the simulator and the interpreter objects.

2) SDL application: As said above the *SDL application* is generated by the Cadvanced SDL to C compiler. We also use the Targeting Expert tool to manage and make easier the complete process of targeting. In the Targeting Expert tool, we configure the compiler and link modules to get an *SDL application* library which is compatible with the NS2 environment. As described above, this library is composed of *SDL system, SDL kernel and environment functions*. Compiled with the *SDL interface*, we obtain an *SDL Agent* which is instantiated in the ns interpreter to create the NoC architecture.

In order to enable the communication between the *SDL applications* and the ns components, we add in the environments functions the actions that should be performed when an *SDL application* send/receive signals to/from the neighboring nodes of the network. For instance, we implement actions that control the execution of the *SDL application* and the *SDL interface* timers, and permits to affect/extract data to/from the SDL signals parameters.

a) The xOutEnv() function: Each time a signal is sent from the SDL system to the environment, the function xOutEnv() is called. The xOutEnv() function will have the current signal to send as parameter, so we have all the information contained in the signals (signal type, the sending and receiving process instance and the parameters of the signal). As shown in listing 3, in the xOutEnv function, the NameNode is used to determine the signal type provided by the SDL system. For each signal type, we extract data parameters carried out by the signal and we activate the *SDL interface* timer to notify that a packet needs to be sent through the network.These information will be used to construct the ns packet by the *SDL interface*.

```
extern void xOutEnv(xSignalNode *SignalOut)
{
...
   if ((*SignalOut) -> NameNode == To_Env)
   {
   ...
   start_req = ((yPDP_To_Env)(*SignalOut)) -> Param1;
   start_rsp = ((yPDP_To_Env)(*SignalOut)) -> Param2;
   ...
   wait_ns = true;
   xReleaseSignal (SignalOut);
   return;
   }
   ...
}
```

Listing 3. Actions achieved in the xOutEnv function.This function prepare SDL signals to be transmitted as ns packet

b) The xInEnv() function: The SDL kernel implement a loop which continually call the xInEnv() function in order to see if signals are sent from the environment to the processes within the SDL systems. When a packet is received by the *SDL interface*, a notification is made that information are to be forwarded to the SDL system. The xInEnv() function implements two functions that allow to forward signals to the SDL system. We use the *xGetSignal* function to obtain a data area suitable to represent an SDL signal and the SDL_Output function to send the signal to the specific receiving process according to the semantic rules of SDL.

Listing 4 show how data are affected to a signal parameters in the xInEnv function. The xGetSignal function provides three parameters. The first parameter is a reference to the symbol table node which represent the current signal type. The second parameter represent the receiver process identifier Pid, in our case we use xNotDefPId to indicate that the signal should be sent as an output without TO clause. The last parameter gives the sender Pid value, in our case we use xEnv value to refers to an environment process instance (i.e. an ns node). In the SDL_output function we use the value (xIdNode *)0 to represent that no via list of channels is present.

```
#ifndef XTENV
   extern void xInEnv (SDL_Time Time_for_next_event)
#else
   extern SDL_Duration xInEnv (SDL_Time
        Time_for_next_event)
#endif
{
   xSignalNode SignalIn;
   ...
   SignalIn = xGetSignal (Fr_Env, xNotDefPId, xEnv);
   ((yPDP_Fr_Env)SignalIn)->Param1 = start_req;
   ((yPDP_Fr_Env)SignalIn)->Param2 = start_rsp;
   ...
   SDL_Output (SignalIn, (xIdNode *)0);
   ...
   #ifdef XTENV
   return SDL_Time_Lit((xint32)0,(xint32)0);
   #endif
}
```

Listing 4. Actions achieved in the xInEnv() function. This function permits to forward information from the ns environment to a particular SDL system.

V. Results

As a proof of concept for the overall methodology, we implement in the simulator a NoC architecture in which each network node implements the ping protocol. Even if the ping protocol is not a NoC protocol, the goal is to illustrate our design methodology and to validate the integration of an SDL specification into the NS2 environment. Therefore, we use the approach presented in Section III to model the ping protocol. As the *SDL interface* is constant for the methodology and implement the required modification, only the *SDL applications* are generated for the implementation of this “NoC”. The simulation model is described in OTcl script. The topology of the NoC (Fig. 5) is built through the instanciation of nodes and links. We built a simple NoC with 4 nodes with a mesh topology. The nodes are interconnected with bidirectional links. We generate a traffic which allow to transmit packets in cyclic manner from node 0 to node 3. The generated traffic is sufficiently significant to involves packets drop.

A periodic bandwidth record is made at each *SDL application* node. The results are shown in the Figure 6. In fact, bandwidth record and links parameters analysis is useful for design of an appropriate switch for the NoC.

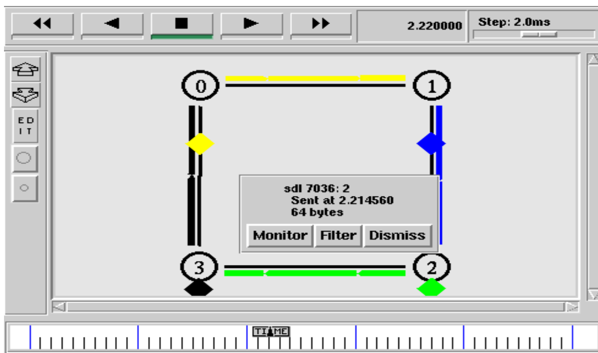


Fig. 5. The experimental ping "NoC". Four SDL systems network shown in the NAM simulator tool. The ping traffic is generated between node 0 and 3.

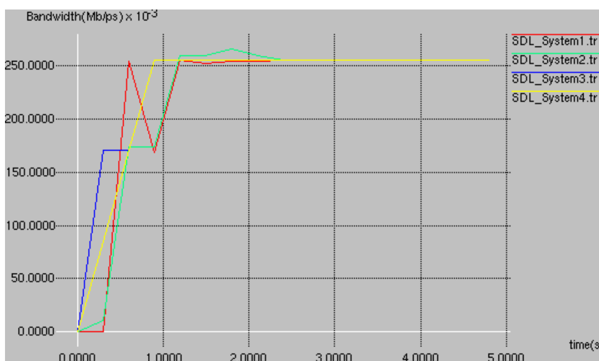


Fig. 6. Bandwidth available for each node in the NoC.

We monitor the queues implemented in the NS2 links to calculate the number of packets in the queue at a periodic time (see Figure 7). For instance, the analysis of packets loss and the amount of packets in the queue allow the choice of the optimal buffer size in switchs.

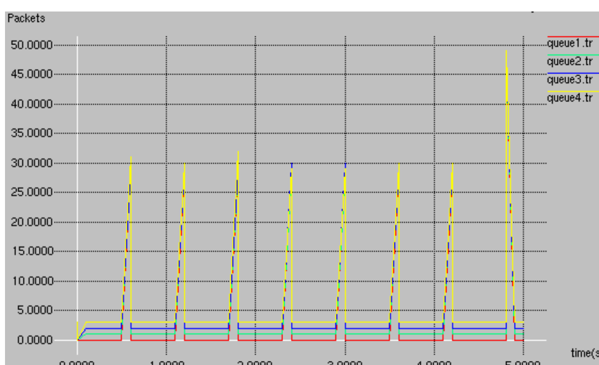


Fig. 7. Queue monitoring of each node.

The main results here is that the methodology permit to simulate a NoC described in SDL language. As all the code is generated and the interface defined it is then easy to make the exploration of NoC parameters exploration. In NS2, it is also possible to analyze the number of packets sent, received and rejected according to the network load traffic (not shown here due to space limitation) to evaluate NoC implementation (by changing the queue size for example) on the overall performance of the constituted network.

VI. Conclusion and Future Works

In this paper a NoC design methodology is presented. First, an overall design flow is described which includes the specification and validation of a NoC. The proposed flow uses the SDL language to describe the NoC protocol independently of any architecture. The integration of the SDL description in the NS2 environment for performance evaluation purpose has been presented and validated on a proof-of-concept example. Finally the NoC implementation can be made by the use of a SDL to VHDL generator.

Also, the fact that using generated code from the SDL description for performances evaluation, permit to not maintain separated code for simulation and description purposes, avoiding consistency check and error in translation.

Currently we are describing a real NoC using our methodology, to compare with real implementation results. Also, in the future, high level models of area and power consumption will be added in the specification level, to evaluate the NoC implementation costs.

References

[1] The network simulator ns2, information sciences institute, university of southern california. [Online]. Available: http://www.isi.edu/nsnam/ns

[2] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Computing Surveys (CSUR)*, vol. 38, pp. 1–51, 2006.

[3] M. Ali, M. Welzl, A. Adnan, and F. Nadeem, "Using the ns-2 network simulator for evaluating network on chips (noc)," in *Proc. International Conference on Emerging Technologies ICET '06*, 2006, pp. 506–512.

[4] Y.-R. Sun, S. Kumar, and A. Jantsch, "Simulation and evaluation for a network on chip architecture using ns-2," in *20th IEEE Norchip Conference*, novembre 2002.

[5] S. Evain, "μspider environment for network on chip design," Ph.D. dissertation, INSA de RENNES, 2006.

[6] Atlas - an environment for noc generation and evaluation. [Online]. Available: http://www.inf.pucrs.br/gaph/AtlasHtml/AtlasIndex_us.html

[7] K. Goossens, "Formal methods for networks on chips," in *ACSD05, Fifth International Conference on Application of Concurrency to System Design*, 2005, pp. 188–189.

[8] R. Holsmark, M. Högberg, and S. Kumar, "Modelling and evaluating of a network on chip architecture using sdl," in *11th SDL Forum*, juillet 2003.

[9] *SDL Methodology guidelines*, Appendice i, recommendation z.100 ed., International Telecommunication Union, March 1993.

[10] Z. Mammeri, *SDL : modelling protocols and reactive systems*. Hermes science publications, 2000.

[11] T. Telelogic Tau Generation 1. [Online]. Available: http://www.telelogic.com/products/tau/index.cfm

FOCAS: A NOVEL FRAMEWORK FOR SYSTEM-ON-CHIP DATAPATH VALIDATION

Balvinder Singh Khurana[1], Atul Gupta[2]
Networking and Multimedia Design, Freescale Semiconductors
Noida, UP, India

[1]Balvinder.Khurana@freescale.com
[2]Atulg@freescale.com

ABSTRACT **– With the enablement of the sub-micron technologies and the advent of multi-core devices, the complexity of the system-on-chip (SOC) is increasing exponentially. This technology drift is pushing the post-silicon validation to a new extreme. The challenge is to rapidly validate the entire application scenario and deliver a zero defect silicon to the customer under squeezed timelines. This paper proposes an innovative Framework for On-Chip Application data-path Stress (FOCAS) validations, with high visibility and control. It also presents the typical problems of debug and control that plague existing validation methodologies and shows how FOCAS solves these, using a novel approach. Finally, we demonstrate the effectiveness of the proposed framework, by illustrating examples and comparative data from other existing application validation techniques.**

I. INTRODUCTION

Sub-micron design challenges are pushing the traditional validation techniques to abstract the complexity of the silicon and adopt an interface based system validation approach [1], [2], [3], [4]. These aim to deliver a zero defect silicon to customers under strict timelines. Though interface based SOC abstractions does facilitate rapid validation of application data paths, high level of abstraction does impose challenges in terms of visibility and control. Keeping this goal in mind and looking at traditional validation techniques, we find them lacking in one parameter or another.

Before we benchmark the traditional techniques, let us list down the basic paradigms of an efficient SOC validation framework (see *Table I*).

With these paradigms in mind, we start looking at traditional validation techniques and evaluate each on an attribute scale (high, low and medium).

Attribute	**Validation Paradigm**
Ease of Use (E)	Test Creation Ease
Rapid Test Creation (R)	Quick Test Creation and parameterization using scripting [11]
Debug (D)	Ability to Debug ➢ Hardware ➢ Source Level Debug ➢ Run/Stop Control
SOC Abstraction(A)	Ability to Test at all SOC Abstractions ➢ Hardware ○ HAL (Hardware Abstraction Layer) ➢ Software ○ Driver Layer ○ Application Layer
Unified(U)	SOC validation with all domains of validation ➢ Directed ➢ Application
Stress (S)	Ability to create stress iterations with easy system parameterization
Infrastructure (I)	Minimal Host Infrastructure ➢ On-Chip Testing
Tests Overhead(O)	Minimal Test Compilations
Random (Rand)	Ability to run random test patterns
Efficient (EF)	Able to detect bugs
Re-Use (Re-U)	Able to Re-Use test patterns

Table I
PARADIGMS OF VALIDATION FRAMEWORK

A. *JTAG Based Testing*

IEEE 1149.1 [5] based JTAG is a serial interface available on almost all modern SOCs and which can be used to connect to a Low Level Debugger (LLD) and a Command Converter Server (CCS).

CCS-LLD [6] is a piece of software that converts high-level user tests generally written in TCL (Tool Command Language) [7] script, to low-level JTAG commands. These JTAG commands are then used to

read/write or program any register or memory connected to the SOC.

B. Random Testing

Another effective validation technique is random testing [8]. This technique makes use of high-level macro language and random library to create the tests on the host and to be run by the core on the SOC over an embedded kernel [13].

C. Linux

Linux is one of the most frequent and effective application validation technique, which uses proven software architecture, with a complex application stack and a real time kernel, to create real life scenarios, but with limited debug [13], [14].

Further, to benchmark the above validation techniques across validation paradigms *(see Table II),* we did functional validation cycle over a Communications Processor SOC for Networking Applications *(Fig.1)*.

From this comparative analysis, we illustrate that there is a dire need of an effective validation framework which should address the caveats of each of these traditional validation techniques.

With this goal in mind this paper proposes *FOCAS.*

The rest of the paper is organized as follows: In section II, we will talk about FOCAS setup and architecture. In section III, we will discuss the implementation and ease of use of the framework by means of illustrative examples. In section IV, we will discuss the implementation of the framework with experimental results from real-time application, demonstrating the effectiveness of the framework. Finally, we conclude and propose to use this framework for SOC validations.

Table II
BENCHMARKING VALIDATION TECHNIQUES
**WW – working weeks*

Attribute	JTAG [5], [6]	Random [8]	Linux [13], [14], [16]
Ease of Use (E)	HIGH All Tests coded in TCL scripting language.	MEDIUM Tests coded in High level macro language. Macro language removes overhead of drafting random assembly level test cases.	LOW Linux tool-chain, drivers, and operating system expertise required to create system scenarios.
Rapid test creation (R)	HIGH IP (Intellectual Property) blocks sanity testing completed in typical 4WW*.	HIGH Random IP block validation completed in typical 8WW*.	MEDIUM Linux partial bring-up in typical 4WW*.
Debug (D)	LOW Limited Testing done at JTAG speed.	MEDIUM Since using macro language, no source level debugs and core run/stop control.	MEDIUM Limited source debug and run/stop control. Multiple Linux process debug still under evolution.
SOC Abstraction (A)	LOW Only test creation at the Hardware Abstraction Layer (HAL) possible.	LOW HAL Abstraction and limited interface abstraction like interrupts were able to test.	MEDIUM Applications and drivers were tested for the use cases but with limited hardware visibility.
Unified (U)	LOW Did limited directed testing and were not able to create application scenarios.	LOW Only able to do IP centric random testing.	MEDIUM Application validation for SOC use cases done, but heavy overhead in test image creation, configuration and debug.
Stress (S)	LOW Were able to create the stress scenarios only on HAL Layer.	MEDIUM Only able to create IP centric random stress scenarios at HAL, but got limited while creating application scenarios, due to limited macros available to construct the applications.	MEDIUM Created the use case stress scenarios but found them hard to parameterize and debug.
Infrastructure (I)	MEDIUM Host needed to connect to JTAG and TCL interpreter.	HIGH Significant host infrastructure requirement in terms of Macros, compilers, libraries, test cases, target connect tools etc.	MEDIUM Host with platform tool-chain/cross compiler required.
Test Overhead (O)	LOW Low Overhead since TCL Tests interpretation on Host and no test compilation.	HIGH High Overhead since tests generated on Host.	MEDIUM Linux 2.6 kernel needs around 30 minutes to build on a 2Ghz Pentium class processor.
Random (Rand)	MEDIUM Limited Random TCL Libraries supported on host CCS-LLD TCL interpreter.	HIGH Full IP Random validation possible.	MEDIUM Constraint Random validation not possible.
Efficient (EF)	LOW Only obvious IP Bugs found, no system bugs.	MEDIUM High coverage for individual IP Bugs but low for system bugs.	MEDIUM High coverage on system bugs but hard to debug
Re-Use (Re-U)	HIGH Platform independent TCL tests used for IP validation.	HIGH Platform independent Random Macro tests used for IP validation.	MEDIUM Limited platform porting required, 4WW of Porting Effort typically.

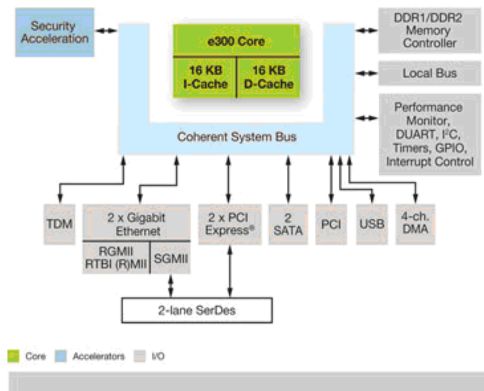


Figure 1: Communications Processor

II. FOCAS SETUP and ARCHITECTURE

The FOCAS setup (Fig. 2) consists of the following entities:

A. Hardware

1) Host – Standard host (Windows/Linux).
2) DUT – Device-Under-Test (Fig. 1).
3) Protocol Exercisers/Analyzers (Optional) - depending on the test.
4) Analog-Cz (Characterization) Setup (Optional) - Oscilloscopes, bit error rate testers etc.

B. Software

1) On HOST
   a. JTAG CCS-LLD.
   b. Serial Console.
2) On DUT
   a. FOCAS Software.

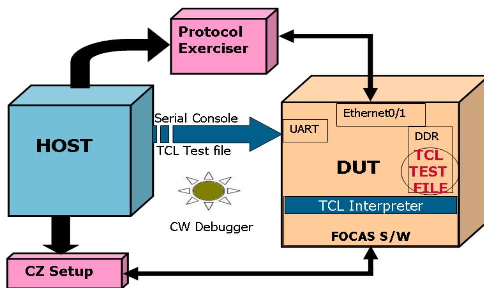


Figure 2: FOCAS Setup

C. What we propose from FOCAS:

1) The user creates an application scenario in TCL.
2) The user runs FOCAS software provided with a TCL interpreter on the DUT. The software also provides a serial console to the user to communicate with the DUT.
3) The user downloads the TCL test to the DUT memory (e.g. DDR, Flash) through the serial console.
4) FOCAS Software TCL interpreter will do On-Chip interpretation of the test case and then execute it.
5) Finally, the host can also trigger the remote Cz setups and analyzers to monitor the DUT.

D. FOCAS ARCHITECTURE

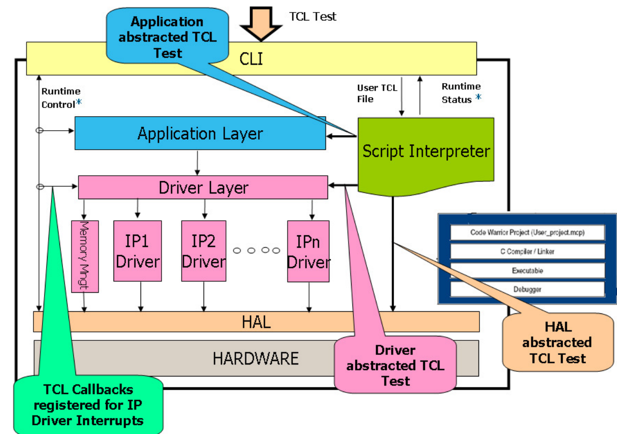


Figure 3: FOCAS Architecture

The FOCAS software architecture *(Fig. 3)* is designed as a small footprint On-Chip framework that runs a simple TCL interpreter at the application layer. Since the framework works in single software context and limited software layers, the system is easy to debug, with a capability to connect to a debugger. The FOCAS software components/layers are organized as follows:

1) Hardware Abstraction Layer (HAL)
2) Driver Layer
3) Application Layer

1) *HAL – Hardware Abstraction Layer*

HAL is closest to the hardware. It comprises of the APIs (Application-Program-Interface) [17] or software services, which interact directly with the hardware. Some of the examples of HAL APIs:

i. mm <address> <bytes> <data> - API to modify number of bytes in memory at address with the data.
ii. md <address> <bytes> - API to read number of bytes in memory at address.

2) *Driver Layer*

This layer abstracts all the peripherals of the SOC, and provides the services of configuring/control of each of those interfaces and their interrupt callbacks [14].

Some of the examples are:

i. configureDMA <base-address> - API to configure the DMA at the base address.
ii. startDMA <base-address> - API to start the DMA operation.

3) Application Layer

This layer is the top most. It comprises of the following components of the design:

i. TCL Interpreter

a. Open source thin TCL Interpreter [9], responsible for run-time interpretation of the downloaded TCL test from user. In addition to the interpretation, the interpreter provides a registration mechanism to the user to add new commands in interpreter (interp), and associates it with procedure (proc), using API Tcl_CreateCommand (Fig.4), which is a part of interpreter library tcl.h. It is done such that whenever cmdName is invoked as a TCL command in a script, the TCL interpreter will call *proc* to process the command [10], [11].
b. The advantage is that the user can identify the data-path to be stressed and then can, register the required services from any software layer to the interpreter, to create a FOCAS software image. The image is then loaded to the on-chip memory, for real-time interpretation of those services, which are now a part of the TCL tests.
c. Additionally, the user can register TCL callback routines to be executed whenever a peripheral interrupt occurs.
d. Registration mechanism also allows the system configuration and driver configuration APIs to be registered to the interpreter, facilitating the test parameterization through TCL without compilation. This approach has also been used in [12].

In this way, FOCAS provides a framework to the user to rapidly code and parameterize a test case at any SOC abstraction level, viz. HAL, driver layer or system configuration layer. The examples mentioned in Section III illustrate the usage.

ii. Serial Command Line Interface (CLI) (Application Abstracted UART Driver)

a. UART CLI forms the interface to the user.
b. User downloads and controls the execution of TCL test from the CLI.
c. CLI is also used for debug prints and to provide IP statistics information to the user during run-time.

Finally, this FOCAS DUT software is maintained as Freescale CodeWarrior IDE (Integrated-Developer-Environment), to be executed on POR (power-on-reset) from on-chip memory and attached to any third party debugger.

Tcl_CreateCommand (interp, cmdName, proc, clientData, deleteProc)
Arguments
Tcl_Interp ***interp** (in)
Interpreter in which to create new command.
char ***cmdName** (in)
Name of the command.
Tcl_CmdProc ***proc** (in)
Implementation of new command: proc will be called whenever cmdName is invoked as a command.

```
typedef int Tcl_CmdProc(
    ClientData clientData,
    Tcl_Interp *interp,
    int argc,
    char *argv[]);
```

When proc is invoked the clientData and interp parameters will be copied to the clientData and interp arguments given to **Tcl_CreateCommand**. Typically, clientData points to an application-specific data structure that describes what to do when the command procedure is invoked. Argc and argv describe the arguments to the command, argc giving the number of arguments (including the command name) and argv giving the values of the arguments as strings.

ClientData **clientData** (in)
Arbitrary one-word value to pass to proc

Tcl_CmdDeleteProc ***deleteProc** (in)
Procedure to call before cmdName is deleted from the interpreter; allows for command-specific cleanup. If NULL, then no procedure is called before the command is deleted.

Example Tcl_CreateCommand
► Tcl_createCommand(interp,"ETSEC_init",Tcl_ETSEC_init,NULL, NULL);

Example Tcl_ETSEC_init
► Tcl_ETSEC_init(Tcl_Interp *interp, int argc, Tcl_Obj *const *argv)

Example Tcl script using ETSEC_init

```
Puts "Initiasing eTSEC controller"
ETSEC_init 1
exit
```

Figure 4: Command Registration Mechanism

III. FOCAS ILLUSTRATIONS

In this section, we substantiate the FOCAS claims of an effective validation framework using illustrative examples, and list FOCAS results for each of the attributes against Table 1.

Example 1

Example 1 is a simple directed test case coded in TCL to test the Ethernet Controller Interface in the loopback mode, as shown in *fig. 5.*

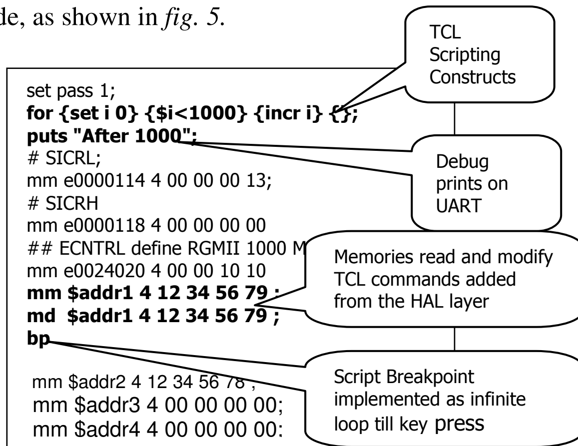


Figure 5: Ethernet Controller Directed Test with HAL Abstraction

Validation Paradigms Results:

i. E, R – HIGH, Test coded in TCL script using the TCL scripting constructs (for, If etc.) and Ethernet data-path coded in 2 days.
ii. A – HIGH, Ethernet data-path validated at HAL Layer.
iii. D – HIGH, Script based breakpoints possible.
iv. Re-U – HIGH, JTAG tests coded in TCL can now run by core.

C. *Example 2*

Example 2 is a driver abstracted Ethernet Controller TCL test, as shown in Fig. 6.

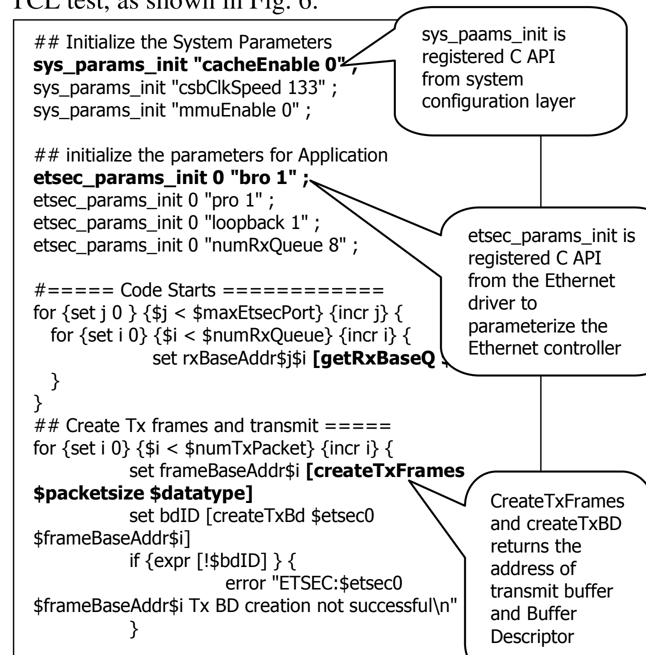


Figure 6: Driver Abstracted Ethernet Controller Test

Validation Paradigms Results:

i. E, A, R, S,RAND – HIGH
   a. Ethernet controller TCL data-path created using driver layer APIs in just 3 days.
   b. System and Ethernet IP configuration APIs registered to the interpreter aided in easy rapid creation of stress scenarios with random configurations. Advantages of this approach are listed in [7].

D. *Example 3*

Example 3 is a DMA test with interrupt call-back. Figure 7 shows the driver DMA test with callbacks.

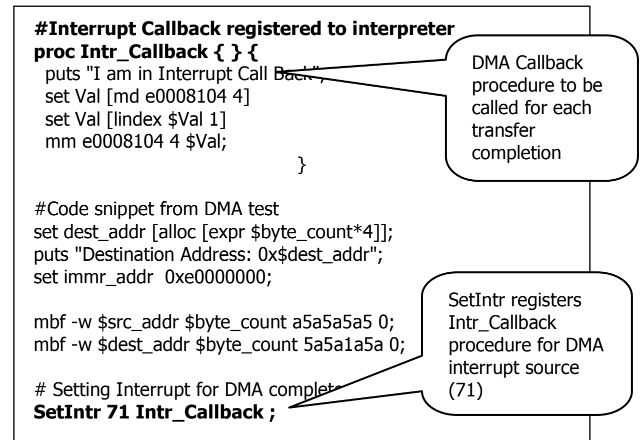


Figure 7: Driver Abstracted DMA Test with Callbacks

Inferences:

i. Peripheral Interrupt callbacks can be coded in TCL, by using the interpreter registered interrupt controller driver API (setintr <intr_source> <callback procedure>), used by DMA interrupt callback for transfer completion.
ii. Any peripheral interrupt can be mapped to a TCL procedure, such that whenever that interrupt is triggered, the respective TCL procedure will get executed.

IV. FOCAS EXPERIMENTS

A. *Example 4*

Example 4 is a L2 Bridge Application data-path on FOCAS. Fig. 8 shows the L2 bridge test setup.

Test Setup:

i. Example SOC [Fig. 1] with Ethernet application layer registered to FOCAS Interpreter.

ii. Ethernet Exerciser connected to Ethernet port 1 and 2. It is used to monitor and exercise traffic and bridge data.

iii. Application abstracted TCL test loaded on the DUT (Fig. 9).

iv. IP Parameterization control in TCL

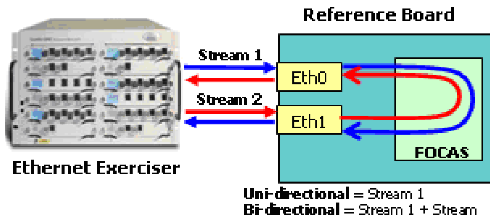


Figure 8: L2 Bridge Test Setup

```
## initialize the parameters for Ethernet controller
etsec_params_init 0 "bro 1" ;
etsec_params_init 0 "pro 1" ;
etsec_params_init 0 "loopback 1" ;
etsec_params_init 0 "numRxQueue 8" ;

#System Configuration part
 ----
 ---
#Application abstracted code .......
 while {1} {
  for {set i 0} {$i < $numRxQueue0} {incr i
   set nxtBd [lindex $nxtUseRxBd0 $i];
   if { [frameRcved $nxtBd $etsec0 $i] } {
     # Get the Baseaddr of Tx queue
     set status [createTxBd $i $etsec1] ;
     ETSEC_TxResumeAfterHalt $etsec1 ;
     #bp;
   }

   if { [frameTxed $etsec1 $i] } {
     #puts "eTSEC 1: Frame TXed";
     #bp
   }
```

Ethernet IP parameterization in control of TCL

Ethernet Application abstracted TCL code

Figure 9: L2 Bridge Test

Validation Paradigms Results:

I. E, A, R, S, RAND, D – HIGH
   a. L2 bridge application coded in TCL in just 2 days.
   b. L2 bridge application stressed with random parameters.
   c. High visibility, since debugger connected for source level debugging and TCL breakpoints and print messages helps further in debug tracing and control.

II. I, O – LOW
   a. Host only required for hosting the serial console and debugger.
   b. No test compile overhead due to on-chip test interpretation.

III. EF – HIGH
   a. Performance at par with Linux.
   b. Multiple data path exercised in single TCL - Ethernet Controller 0, Ethernet Controller 1, DMA, and UART.

Fig. 10 and Fig. 11 show the Performance Benchmarking Results of FOCAS vs. Linux [15].

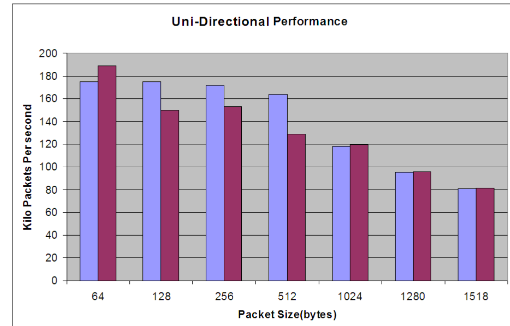


Figure 10: Uni-Directional Performance

Linux Performance
FOCAS Performance

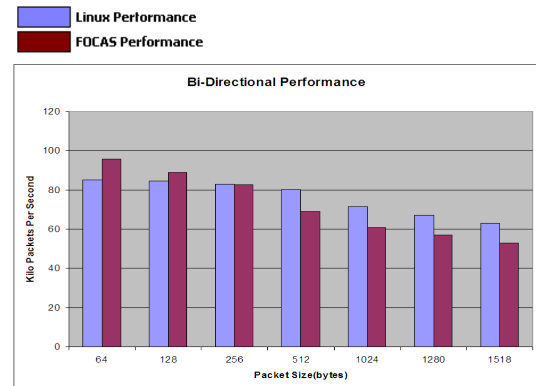


Figure 11: Bi-Directional Performance

V. CONCLUSIONS

In this paper, we have first discussed the paradigms of an ideal validation framework. Existing validation methodologies were then rated against each of these parameters, and were found to be inadequate in covering all the validation aspects satisfactorily.

To account for these insufficiencies, we proposed an innovative generic Framework for On-Chip Application Stress (FOCAS), which facilitates rapid application data path creation and control, with very high debug and observability. FOCAS setup and architecture is discussed in detail with examples to illustrate the effectiveness of FOCAS over other methodologies. Finally, experimental results have been

provided to demonstrate the effectiveness of the claims made.

REFERENCES

[1] Debashis Panigrahi, Clark N. Taylor and Sujit Dey, "*Interface based Hardware/Software Validation of a System-on-Chip*," High-Level Design Validation and Test Workshop, 2000. Proceedings, IEEE International.

[2] J. A. Rowson and A. Sangiovanni-Vincentelli, "*Interface-Based Design*", Proc. of Design Automation Conference, 1997.

[3] Kirovski, D.; Potkonjak, M.; Guerra, L.M,"*Improving the observability and controllability of datapaths for emulation- based debugging*," Transactions on Computer-Aided-Design of Integrated Circuit and Systems, Vol. 18, No. 11, November 1999.

[4] D. Kirovski, M. Potkonjak, and L. M. Guerra, "*Functional Debugging of systems-on-chip*," Conf, Computer-Aided-Design, 1998.

[5] Be Van Ngo, Peter Law, Antony Sparks,"*Use of JTAG Boundary- Scan for Testing Electronic Circuit Boards and Systems*,"AUTOTESTCON, 2008 IEEE.

[6] Universal Command Converter User's Manual [Online], Avaliable: http://irtfweb.ifa.hawaii.edu/~m2/tony/DSP56000_ref/doc/cconvert/ucc70um.pdf

[7] John K. Ousterhout, "*Tcl: An Embeddable Command Language,*" 1990, [Online], Available: htttp://citeseerx.ist.psu.edu/viewdoc/summary?/doi=10.1.1.38.820

[8] Duran, Joe W.; Ntafos, Simeon C, "*An Evaluation of Random Testing*,"Software Engineering, IEEE Transactions on Software Engineering, Issue Date: July 1984, Volume: SE-10, Issue: 4.

[9] The Jim Interpreter [Online], Available: http://jim.berlios.de/

[10] C Programming and Tcl [Online], Available: http://www.beedub.com/book/3rd/Cprogint.pdf.

[11] John Menges Mark, Mark Parris, "*Tcl and Tk Use in the Artifact Based - Collaboration-System*", [Online], Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.289

[12] Fahri Basegmez, "*Extending a Scientific Application using Scripting Capabilities*", Computing in Science Engineering, Issue Date: Nov/Dec 2002, Volume: 4, Issue: 6.

[13] Greg Kroah-Hartman, Linux Kernel in a Nutshell. O'Reilly Media, December 2006.

[14] Alessandro Rubini, Linux Device Drivers, O'Reilly Media, February 1998.

[15] J. T. Yu: "*Performance Evaluation on Linux Bridge*", Telecommunications System Management Conference 2004, Louisville, Kentucky (April 2004).

[16] Advantages and Disadvantages of Linux [Online], Available: http://www.rtcubed.com/consulting/linux-advantages-disadvantages.html

[17] Aldred, B.K., Bonsall, G.W., Lambert, H.S., Mitchell H.D. , "An application programming interface for collaborative networking", Telecommunications, 1993.

Synthesis of Reversible Logic Circuit using Unitary Matrix

Bikromadittya Mondal
B. P. Poddar Institute of Management and Technology, Kolkata, W.B., India, (E-mail: bikmondal@gmail.com)
Pradyut Sarkar
Simplex Infrastructures Limited, Kolkata, W.B., India, (E-mail: pradyut_sarkar77@yahoo.com)
Susanta Chakraborty
Bengal Engineering and Science University, Shibpur, Howrah, W.B., India, (E-mail: susanta_chak@yahoo.co.in)
(Communicating author and supervisor)

Abstract

Reversible logic plays a significant role in the low power circuit design, quantum computing and nanotechnology. Synthesis of reversible logic circuit is a complex and challenging problem. The paper proposes a novel synthesis technique of a reversible logic circuit based on unitary matrix. To synthesis the reversible logic circuit, the unitary matrix of the functional specification is mapped to an identity matrix by performing circular shift operations. The proposed heuristic algorithm performs circular shift operations in each row of the unitary matrix. The unitary matrices generated by the algorithm are replaced by equivalent reversible logic gates if a suitable match is found. Otherwise, the synthesis technique is applied until the final circuit is constructed. Experimental results on reversible logic circuits show effectiveness of the proposed technique.

Keywords - *Synthesis*, r*eversible logic gate, unitary matrix, identity matrix, quantum computing.*

1. Introduction

Reversible circuits can be focused as a special case of quantum circuits because quantum computations are inherently reversible in nature. Quantum circuits can solve some problems like prime factorization, exponentially faster compared to non-quantum (classical) approaches. Quantum circuits store information in microscopic states and process it using quantum mechanical operations or "gates" that modify these states. The unit of quantum information is called a ***qubit.*** A qubit can be in a zero or a one state, conventionally denoted by $|0>$ and $|1>$ respectively. However, it can also be in a superposition of these states, i.e., $\infty_0|0> + \infty_1|1>$, where ∞_0 and ∞_1 are complex numbers called amplitudes.

A circuit or gate is said to be reversible if the (Boolean) function it computes is bijective i.e., there is a one-to-one and onto correspondence between its input and output. The input vector can be uniquely determined from the output vector and vice versa. A necessary condition is that the circuit has the same number of input and output wires. Functions of most of the classical logic gates like AND, OR, EXOR are not reversible. Controlled–NOT (CNOT) gates proposed by Feynman[7], Toffoli gates[10], and Fredkin[8] gates are well known reversible logic gates that are needed to design reversible logic circuits.

Recently many algorithms have been proposed by several authors to synthesize reversible logic circuit. Toffoli [14] proposed CNOT based gates synthesis algorithm. Local transformation based algorithm is presented in [13]. Miller [14] uses Rademacher-Walsh spectral and two-place decompositions techniques to synthesize a reversible logic circuit. A new heuristic algorithm [15] uses binary decision diagram to represent reversible logic circuit. Shende et al in [16] proposed a number of techniques to synthesize optimal and near-optimal reversible logic circuits that require little or no temporary storage. Saeedi et al, [18] investigated the behavior of substitution-based techniques to synthesize reversible logic circuit and proposed a new hybrid DFS/BFS synthesis algorithm. Gupta et al, [12] proposed an algorithm that uses the positive-polarity Reed–Muller expansion of a reversible function to synthesize the function as a network of Toffoli gates. The algorithm utilizes a priority based search tree and heuristics are used to rapidly prune the search space.

This paper addresses the synthesis of reversible logic circuit using unitary matrix. The unitary matrix of the reversible circuit is mapped to an identity matrix by performing circular shift operations. A heuristic algorithm is proposed to construct the final reversible logic circuit by the equivalent reversible logic gates of the unitary matrices.

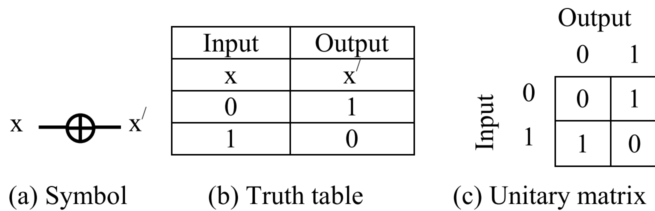
2. Preliminaries

2.1 Unitary Matrix

A **unitary matrix** is a square matrix U whose entries are complex numbers and whose inverse is equal to its conjugate transpose U^*. This means that $U^*U = UU^* = I$, where U^* is the conjugate-transpose of U and I is the identity matrix. A unitary matrix in which all entries are real is the same thing as an orthogonal matrix. Just as an orthogonal matrix G preserves the (real) inner product of two real vectors, thus $<G\mathbf{x}, G\mathbf{y}> = <\mathbf{x}, \mathbf{y}>$, so also a unitary matrix U satisfies $<U\mathbf{x}, U\mathbf{y}> = <\mathbf{x}, \mathbf{y}>$ for all *complex* vectors x and y, where $<.,.>$ stands now for the standard inner product on $\mathbf{C}^n$. A matrix is unitary if and only if its columns form an orthonormal basis of $\mathbf{C}^n$ with respect to this inner product. Reversible logic gates and circuits are represented by unitary matrices. Unitary matrices of reversible logic gates and reversible circuits are presented in section 2.2 and 2.3 respectively.

2.2 Reversible logic gates

The NOT gate shown in Fig.1 is a one input one output gate. It inverts the input. The CNOT gate shown in Fig.2 is a 2x2 gate. The value at the first input is left unchanged, and the value on the second input is inverted if and only if the value at the first input is 1, else remains unchanged. The Toffoli gate (2-CNOT) shown in Fig.3 is a 3x3 gate. It passes the first two inputs through and inverts third if the first two are both 1, else remain unchanged.

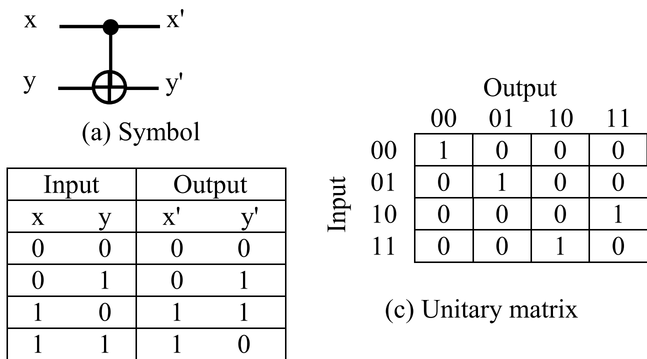


Input	Output
x	x/
0	1
1	0

Input \ Output	0	1
0	0	1
1	1	0

(a) Symbol (b) Truth table (c) Unitary matrix

Fig.1: NOT gate representation



(a) Symbol

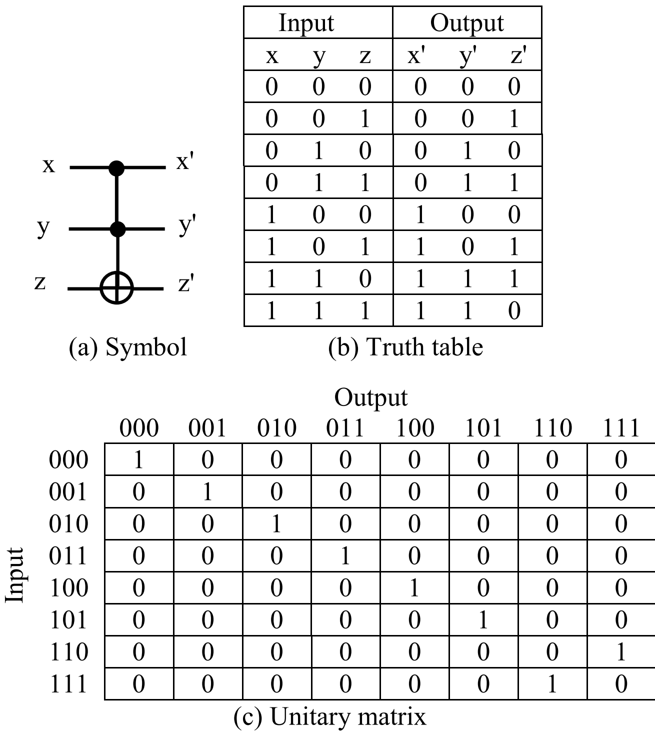
Input		Output	
x	y	x'	y'
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Input \ Output	00	01	10	11
00	1	0	0	0
01	0	1	0	0
10	0	0	0	1
11	0	0	1	0

(c) Unitary matrix

(b) Truth table

Fig.2: CNOT gate representation



Input			Output		
x	y	z	x'	y'	z'
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

(a) Symbol (b) Truth table

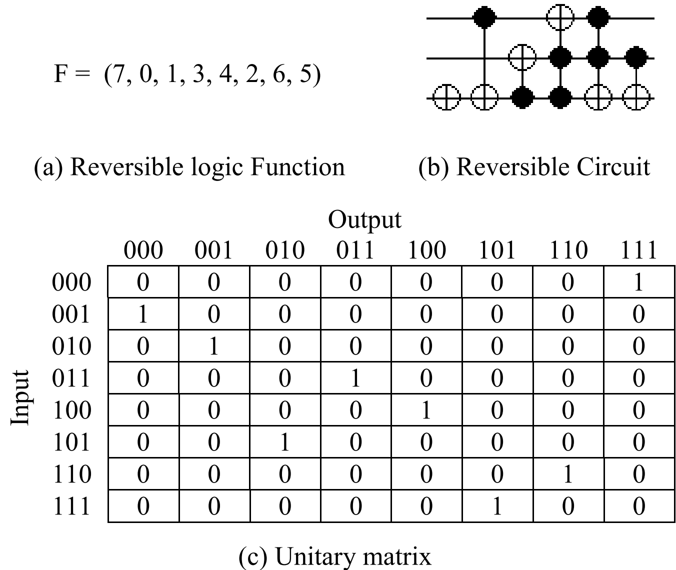
Input \ Output	000	001	010	011	100	101	110	111
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	0	0	0	1	0	0
110	0	0	0	0	0	0	0	1
111	0	0	0	0	0	0	1	0

(c) Unitary matrix

Fig.3: Toffoli (2-CNOT) gate representation

2.3 Reversible Logic Circuit

A reversible circuit and it's unitary matrix is shown in Fig 4(b) and 4(c) respectably. The Unitary matrix of a reversible circuit is directly obtained from the output function of the circuit or multiplying the unitary matrices of the reversible logic gates present in the circuit.



F = (7, 0, 1, 3, 4, 2, 6, 5)

(a) Reversible logic Function (b) Reversible Circuit

Input \ Output	000	001	010	011	100	101	110	111
000	0	0	0	0	0	0	0	1
001	1	0	0	0	0	0	0	0
010	0	1	0	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	1	0	0	0	0	0
110	0	0	0	0	0	0	1	0
111	0	0	0	0	0	1	0	0

(c) Unitary matrix

Fig.4: Unitary matrix of a 3-bit reversible circuit

3. Proposed Synthesis Technique

We propose unitary matrix based synthesis of reversible logic circuit. The unitary matrix (OM) of order n of a reversible functional specification (q bits) is mapped to an identity matrix by shifting of '1'. If the unitary matrix is matched with a reversible logic gate then it is replaced by that reversible gate. Otherwise, the number of circular right shift of '1' is counted for each row of the unitary matrix with respect to an identity matrix. If the number of shift $|S| \neq 2^m$ (where $0 \leq m < q$) and $|S| = (n - 2^p - p)$ (where $0 \leq p < q - 1$) then the row is shifted to 2^p positions. Suppose for a unitary matrix of order 8 of a 3-bit reversible logic circuit, if the number of shift in a row is 7 then $7 = 8-2^0-0$ and the position of '1' is shifted 2^0 (=1) position. Similarly, if the number of shift in a row is 5 then $5 = 8-2^1-1$ and the position of '1' is shifted 2^1 (= 2) positions. In other case, if the number of shift $|S| \neq 2^m$ (where $0 \leq m < q$), it is shifted to 2^{q-1} positions. Due to the shift operations all the other affected rows are adjusted to maintain the unitary matrix property. The amount of shifting is decided such that it can be realized by equivalent reversible logic gates. Number of rows in the unitary matrix of a k-CNOT reversible logic gate differ by $2^{\eta-k}$ with that of an identity matrix and number of circular shifts in a row are 2^γ, where η= number of bits, k = number of control bits and γ is the binary position (from right) of the target bit(line).The unitary matrix thus generated by the shifting process is named as NM. Back-tracing on the OM and NM generates another unitary matrix named as BM. If '1' is found in the (i^{th} row & j^{th} column) in the NM and in the (j^{th} row and k^{th} column) in the OM, place '1' in the ith row and k^{th} column in the BM.

The BM is compared with the unitary matrices of reversible logic gates. The total number of reversible logic gates are $m({}^{m-1}C_0 + {}^{m-1}C_1 + {}^{m-1}C_2 + \ldots\ldots\ldots + {}^{m-1}C_{m-1}) \Rightarrow m2^{m-1}$, where n= order of the unitary matrix & m = $\log_2 n$. If a match is found then the BM is replaced by the reversible logic gate. Otherwise, repeat the synthesis technique on the BM until a suitable matching is found. Then apply similar matching and synthesis technique on the NM.

3.1 Proposed Algorithm

1. Read the unitary matrix (OM) of order n of a q bit reversible logic circuit
2. For each row count the number of circular right shift with respect to identity matrix and let it be S.

   If $|S| \neq 2^m$ $(0 \leq m < q)$
       If $|S| = n - 2^p - p$ $(0 \leq p < q - 1)$
           Shift 2^p positions
       Else
           Shift 2^{q-1} positions
       Endif
   Endif
   The new unitary matrix thus generated is named NM
3. Generate another unitary matrix (BM) from the OM and NM by back-tracing.
4. If BM matches with a reversible logic gate

   Replace it by the reversible logic gate
   Else
   Apply the synthesis technique on BM
   Endif
5. If NM matches with a reversible logic gate

   Replace it by the reversible logic gate
   Else
   Apply the synthesis technique on NM

   Endif

Example: The given reversible logic function is shown in Fig.5(a) and the corresponding unitary matrix is in Fig.5(b).

F (a, b, c) = (0, 1, 2, 3, 4, 6, 5, 7)

Fig.5(a): Reversible logic function

1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	1

Fig.5(b): Unitary matrix

1	0	0	0	0	0	0	0	→0
0	1	0	0	0	0	0	0	→0
0	0	1	0	0	0	0	0	→0
0	0	0	1	0	0	0	0	→0
0	0	0	0	1	0	0	0	→0
0	0	0	0	0	0	1	0	→7
0	0	0	0	0	1	0	0	→1
0	0	0	0	0	0	0	1	→0

Fig.5(c): OM

1	0	0	0	0	0	0	0	→0
0	1	0	0	0	0	0	0	→0
0	0	1	0	0	0	0	0	→0
0	0	0	1	0	0	0	0	→0
0	0	0	0	1	0	0	0	→0
0	0	0	0	0	0	0	1	→6
0	0	0	0	0	1	0	0	→1
0	0	0	0	0	0	1	0	→1

Fig.5(d): NM

The right hand side in Fig.5(c) shows the number of circular right shift in each row with respect to the identity matrix. Shifting operation is performed according to the expression shown in step 2 and the NM is shown in Fig.5(d).

1	0	0	0	0	0	0	0	→0
0	1	0	0	0	0	0	0	→0
0	0	1	0	0	0	0	0	→0
0	0	0	1	0	0	0	0	→0
0	0	0	0	1	0	0	0	→0
0	0	0	0	0	1	0	0	→0
0	0	0	0	0	0	0	1	→7
0	0	0	0	0	0	1	0	→1

≡

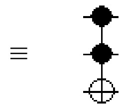


Fig.5(e): BM Fig.5(f): Reversible logic gate

BM (Fig.5(e)) is generated by applying back-tracing on the two matrices (Fig.5(c) and Fig.5(d)). The BM is replaced by equivalent reversible logic gate shown in Fig.5(f). The matrix in Fig.5(d) is not matched with reversible logic gate and further synthesis generates NM shown in Fig.5(g) and then back-tracing generates BM shown in Fig.5(i). The unitary matrices in Fig.5(g) and Fig.5(i) are equivalent to reversible logic gates shown in Fig.5(h) and Fig.5(j) respectively and no further synthesis is required.

1	0	0	0	0	0	0	0	→0
0	1	0	0	0	0	0	0	→0
0	0	1	0	0	0	0	0	→0
0	0	0	1	0	0	0	0	→0
0	0	0	0	1	0	0	0	→0
0	0	0	0	0	1	0	0	→0
0	0	0	0	0	0	0	1	→7
0	0	0	0	0	0	1	0	→1

≡

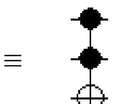


Fig.5(g): Updated NM Fig.5(h): Reversible logic gate

1	0	0	0	0	0	0	0	→0
0	1	0	0	0	0	0	0	→0
0	0	1	0	0	0	0	0	→0
0	0	0	1	0	0	0	0	→0
0	0	0	0	1	0	0	0	→0
0	0	0	0	0	0	0	1	→6
0	0	0	0	0	0	1	0	→0
0	0	0	0	0	1	0	0	→2

≡

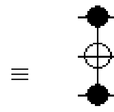


Fig.5(i): Updated BM Fig.5(j): Reversible logic gate

The final reversible logic circuit is constructed by putting the reversible logic gates in reverse order as shown in Fig.6.

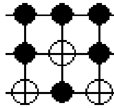


Fig.6: Reversible logic circuit after synthesis

Lemma 1: Number of rows in the unitary matrix of a k-CNOT reversible logic gate differ by $2^{\eta-k}$ with that of an identity matrix, where η= number of bits and k = number of control bits.

Proof: Let η be the number of bits in the reversible logic gate. So, the number of control bits(k) will be $0 \le k < \eta$. There will be 2^{η} rows in the unitary matrix of the reversible logic gate. Each row corresponds to an input vector. If 0 or no control bit is involved, all the rows will be affected. If a single control bit is involved, the control bit is 1 in half of the input vectors and half of the rows will be affected. If two control bits are involved, both the control bits are 1 in one fourth of the input vectors and one-fourth of rows will be affected. Mathematically we can show

If k=0, number of rows affected = $2^{\eta}/2^{0} \Rightarrow 2^{\eta-0}$

If k=1, number of rows affected = $2^{\eta}/2^{1} \Rightarrow 2^{\eta-1}$

If k=2, number of rows affected = $2^{\eta}/2^{2} \Rightarrow 2^{\eta-2}$

…………………………..

If c= η-1, number of rows affected = $2^{\eta}/2^{\eta-1} \Rightarrow 2^{1}$

So, number of rows differ = $2^{\eta-k}$, where $0 \le k < \eta$. □

Lemma 2: Number of circular shifts in a row of the unitary matrix (compared to an identity matrix) of a k-CNOT reversible logic gate are 2^{γ}, where γ is the binary position (from right) of the target bit(line).

Proof: In a k-CNOT reversible logic gate, a single bit is changed for any input irrespective of the value of k. If γ is the target line then only the bit in the γth position will be changed and other bits remain unchanged. The changes are shown as follows

$$x_{n-1}x_{n-2}\ldots\ldots x_{\gamma}\ldots\ldots x_1x_0 \rightarrow x_{n-1}x_{n-2}\ldots\ldots x_{\gamma}'\ldots\ldots x_1x_0$$

Therefore,

$$|x_{n-1}x_{n-2}\ldots\ldots x_{\gamma}\ldots\ldots x_1x_0 - x_{n-1}x_{n-2}\ldots\ldots x_{\gamma}'\ldots\ldots x_1x_0| = 2^{\gamma}$$

Hence, the numbers of shifts in a row of the unitary matrix of a reversible logic gate are 2^{γ}. □

4. Experimental Results

The proposed algorithm is implemented and applied to reversible logic circuits. Experimental results are shown in Table 1. The 2nd column of the table represents the functional specifications of reversible logic circuits. The 3rd, 4th and 5th column shows the number of k-CNOT gates, quantum cost and circuit representation respectively.

Table 1

Circuit	Specification	Number of Gates	Quantum Cost	Circuit Representation
1	(7, 0, 1, 2, 3, 4, 5, 6)	3	5	$(f_3), (f_3, f_2), (f_3, f_2, f_1)$
2	(0, 1, 2, 3, 4, 6, 5, 7)	3	15	$(f_1, f_2, f_3), (f_1, f_3, f_2), (f_1, f_2, f_3)$
3	(0, 1, 2, 4, 3, 5, 6, 7)	5	13	$(f_1, f_3), (f_1, f_3, f_2), (f_2, f_3, f_1), (f_1, f_3, f_2), (f_1, f_3)$
4	(0, 1, 7, 6, 2, 3, 5, 4, 10, 11, 13, 12, 8, 9, 15, 14)	6	6	$(f_1, f_2), (f_3, f_4), (f_2, f_3), (f_3, f_2), (f_2, f_1), (f_3, f_4)$
5	(1, 2, 3, 4, 5, 6, 7, 0)	3	5	$(f_2, f_3, f_1), (f_3, f_2), (f_3)$
6	(6, 3, 4, 1, 0, 5, 2, 7)	3	3	$(f_1), (f_1, f_2), (f_3, f_1)$
7	(0, 1, 7, 6, 12, 13, 11, 10, 8, 9, 15, 14, 4, 5, 3, 2)	3	3	$(f_2, f_1), (f_3, f_4), (f_3, f_2)$
8	(0, 1, 7, 6, 4, 3, 5, 2)	3	7	$(f_2, f_3), (f_1, f_3, f_2), (f_2, f_1)$
9	(0, 5, 7, 2, 3, 6, 4, 1)	3	3	$(f_1, f_2), (f_2, f_3), (f_3, f_1)$
10	(0, 1, 3, 6, 5, 7, 2, 4)	4	12	$(f_1, f_3, f_2), (f_2, f_1), (f_1, f_3), (f_2, f_3, f_1)$
11	(3, 2, 0, 1, 5, 6, 4, 7)	4	8	$(f_3), (f_2, f_3), (f_2), (f_1, f_3, f_2)$
12	(2, 1, 0, 3, 5, 6, 4, 7)	3	7	$(f_2), (f_1, f_2, f_3), (f_3, f_2)$
13	(7, 0, 1, 3, 4, 2, 6, 5)	6	10	$(f_3), (f_1, f_3), (f_3, f_2), (f_3, f_2, f_1), (f_2, f_3), (f_1, f_2, f_3)$
14	(0, 1, 3, 2, 6, 4, 5, 7)	3	7	$(f_1, f_3, f_2), (f_2, f_3), (f_1, f_2)$
15	(0, 2, 1, 4, 7, 5, 6, 3)	5	7	$(f_2, f_3, f_1), (f_3, f_2), (f_2, f_3), (f_1, f_3), (f_3, f_2)$

5. Conclusions

The paper proposes a novel synthesis technique of reversible logic circuit based on unitary matrix. The unitary matrix of a reversible circuit is mapped to an identity matrix. Finally the reversible logic circuit is constructed by the reversible logic gates of equivalent unitary matrices. Experimental results are provided to support our proposed technique. This work may be extended to the fault detection and testing of reversible logic circuit based on unitary matrix.

References

[1] R. Landauer, "Irreversibility and Heat Generation in the Computing Process," IBM Journal, vol. 5, pp. 183-191, July 1961.

[2] C. Bennett, "Logical Reversibility of Computation," IBM Journal, vol. 17(6), pp. 525-532, November 1973.

[3] G. Schrom, "Ultra-Low-Power CMOS Technology," PhD Thesis, Technischen Universitat Wien, June 1998.

[4] E. Knill, R. Laamme, and G. J. Milburn, "A Scheme for Efficient Quantum Computation with Linear Optics," Nature, pp. 46-52, January 2001.

[5] M. Nielsen and I. Chuang, "Quantum Computation and Quantum information," Cambridge University Press, 2000.

[6] A. Mishchenko and M. Perkowski, "Logic synthesis of Reversible Wave Cascades," in Proc. Int. Workshop. Logic Synthesis, pp. 197-202, June 2002.

[7] R. Feynman. "Quantum Mechanical Computers," Optic News, pp.11-20, 1985.

[8] E. Fredkin, and T. Toffoli, "Conservative Logic," International Journal of Theoretical Physics, 21:219-253, 1982.

[9] D. Maslov, "Reversible logic synthesis benchmarks page," May 2005. http://www.cs.uvic.ca/~dmaslov/

[10] T. Toffoli, "Reverible computing," Tech memo MIT/LCS/TM-151, MIT Lab for Computer Science, 1980.

[11] T. Sasao, "Logic Synthesis and Optimization," Kluwer Academic Publishers, 1993.

[12] P. Gupta, A. Agrawal, and N. K Jha, "An Algorithm for Synthesis of Reversible Logic Circuits," TCAD, November 2006.

[13] D. M. Miller, D. Maslov, and G. W. Dueck, "A Transformation Based Algorithm for Reversible Logic Synthesis," DAC, pp. 318-323, 2003.

[14] D. M. Miller, "Spectral and Two-Place Decomposition Techniques in Reversible Logic," MWSCAS, pp. 493-496, 2002.

[15] P. Kerntopf, "A New Heuristic Algorithm for Reversible Logic Synthesis," DAC, pp. 834-837, 2004.

[16] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Synthesis of Reversible Logic Circuits," TCAD, vol. 22(6), pp. 710-722, 2003.

[17] W. Hung, X. Song, G. Yang, J. Yang, and M. Perkowski, "Quantum Logic Synthesis by Symbolic Reachability Analysis," DAC, pp. 838-841, 2004.

[18] M. Saeedi, M. S. Zamani, M. Sedighi, "On the Behavior of Substitution-based Reversible Circuit Synthesis algorithms: Investigation and Improvement," IEEE Computer Society Annual Symposium on VLSI (ISVLSI'07), 2007.

[19] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes., "Reversible logic circuit synthesis," Proc. IEEE/ACM Intl. Conf. on Computer Aided Design, pages 353–60, November 2002.

[20] V. V. Shende, S. S. Bullock, I. L. Markov, "Synthesis of Quantum Logic Circuits," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 25(6), pp. 1000-1010, June 2006.

[21] F. S. Khan, and M. M. Perkowski, "Synthesis of Ternary Quantum Logic Circuits by Decomposition," in Proc. Of the 7th International Symposium on Representations and Methodology of Future Computing Technologies, 2005.

[22] G. Dahl, E. Ovrum, J. M. Leinaas, J. Myrheim, "A tensor product matrix approximation problem in quantum physics", June 15, 2006.

[23] A. Muthukrishnan (RCQI), "An Introduction to Quantum Computing", Quantum Information Seminar Friday, September, 1999.

Efficient SOPC-Based Multicore System Design Using NOC

T.Vanchinathan,
Femtologicdesign,Chennai,
India
e-mail:tvanchi@gmail.com

S.Arunraj,
M.E., VLSI Design Student,
Adhiparasakthi Enginering College,India
e-mail:arunmakes@gmail.com

***Abstract* — Due to the advancement of VLSI (Very Large Scale Integrated Circuits) technologies, we can put more cores on a chip, resulting in the emergence of a multicore embedded system. This also brings great challenges to the traditional parallel processing as to how we can improve the performance of the system with increased number of cores. In this paper, we meet the new challenges using a novel approach. Specifically, we propose a SOPC (System on a Programmable Chip) design based on multicore embedded system. Under our proposed scheme, in addition to conventional processor cores, we introduce dynamically reconfigurable accelerator cores to boost the performance of the system. We have built the prototype of the system using FPGAs (Field-Programmable Gate Arrays). Simulation results demonstrate significant system efficiency of the proposed system in terms of computation and power consumption. Our approach is to develop a highly flexible and scalable network design that easily accommodates the various needs. This paper presents the design of our NOC (Network on Chip) which is a part of the platform that we are developing for a reconfigurable system. The major drawback of SOPC based systems lies in the routing of the various on-chip cores. Since it is technically difficult to integrate more than one core on a single chip, we come across several routing problems which lead to inefficient functioning. Thus we implemented several NOC based routing algorithms which considerably improve accessing speed and enhance the system efficiency.**

Keywords: Multicore system, System on a Programmable Chip (SOPC), Network on Chip (NOC), Multiprocessor System-on-Chip (MPSOC).

I. INTRODUCTION

During the 1990s more and more processor cores and large reusable components have been integrated on a single silicon die, which has become known under the label of System-on-Chip (SOC). Buses and point-to-point connections were the main means to connect the components, Hence they can be used very cost efficiently. As silicon technology advances further, several problems related to buses have appeared. [1][5] Buses can efficiently connect 3-10 communication partners but they do not scale to higher numbers. As a result, around 1999 several research groups have started to investigate systematic approaches to the design of the communication part of SOCs. It soon turned out that the Problem has to be addressed at all levels from the physical to the architectural to the operating system and application level. Hence, the term Network on Chip (NOC) is today used mostly in a very broad meaning, encompassing the hardware communication infra-structure, the middleware and operating system communication services and a design methodology and tools to map applications onto a NOC. All this together can be called a NOC platform. [4] Networks on Chip (NOCs) have emerged as a viable option for designing scalable communication architectures. For multiprocessor System-on-Chips (MPSOCs), on-chip micro networks are used to interconnect the various cores. The main idea with NOCs, besides the solutions to the physical issues, is the possibility for more cores to communicate simultaneously, leading to larger on-chip bandwidths. The adoption of NOC architecture is driven by several forces: from a physical design viewpoint, in nanometer CMOS technology interconnects dominate both performance and dynamic power dissipation, as signal propagation in wires across the chip requires 2 multiple clock cycles. NOC links can reduce the complexity of designing wires for predictable speed, power, noise, reliability, etc., thanks to their regular, well controlled structure. From a system design viewpoint, with the advent of multi-core processor systems, a network is a

natural architectural choice ..NOC can provide separation between computation and communication; support modularity and IP reuse via standard interfaces; handle Synchronization issues; serve as a platform for system test and hence increase engineering productivity.

II. DIFFERENT NOC TOPOLOGIES

The Network-on-Chip (NOC) architecture, as outlined in Figure 1, provides the communication infrastructure for the resources. In this way it is possible to develop the hardware of resources independently as stand-alone blocks and create the NOC by connecting the blocks as elements in the network.

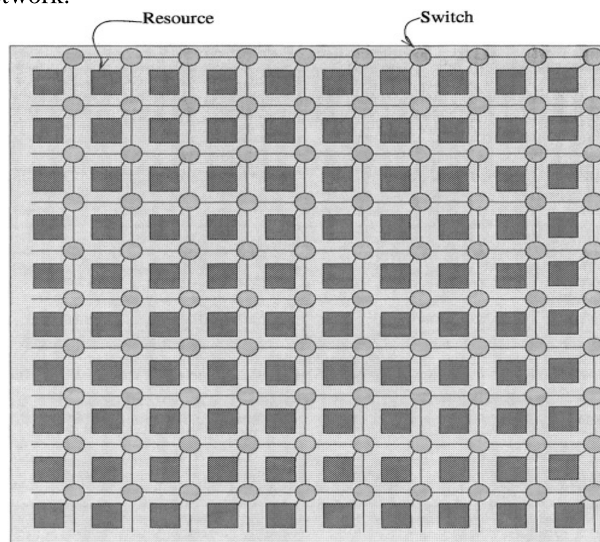


Figure.1. Network on chip [7]

A number of different NOC topologies have been proposed. They all have in common that they connect resources to each other through networks and that information is sent as packets over the networks [7]. Network on Chip (NOC) has evolved as an important research topic during the last few years. The idea is that scalable switched networks are used for on-chip communication among processing units, in order to cope with design of continuously growing systems. Design complexity promotes reuse of investment in earlier designs as well as purchase of outside intellectual property (IP). However, in larger designs, communication among components will become a bottleneck using traditional techniques like common buses. NOC is one solution to address this issue because packet switched communication can provide higher flexibility, throughput and reusability. To gain full advantage when using this concept in NOC architecture design, the size of resources should be similar and the communication facilities should be homogeneous.

2.1 Honey Comb Technology

In NOC design, the resources communicate with each other by sending addressed packets of data and routing them to the destinations by the network of switches [7]. Though many topologies are possible, we will first discuss about Honey comb topology. The overall organization is in the form of a honeycomb, as shown in Figure.2. The resources - computational, storage and I/O - are organized as nodes of the hexagon with a local switch at the centre that interconnects these resources. Hexagons at the periphery would be primarily for I/O, whereas the ones in the core would have storage and computational resource. To further improve the connectivity, switches are directly connected to their next nearest neighbors, as shown in Figure 2, allowing any resource to reach 27 additional resources with two hops. As a last measure to further improve connectivity, every alternate switch is directly connected making each resource element reach a lot more elements with minimal number of hops.

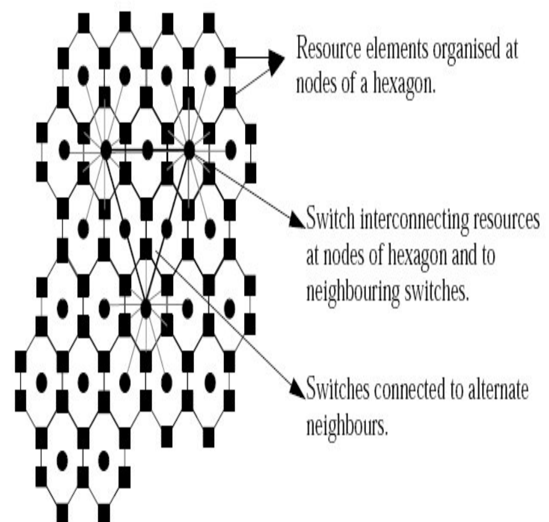


Figure.2. A honey comb structure for NOC [7]

2.2 Mesh Topology

NOC is a scalable packet switched communication platform for single chip systems. The NOC architecture consists of a mesh of switches together with some resources which are placed on slots formed by the switches.[2] Figure 3 shows NOC architecture with 16 resources. Each switch is connected to four neighboring switches and one resource. Resources are heterogeneous. A resource can be a processor core, a memory block, a FPGA, custom hardware block or any other intellectual property (IP) block, which fits into the available slot and complies with the interface with the NOC switch. We assume switches in NOC have buffers to manage data traffic.

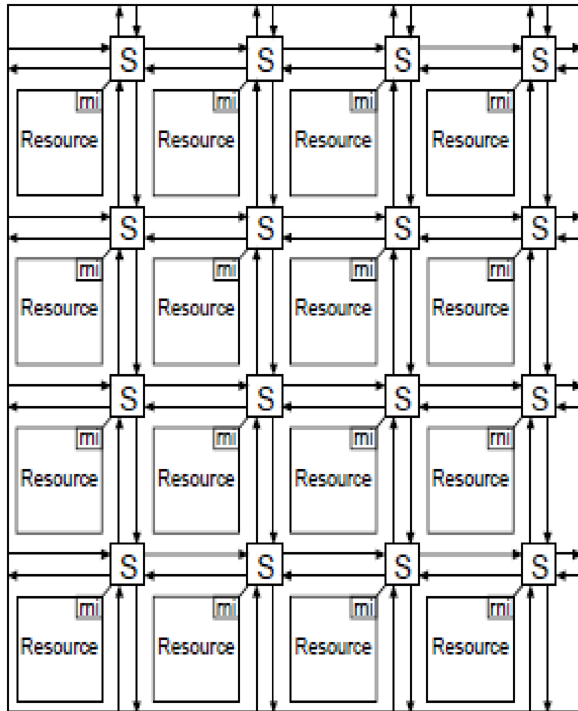


Figure.3. 4×4 NOC switch [2]

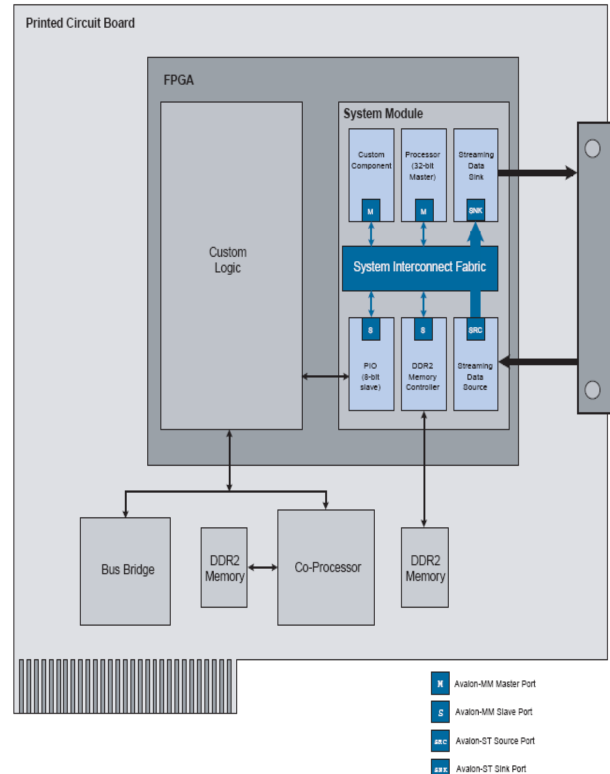


Figure.4. System example [3]

Every resource has a unique address and is connected to a switch in the network via a resource network interface (RNI). The NOC platform defines four protocol layers: the physical layer, the data link layer, the network layer, and the transport layer. The RNI implements all the four layers, whereas every switch to switch interface implements the three of four layers except physical layer. The NOC architecture also has a concept of region allows us to handle physically larger resources and can be used to provide fault tolerance. A typical NOC architecture will provide a scalable communication infrastructure for interconnecting cores. The area of multi-media is a very suitable candidate for using this high computing capacity of NOCs. NOC is a general paradigm and one needs to specialize a NOC based architecture for every application area.

III. SOPC BUILDER

SOPC Builder is a powerful system development tool. SOPC Builder enables us to define and generate a complete system-on-a-Programmable-chip (SOPC) in much less time than using traditional, manual integration methods. SOPC Builder is included as part of the Quartus II software (www.Altera.com). We used SOPC Builder to create systems based on the Nios® II processor. [3]

Figure.4 shows an FPGA design that includes an SOPC Builder system and custom logic modules. We can integrate custom logic inside or outside the SOPC Builder system. In this example, the custom component inside the SOPC Builder system communicates with other modules through an Avalon-MM master interface. The custom logic outside of the SOPC Builder system is connected to the SOPC Builder system through a PIO interface. The SOPC Builder system includes two SOPC Builder components with Avalon-ST source and sinks interfaces. The system interconnect fabric shown below in Figure.5 connects all of the SOPC Builder components using the Avalon-MM or Avalon-ST system interconnects as appropriate. The systems interconnect fabric [3] for memory-mapped interfaces are a high-bandwidth interconnects structure for connecting components that use the Avalon® Memory-Mapped (Avalon-MM) interface. The system interconnect fabric consumes minimal logic resources and provides greater flexibility than a typical shared system bus. It is a cross-connect fabric and not a tri-stated or time domain multiplexed bus. Here we describe the functions of system interconnect fabric for memory-mapped interfaces and the implementation of those functions.

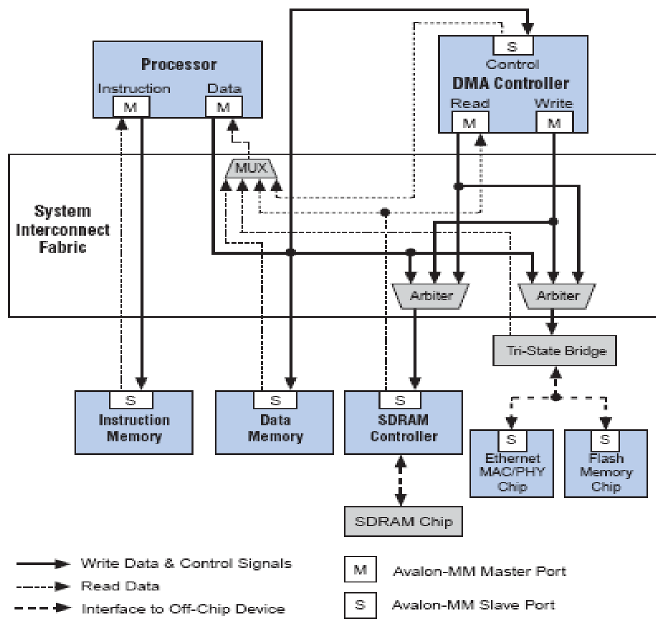


Figure.5. System interconnect fabric

3.1. Chip Planner

The Chip Planner provides a visual display of chip resources. It can show logic placement, Logic Lock and custom regions, relative resource usage, detailed routing information, fan-in and fan-out paths between registers, and delay estimates for paths.

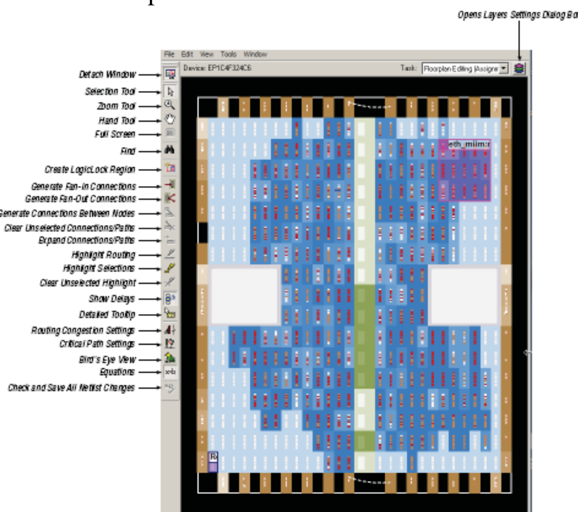


Figure.6. Chip planner tool bar from Quartus II Software

With the Chip Planner, we can view critical path information, physical timing estimates, and routing congestion. We can also perform assignment changes with the Chip Planner, such as creating and deleting resource assignments, and post-compilation changes like creating, moving, and deleting logic cells and I/O atoms. By using the Chip Planner in conjunction with the Resource Property Editor, we can change connections between resources and make post-compilation changes to the properties of logic cells, I/O elements, PLLs, and RAM and digital signal processing (DSP) blocks. With the Chip Planner, we can view and create assignments for a design floor plan, perform power and design analyses, and implement ECOs in a single tool.

3.2. Viewing Routing Congestion

The Routing Congestion view allows us to determine the percentage of routing resources used after a compilation. This feature identifies where there is a lack of routing resources. This information helps us to make decisions about design changes that might be necessary to ease the routing congestion and thus meet design requirements. The congestion is visually represented by the color and shading of logic resources. The darker shading represents greater routing resource utilization. We can set a routing congestion threshold to identify areas of high routing congestion. After selecting the Routing Utilization layer setting, click on the Routing Congestion icon on the taskbar.

3.3. Viewing I/O Banks

The Chip Planner can show all of the I/O banks of the device. To see the I/O bank map of the device, click the Layers icon located next to the Task menu. Under Background Color Map, select I/O Banks.

3.4. Generating fan-in and fan-out Connections

This feature enables us to view the immediate resource that is the fan-in or fan-out connection for the selected atom. For example, selecting a logic resource and choosing to view the immediate fan-in enables us to see the routing resource that drives the logic resource. We can generate immediate fan-in and fan-outs for all logic resources and routing resources. To remove the connections that are displayed, click the “Clear Connections” icon in the toolbar.

3.5. Highlight Routing

This feature enables us to highlight the routing resources used for a selected path or connection.

3.6. Delay Calculation

We can view the timing delays for the highlighted connections when generating connections between elements. For example, you can view the delay between two logic resources or between a logic resource and a routing resource.

3.7. Viewing Assignments in the Chip Planner

Location assignments can be viewed by selecting the appropriate layer set from the tool. To view location assignments in the Chip Planner, select the Floor plan Editing (Assignment) task or any custom task with Assignment editing mode. The Chip Planner shows location assignments graphically, by displaying assigned resources in a particular color (gray, by default). We can create or move an assignment by dragging the selected resource to a new location.

IV. RESULTS

Using SOPC Builder in Quartus II tool, we designed and simulated efficient SOPC-based Multicore System, and the results are listed in Figure 7-11.

Figure 7 shows the screenshot of the developed SOPC builder system. It is done using SOPC builder, this build system has a design of multicore system with 2 CPU's, Avalon tri state bridge, flash memory, LCD and PIO's. It's a FPGA design which includes SOPC builder system and custom logic modules. We can integrate custom logic inside or outside the SOPC builder system. In this design the custom logic modules inside the SOPC builder system communicates with other modules through an Avalon-MM-master interface. The custom logic modules outside of the SOPC builder system is connected to SOPC system through a PIO interface.

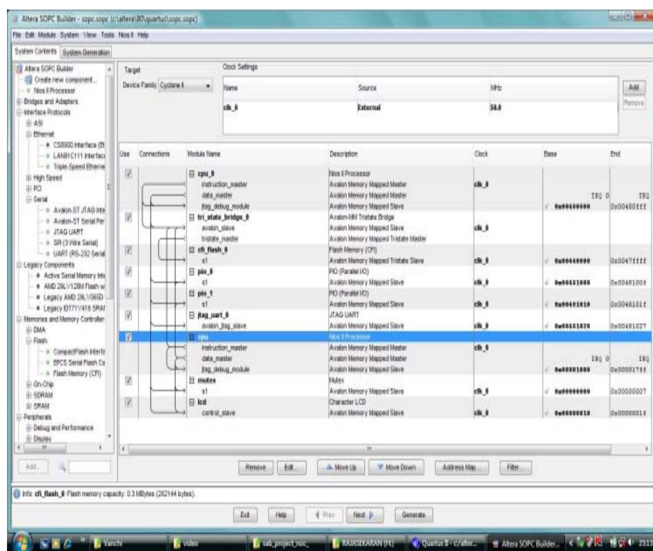


Figure.7. SOPC builder for system building

The block diagram of the symmetric file is shown in Figure 8. SOPC builder allows us to design the structure of a hardware system. The GUI allows adding components to a system configure the components and specify the connectivity. After adding and parameterize components, SOPC Builder generates the system interconnect fabric, outputs HDL files and .BDF during system generation. This .BDF file shown in Figure8 represents the top –level SOPC system for use in Quartus II.

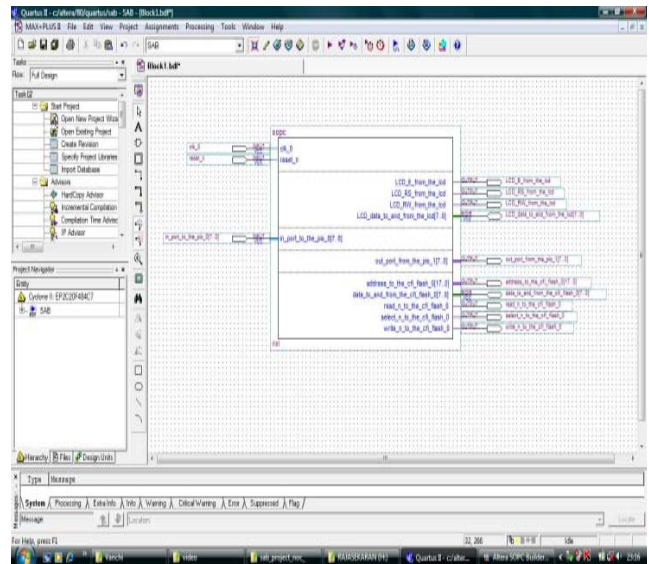


Figure.8. Block diagram of the symmetric file

The compilation result is shown in Figure 9. Once the system design is over it need to be verified whether the designed system has no errors so in order to test the system we compile our system. It shows 100% full compilation during synthesize of our SOPC system.

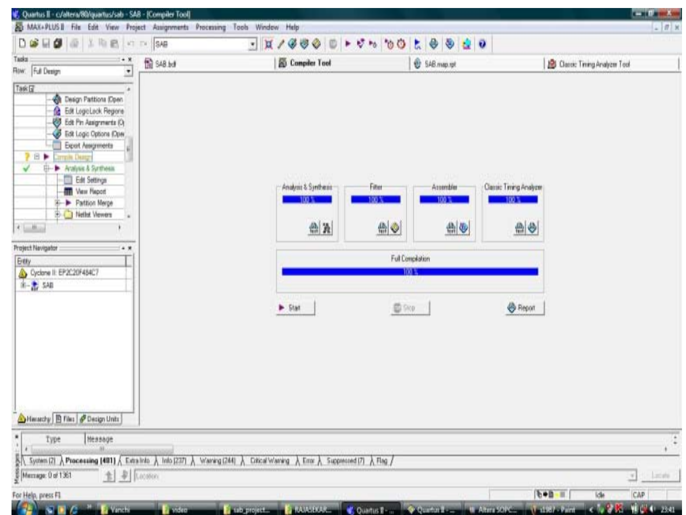


Figure.9. Compilation

The chip planner view of the compiled design is shown in Figure 10. In this screenshot we can see the build components placed in this chip planner. We can view critical path information, physical timing estimation and routing congestion. The Chip Planner uses a hierarchical zoom viewer that shows various abstraction levels of the

targeted Altera device. As we increase the zoom level, the level of abstraction decreases, thus revealing more detail about our design.

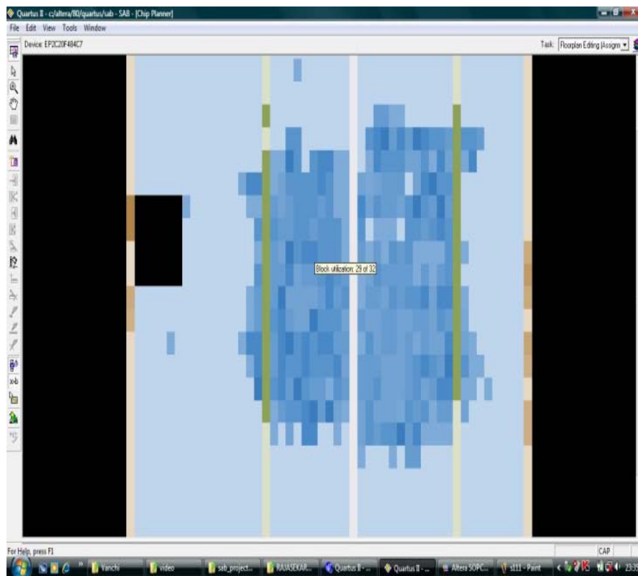


Figure.10. Chip planner view of compiled design

The routing utilization is shown in Figure 11. It allows us to determine the percentage of routing resources used after compilation. This information helps us to make decisions about design changes which is necessary to ease the routing congestion to meet the design requirements. The routing congestion is visually represented by the color and shading of logic resources. In Figure 11 we can see some areas are dark and some areas are bright. The dark regions represent greater routing resource utilization and the bright regions represent no routing congestion.

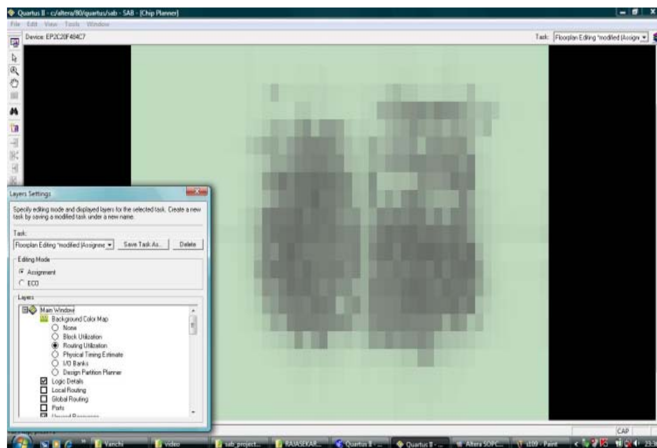


Figure.11. Routing utilization (Darker areas Show dense routing connections)

V. CONCLUSIONS AND FUTURE WORK

In this paper, a simple multicore embedded system was developed and synthesized using Altera SOPC Builder. The synthesize results demonstrate that routing will be the greater issue when it comes to the chip design. Using network-on-chip architecture a prototype system based on networking was developed to overcome the routing issue. As a result, network-on-chip micro networks are used in multicore processors to interconnect the various cores to communicate simultaneously. This leads to larger on-chip bandwidth and reduces routing congestion so that the system efficiency is enhanced. Our designed multicore system has two CPU cores which are individually optimized to the particular computational characteristics of different application fields, complementing each other to deliver high performance levels with high flexibility at reduced cost.
The research focus is shifting from implementation of NOC to investigation of its optimal use. The research problems in NOC design are identified as synthesis of communication infrastructure, choice of communication paradigm, application mapping and optimization. In the future, we will continue to design an efficient multicore SOPC with optimized timing constraints, reduced latency and improved programmability. We will also develop highly embedded, multi-core systems with more number of cores which in turn increases the system performance and many applications can run at the same time.

REFERENCES

[1] Luca Benini, Giovanni De Micheli, “Networks on Chips: A New SoC Paradigm”, *Computer*, v.35 n.1, p.70-78, January 2002.

[2] S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, A. Hemani, “A network on chip architecture and design methodology”, *Proceedings of IEEE Computer Society Annual Symposium on VLSI*, pp. 105-112, April 2002.

[3] Quartus II Handbook, SOPC Builcer, Version 9.0, Volume 4, URL:
www.altera.com/literature/hb/qts/qts_qii5v4.pdf

[4] Axel Jantesh and Hannu Tenhumen, “Networks on Chip”, *Kluwer Academic Publications*, 2003, Boston, USA.

[5] Muhammad Ali, Michael Welzl, Sybille Hellebrand, “A dynamic routing mechanism for network on chip”, *Proceedings of IEEE NORCHIP*, Oulu, Finland, 21-22, Nov. 2005.

[6] International Technology Roadmap for Semiconductors 2003, URL: http://public.itrs.net

[7] H. Tenhunen and A. Jantsch, "Networks on Chip", Springer, 1st edition, ISBN: 1402073925, Jan. 2003.

Author Index

A

Achira Pal......48
Alak Datta......48
Atal Chaudhuri 48
Adithya Thaduri 60
Ajit Verma 60
Adrouche Djamel 71
Atul Gupta 77
Arunraj Subramanyan 89

B

Breeta SenGupta 13
Bikramadittya Mondal...... 37,84
Balvinder Khurana 77

C

......

D

......

E

Erik Larsson 13

F

......

G

Gopika Vinod......60

H

Hideo Fujiwara 7
Hiroaki Yoshida 19,54
Harikrishna Parmar 65

I

......

J

Jaeho Lee 19

K

KS Dasgupta......65

L

......

M

Makoto Nakao 7
Michiko Inoue 7
Masahiro Fujita......19
Mark Zwolinski 43,54
M. Rajesh Gopalan 60

N

Nirjan Devashrayee......65

O

......

P

Priadarshini Shanmugasundaram .25
Pradyut Sarkar......37,84
Pillement Sebastien 71

Q

......

R

Rajat Pal......48

S

Senling Wang 31
Seiji Kajihara 31
Sudhakar Reddy 31
Susanta Chakraborty 37,84
Shohei Ono 54
Sadoun Rabah 71

T

Tomokazu Yoneda 7
Tarak Mandal......48

U

Urban Ingelsson......13
Usha Mehta......65

V

Vishwani Agrawal 25
Vanchinathan Thankavel......89

W

......

X

Xiaoxin Fan......31

Y

Yasuo Sato......7,31
Yang Lin......43

Z

......