저작자표시-비영리-변경금지 2.0 대한민국

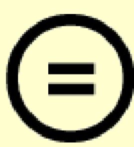
이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



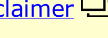
변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer



[UCI]I804:24011-200000267388

August 2019
PhD Thesis

Revisiting Concurrent Error Detection Through Implication Selection Scheme

Graduate School of Chosun University

Department of Computer Engineering

Abdus Sami Hassan

Revisiting Concurrent Error Detection Through Implication Selection Scheme

논리적 함축 관계를 이용한 동시 오류 탐지 기법

August 23, 2019

Graduate School of Chosun University

Department of Computer Engineering

Abdus Sami Hassan

Revisiting Concurrent Error Detection Through Implication Selection Scheme

Advisor: Prof. Jeong A. lee

A thesis submitted in partial fulfillment of the requirements for a PhD degree

April, 2019

Graduate School of Chosun University

Department of Computer Engineering

Abdus Sami Hassan

하싼 압두스 사미 박사학위논문을 인준함

위원장	조선대학교 교수	모상만	
위 원	조선대학교 교수	신석주	
위 원	조선대학교 교수	강문수	(인)
위 원	전남대학교 교수	김철홍	
위 원	조선대학교 교수	이정아	

2019년 06월

조선대학교 대학원

조선대학교
CHOSUN UNIVERSITY

TABLE OF CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS iii

ABSTRACT vi

한글요약 viii

I. INTRODUCTION 1

A. An Overview of CED 1

B. Motivation 3

C. Preliminaries 4

1. Implications 4

2. Direction of Implications 6

3. Probability of Error Detection ($P_{detection}$) 7

D. Related Works 8

E. Contribution 11

F. Thesis Layout 12

II. INPUT VULNERABILITY-AWARE ERROR DETECTION 13

A. Input Vulnerability-Aware Detection 13

B. Proposed Methodology 15

1. Implication Identification 17

2. Weak Implications Removal 18

3. Implication Selection 18

III. RESULTS AND ANALYSIS 22

A. Experimental Methodology for CED 22

Collection @ chosun

1. Execution Time Evaluation 25
2. Compression Rate Evaluation 26
3. $P_{detection}$ Evaluation 27
4. Area and Delay Overheads 30
5. Relationship between $P_{detection}$ and Selected Implications 32

IV. CONCLUSION **33**

PUBLICATIONS & PATENTS **33**

A. Journals 34
B. Conferences 34
C. Patents 35

REFERENCES **42**

ACKNOWLEDGEMENTS **43**

LIST OF ABBREVIATIONS AND ACRONYMS

ATMR	Approximate Triple Modular Redundancy
FATMR	Full Approximate Triple Modular Redundancy
PI	Prime Implicants
UIV	Unprotected Input Vectors
ITMR	Inexact Triple Modular Redundancy
UIV	Unprotected Input Vectors
MEROP	Minterms/Maxterms to expand reduce original PI
MESOP	Minterms set to Expand Original PI
LMEROP	List of MERROPs
SOP	Sum of Product
MSB	Most Significant Bits
LSB	Least Significant Bits
CED	Concurrent Error Detection
ECC	Error-correcting Codes
DMR	Dual Modular Redundancy
TMR	Triple Modular Redundancy
PI	Primary Input
PO	Primary Output
SAT	Boolean Satisfiability algorithm
CNF	Conjunctive Normal Form
RT	Register Transfer
$P_{detection}$	Probability of Error Detection
FAN	Fan-out Oriented Algorithm
RT	Register Transfer
ATPG	Automatic Test Pattern Generation
FPGA	Field-programmable Gate Array

LIST OF FIGURES

1 General architecture for concurrent error detection scheme [3]. . 2

2 Parallel error detection with implications. 4

3 An example combinational circuit for implications. 5

4 Circuit diagram to check $N3(0) \rightarrow N24(0)$ implication. 6

5 Proposed flow to obtain input-aware vectors. 14

6 Flow chart for implication generation. 16

7 A set of wires covered in a LUT. 27

8 $P_{detection}$ plots for the proposed method and [13]. 28

9 Area overhead comparison between the proposed method and [13]. 30

10 Circuit delays for the proposed method and the method in [13]. . 31

11 Relationship between $P_{detection}$ and implication-count. 32

LIST OF TABLES

1 Example: Selection of implication in a 3 input circuit. 15

2 Data structure to store information. 20

3 Circuit profiles. 23

4 Execution times for the implication selection step. 25

5 Compression rate for implication count. 26

ABSTRACT

Revisiting Concurrent Error Detection Through Implication Selection Scheme

Abdus Sami Hassan
Advisor: Prof. Lee, Jeong-A, Ph.D.
Department of Computer Engineering
Graduate School of Chosun University

High operating reliability is a common in-demand trait for digital systems. The technique of Concurrent error detection (CED) is an answer to this need where inspection for computational errors on run-time in parallel to circuit operation is performed. The CED methodology typically comprises of a module which independently predicts certain special characteristic of system output for each input sequence. The checker module establishes that if the special characteristic of the output originally produced by the system in retort to input sequence is the identical as the prediction and produces an error signal in case of occurrence of disparity. Recently, the concept of using implications for online error detection was introduced. The concept revolved around automatic identification of gate-level invariant relationships, called implications, that need to be realized for the proper operation of circuit. Since thousands of implication relations can exist in a circuit, it is not practicable to check all the relationships, as then the supplementary checker hardware would consequence in massive area and delay overheads. Nevertheless, picking only few valuable implications for checking can assist in revelation of maximum number of errors. If these abridged set of implications are recognized precisely, it can be a powerful tool for low cost error detection.

As implication-based CED does not assure 100% error coverage, it is vastly beneficial for those applications where a perfect function is anticipated but not necessary. Utilization of implications for CED is considerably flexible, since a number of existing literatures, deliberate upon auxiliary methods for enhancing $P_{detection}$ by procedural inclusion of additional implications. These works due to their heuristic algorithms cannot assure a lossless $P_{detection}$.

This thesis presents a technique for CED where automatic test patterns are used to select the most valuable implications. A new input-aware implication selection algorithm is developed with the help of ATPG which reduces loss on $P_{detection}$. Moreover, the thesis also presents method on ATPG for fast and thoughtful estimation of $P_{detection}$.

Proposed algorithm carefully evaluates the detectability of errors for each candidate implication using error prone vectors. The evaluation results are then used to select the most efficient candidates for achieving optimal $P_{detection}$. The simulation results on 15 MCNC characteristic combinatorial benchmark suite show that the implications chosen from our algorithm attain better $P_{detection}$ in evaluation to the state of the art. The proposed method also offers better performance. We are able to improve impact-level by 41.10%, which is the ratio of achieved $P_{detection}$ to the implication count. The execution of our algorithm is also 4.5 times faster on average.

한 글 요 약

논리적 함축 관계를 이용한 동시 오류 탐지 기법

하싼 압두스 사미
지도 교수: 이정아
컴퓨터공학과
대학원, 조선대학교

동시 오류 탐지 기법 (Concurrent Error Detection) 은 회로 작동 시간에 발생하는 오류를 탐지하는 것으로 디지털 시스템의 안정적 작동에 기여한다. 이 기법은 입력 시퀀스에 대한 시스템 출력의 특성을 독립적으로 예측하는 모듈로 구성된다. 체크인 모듈은 입력 시퀀스에 따른 시스템 출력의 특성이 예측과 동일하지 않은 경우, 오류 신호를 생성한다. 최근에 동시 오류 탐지에 논리적 함축 관계를 이용하는 기법이 도입되었다. 이 기법은 회로에 내재하는 게이트 레벨의 불변성 관계의 자동 식별에 기반하여 작동한다. 회로에 수천 개의 논리적 함축 관계가 존재할 수 있으므로, 모든 관계를 확인하는 것은, 추가적인 하드웨어와 지연 오버헤드를 야기하기 때문에, 실무적으로 가능하지 않다. 반면에 몇 가지 중요한 논리적 함축 관계의 선택은 오류의 최대 발견에 도움이 될 수 있다. 만약 선택된 논리적 함축 관계가 오류 탐지에 미치는 영향을 예측할 수 있다면, 그것은 저비용 동시 오류 탐지를 실현할 수 있는 강력한 도구가 될 수 있다.

논리적 함축 관계를 이용한 동시 오류 탐지 기법은 100% 오류 커버리지를 보장하지 않기 때문에 안정적으로 작동하되, 완벽한 오류 탐지는 요구하지 않는 애플리케이션에 유용하다. 기존의 방법들은 논리적 함축 관계의 추가적인

선택에 따른 $P_{detection}$을 개선하기 위한 다양한 방법을 제안하였으나, 이러한 휴리스틱 알고리즘은 $P_{detection}$의 무손실 감지를 보장하지 못한다.

본 논문은 회로에 존재하는 다양한 논리적 함축 관계 중에서 출력 오류 탐지의 최대 발견에 필요한 논리적 함축 관계를 선택하기 위하여, 테스트 패턴을 자동적으로 생성하는 ATPG(Automatic Test Pattern Generation)를 활용한다. 자동 테스트 패턴 생성(ATPG)을 활용하면, $P_{detection}$을 최대화할 수 있는 논리적 함축 관계를 효율적으로 선택할 수 있다. 본 논문에서 제안한 기법은, ATPG를 이용하여 출력 오류에 취약한 입력 벡터와 연결된 논리적 함축 관계를 중점적으로 평가하고, 출력 오류를 최대한 탐지할 수 있는 논리적 함축 관계를 선택하는 고속화를 달성하여, 기존에 비하여 속도를 약 4. 5배 향상하였다. 15 MCNC 특성 조합 벤치마크에 대한 시뮬레이션 결과는 본 논문에서 제안한 방법에 따른 논리적 함축 관계의 선택은 기존에 제안된 방법에 비하여 우수함을 보였다.

Collection @ chosun

I. INTRODUCTION

A. An Overview of CED

In electronic circuits, for acquiring lower power and higher frequencies, the prevailing method is to explore the aggressive scaling of semiconductors [1]. However, this method to reduce the feature size also introduces a number of reliability related problems such as defects and single event upsets. Although, mission-critical applications in space, automotive and medicine need a high operating reliability [2]. Hence, an error detection method is needed for ensuring the correctness of results. Concurrent error detection (CED) approaches can fulfill this need by scrutiny for computational errors on run-time in parallel to circuit operation. In the case of an error, CED schemes warn the operator by triggering an error flag or activate an automatic system repair technique.

CED approaches are widely utilized to enhance system dependability. Almost all CED approaches function in accordance with the following principle: Let us assume that the considered system realizes a function f and generates an output f(i) in reply to an input sequence i. A CED technique generally encompass another unit that independently predicts some special characteristic of system-output f(i) for every input sequence i. Lastly, a checker unit determines if the special characteristic of the output really generated by the system in response to input sequence i is the same as the forecast and generates an error signal when a disparity occurs. Some instances of the characteristics of f(i) are: f(i), its parity, 1's count, 0's count, transition count, etc. General CED scheme is illustrated through the architecture shown in Figure 1. Any CED method is illustrated by the class of failures in the occurrence of which system data integrity is maintained [3]. By data integrity, its meant that the system either generates correct outputs or

detects erroneous situations when incorrect outputs are generated. In the existing literature on fault-tolerance, this property is known as fault-secure property. Notice that the general architecture of a CED technique for example Figure 1 depends on use of hardware redundancy (predictor and checker circuits) aimed at error-detection. Time redundancy schemes such as alternate recomputation and data-retry with shifted operands can be utilized for concurrent error detection. Time redundancy directly impacts the system performance though the hardware cost is typically less than that of hardware redundancy.

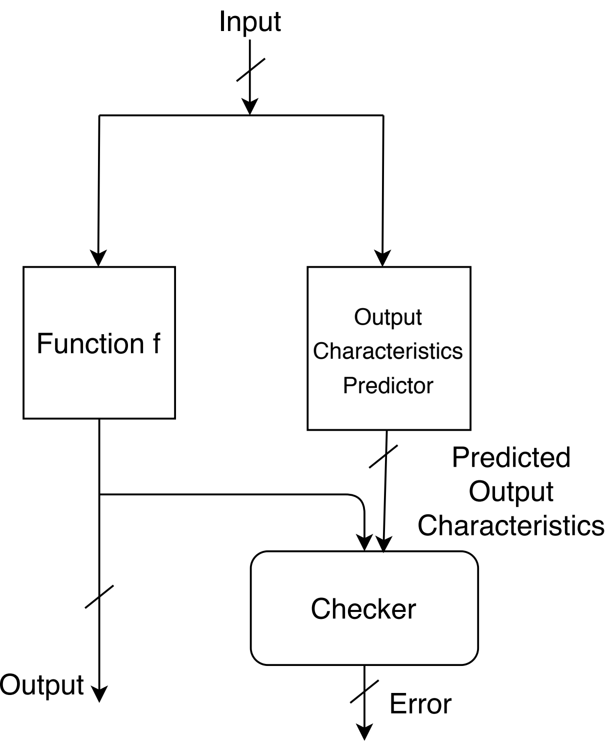


Figure 1: General architecture for concurrent error detection scheme [3].

Collection @ chosun

B. Motivation

The work of Alves et al. [13] introduced the notion of using implications for online error detection. The proposed technique automatically identifies gate-level invariant relationships, known as implications, which are required to be fulfilled for the circuit to be operating properly. Invariant relationships are logic value implications between various wires in a circuit. The arbitrary combinational logic is tested with invariant relationships on runtime. Accurate operation is certain as long as these relationships are fulfilled during operation. For example, think of an AND gate; if any input is logic-0, logic-0 is implied at the gate output. Upon occurrence of error occur during normal operation, implication's checker can notify the user.

Figure 2 illustrates the parallel error detection though implications. Error signals are formed by the checkers checking these implications whenever implication associations are violated. As thousands of implication relations can exist in a circuit, it is not feasible to check all the relationships, since then the additional checker hardware would result in huge area and delay overheads [4]. Though, selecting only few valuable implications for checking can aid in detection of maximum number of errors. If these reduced set of implications are identified accurately, it can turn into a powerful tool for low cost error detection. Another advantage of using implications for CED is that identification of invariant relationships need no knowledge of high-level behavioral constraints. As implication-based CED does not guarantee 100 % error coverage, it is highly useful for those applications where a perfect function is desirable but not necessary. Using implications for CED is much flexible, as few existing studies [4], [5] discuss additional methods for enhancing $P_{detection}$ by methodological

inclusion of more implications.

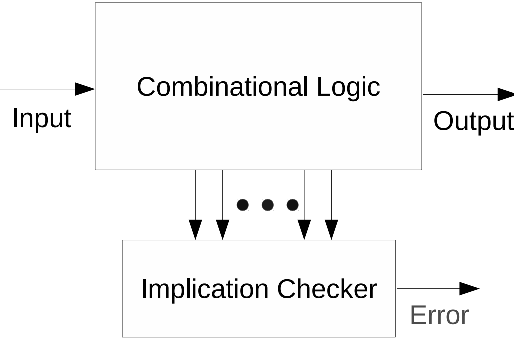


Figure 2: Parallel error detection with implications.

Nevertheless, in opting for reduced set of valuable implications, the current works are unable to reflect each of the errors in a circuit, which are noticeable at the outputs due to cogitating only a subgroup of the total input space of the indented circuit. Hence, in this work, we present a method using automatic test patterns to more precisely target those invariant relationships which are pondered esteemed for the total input space.

C. Preliminaries

This section begins from background on implications and then to the selection for valued implications. The implications are believed valuable when they can perceive most of the errors which disseminate to the output.

1. Implications

Implications are invariant relations between diverse wires in a logic circuit. Consider an OR gate for instance; if any of its input is logic-1, logic-1 is inferred

at the output. Such relations can also be discovered amongst wires at diverse logic levels or even between wires of distinct logic cones.

The example combinational circuit in Figure 3 shows that, logic-0 is employed at $N13$, $N12$ and $N11$ when when $N3$ is at logic-0. As a result, whenever $N3$ is at logic-0, $N24$ is at logic-0. This implication is signified as follows.

$$I1 : N3(0) \rightarrow N24(0)$$

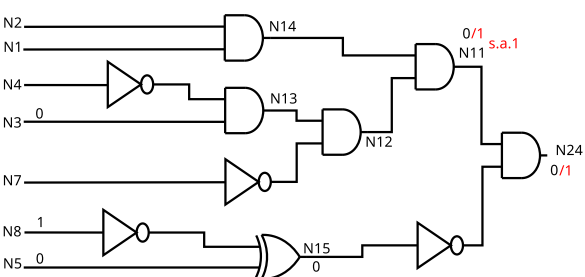


Figure 3: An example combinational circuit for implications.

The wire on the right is called the *implicand* while the left side is called the *implicant*. An implication is enabled when the condition defined for the affiliation is valid. Thus, it is enable when the implicant wire is at the specific logic value. For instance, logic-0 is implied at $N24$ whenever when $N3$ is at logic-0, and implication $I1$ is believed to be enabled.

Implications can be used for concurrent error detection when they are enable. Consider the subsequent scenario where $N8$, $N3$ and $N5$ are at logic-1, logic-0 and logic-0, respectively. This drive $N15$ and $N11$ to logic-0. Let us adopt that a stuck-at-1 fault transpires at $N11$ changing its state from 0 to 1. This roots an erroneous output to be detected at $N24$. Implication $I1$ is defied in this case and a checker checking $I1$ can notice this error. Checker circuits to check violations of two-wire

implications are specified in [4]. For example, the checker for $N3(0) \rightarrow N24(0)$ can be realized as shown in Figure 4. Checkers are intended to raise violation flag only when the particular implication is enable and a disruption occurs.

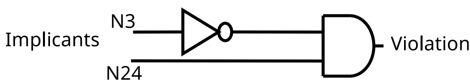


Figure 4: Circuit diagram to check $N3(0) \rightarrow N24(0)$ implication.

2. Direction of Implications

An implication is a provisional statement and, according to formal logic, any provisional proposition has a contrapositive which is shaped by contradicting both the theory and the deduction and switching their positions. Implications can be of two categories: backward implications and forward implications. For ease of readers, consider the forward implication $A(0) \rightarrow Y(0)$, its backward implication interpretation to the definition given above is attained by switching the positions of the implicant and implicand among each other and negating them both i.e., $Y(1) \rightarrow A(1)$. For an implication to notice a disruption, a fault must only disturb the implicand. If a fault upsets both the implicant and the implicand, the implication is cogitated ineffective. This condition is mostly existent in case of backward implications once both the implicant and the implicand wires are located on the same path to the output. If a fault disturbs the implicand wire, then it will also disturb the implicant wire since the implicand wire is situated at a subordinate logic level than implicant wire. Removing backward implications will also decrease the computational resources and effort needed in the valuable implications selection step [6].

Backward implications are not studied in this work, similar to [4], [6], since

they are not useful in improving $P_{detection}$.

3. Probability of Error Detection ($P_{detection}$)

$P_{detection}$ is calculated as the fraction of the total number of detected errors to generated errors.

$$P_{detection} = \frac{\text{Number of Detected Errors}}{\text{Total Number of Generated Errors}}$$

To compute an exact $P_{detection}$, all input patterns should be applied thoroughly to the given circuit which is impossible. If all the implications in a circuit are included, even then a 100% $P_{detection}$ is not probable. Moreover, checking all implications in the circuit results in great hardware overheads [4] which makes it unreasonable to include all implications in the checker hardware. Thus, the goal is to select least number of implications such that a adequately high $P_{detection}$ is attained. We select, one implication per fault which perceives the maximum number of errors for set fault like [13]. While *essential* implications for each fault such that the maximum number of detectable errors can be increased for each fault were added by [4]. This somewhat increases the achieved $P_{detection}$ with rise in the hardware overheads.

The amount of errors that an implication can discover for a specified fault is termed *detectability* of the implication for that fault. The approach used to guesstimate the detectability of each implication in [4] is founded on simulating the circuits using 32,000 pseudo random test vectors. This method does not assure a very precise detectability estimation for all implication since all of the input vectors are not studied. Hence, it may assign a elevated detectability to an implication when in deed that implication could have a low detectability. This

is true, particularly in the case of circuits with a huge input space. Thus, it is vital to consider preferably all the input vectors in detectability estimation of implications.

D. Related Works

For Concurrent Error Detection (CED) in instance of memories, usually, error-correcting codes (ECC) are used [7]. However, detecting errors parallelly in logic circuits is hard as they realize complex boolean functions. Triple Modular Redundancy (TMR) and Dual Modular Redundancy (DMR) are two very well-known error detection approaches [8] for logic circuits. In theory, TMR and DMR provide 100% likelihood of error detection ($P_{detection}$) at the expense of 200% and 100% area overheads, correspondingly. For reducing the enormous hardware overheads suffered by TMR, some recent works emphasize on approximating the TMR for trading-off cost of mitigation with the circuit reliability [9]–[11].

In [12], full ATMR was promoted as an answer for drastic area overhead reduction. Slightly modified Boolean factoring algorithm was used in Approximate functions for ATMR of [12], [13]. The Boolean factoring algorithm was founded upon functional composition paradigm. The method of [12] needs investigation of a variety of approximate circuits to find the optimal ATMR circuit structure. Anyhow, no clear criteria for extracting the optimal ATMR by bringing together approximate modules from the list of approximate candidates was mentioned. In FPGAs, for trading off reliability with the cost of mitigation, partial TMR circuits utilizing approximate logic circuits were proposed as a scalable technique with fine granularity in [14] while in Alves et al. [13] implications based methods for error detection were introduced. In [14] the scheme of line

substitution was adopted, where some of the lines of the circuit are replaced with logic constants and the logic primarily used to implement the substituted line is eliminated. Though Full-ATMR generally leads to better improvement in terms of decrease in area overhead, this technique cannot be utilized to generate a full-ATMR. Furthermore, it was established by [15] that the approach of [14] is only meant for programmable systems because of the need of the under/over-approximations. Two novel voting schemes were proposed by [15], IDMR with TMR (ITDMR) and inexact double modular redundancy (IDMR). ITDMR and IDMR use approximate criteria during comparison and assessment of the outputs of at least two modules for the purpose of reliable computing. However, these schemes [15] have limitations, since they depend on taking averages of inputs. Hence, they are only appropriate for data flow applications. It was concluded in [15] that taking averages of inputs for control-flow dominated applications may lead to chaos. A method in [16] considers the failure probabilities of gates. Cubes are ranked where the best cube is chosen for addition to approximate function (F or H). Only chosen gates of the function will receive fault coverage. Hence, selective fault tolerance is carried out, as not all input vectors of G are provisioned with fault coverage. Though the author maintains that the technique is scalable, the given results are only for up to 14 input functions. Local observability don't-cares are utilized to exploit a richer number of approximations in [17]. This does not guarantee correctness of the solution. A satisfiability (SAT) algorithm must be applied to detect incorrect approximations that should be discarded [18]. The authors claim that the technique is scalable, but do not show results for different approximations. Selective fault tolerance for TMR is focused by [18] and [13], [19]. The area overhead of TMR is decreased by [18] through provision of fault tolerance for particular input signals. The technique ensures that a subset of all

possible inputs has the same fault tolerance as that of TMR whilst the remaining subset is not fault tolerant. A heuristic approach is presented in [19] which allows the applicability of selective fault tolerance for industrialized designs.

Existing literature proposes a number of methods for effective concurrent detection of errors in logic circuits. A self-checking technique using parity for concurrent error detection was given by [20]. In [3], simulation results comparing various parity-based CED systems based on their area overhead are accessible. $P_{detection}$ of parity-based error detection technique is limited as only odd number of errors are detectable [4]. Few of the previous studies have proposed Register Transfer (RT) level CED schemes[21], [22]. Nevertheless, a problem with this method is that the RT-level information might not be accessible in the case of a third-party design [4].

In [23], the authors explore the applicability of Duplication-based CED for asynchronous circuits and illustrate that comparison synchronization can be carried out using a modified comparator that exploits local synchronization protocols which exist in many asynchronous circuits. In [24], a CED method for Quasi Delay-Insensitive circuits is proposed based on unordered codes. In [25], the designing VLSI decoders to implement the Completion-Detection (CD) test on asynchronous buses is studied. In [26], an activity monitoring CD method is presented that detects completion if no transitions are noticed in the circuit during a time period and a transition-monitoring CD technique which guarantees the detection of all glitches in completion logic. Lastly, an efficient and systematic technique for performing CD by multioutput threshold logic is proposed in [27]; an in-depth investigation on various CD strategies can be found in [28]. CED techniques have also been proposed for asynchronous circuits by the use of the Differential Cascade Voltage Switch (DCVS) logic [29], [30],

that is utilized to generate handshaking signals for interacting asynchronous modules. In [29], the authors use the handshaking signals of the DCVS logic to implement a self-checking majority voter. A CED method used for asynchronous interfaces in globally asynchronous locally synchronous circuits is explained in [31]. Asynchronous controllers shows different characteristics that limit the effectiveness and applicability of CED methods developed for their synchronous counterparts. Work [32] explains effective CED in Asynchronous Burst-Mode Machines, which requires a synchronization method that appropriately enables the checkers in order to avoid false alarms. The new paradigm of online error detection in [13] focused on utilization of implications.

E. Contribution

The existing works do not reflect for every error in a circuit that can propogate to the outputs because of cogitating only a subgroup (32,000) of the total input space of the circuit under test. Hence, the thesis targets the problem of selecting implications for achieving minimal loss on $P_{detection}$ for CED. To this end, we make the following contributions.

1. A scheme based on Automatic Test Pattern Generation (ATPG) for defining the number of errors detected by every implication, which allows an enhanced selection of a condensed valuable set of implications for achieving maximum $P_{detection}$ is presented.

2. Circuits implemented on Field Programmable Gate Arrays (FPGAs) for investigation, which makes the testing more illustrative of how implications would achieve in an implemented circuit rather than working on netlists based in simple logic gates is implemented.

3. We present a technique based on ATPG for fast and thoughtful guesstimate of $P_{detection}$.
4. To perceive the implication selection impact on $P_{detection}$ we propose a new metric called impact-level (IL).

F. Thesis Layout

The thesis is structured as follows. In Chapter I, we present an overview of the Concurrent Error Detection, the motivation behind the work and preliminaries containing the relevant terminologies such as Implication. Chapter I, also, provides the related work, our contribution and thesis layout. Then in Chapter II, we introduce the concept of input vulnerability aware error detection followed by the proposed methodology. Next in Chapter III, we put foward the experimental methododlogy for proposed CED technique and evaluate by means of various metrics systematically. And Finally, we conclude the thesis in Chapter IV.

II. INPUT VULNERABILITY-AWARE ERROR DETECTION

A. Input Vulnerability-Aware Detection

As stated in the preceding segment, to proficiently select a compact set of implications, all of input space must be cogitated in the evaluation. However, this, is unfeasible when there are a huge number of input vectors for an aimed circuit.

It should be noticed that the whole input space does not hold equivalent significance as some portion of input space is more susceptible to errors than the rest [33], [34] (i.e., the slice of input space which makes the greatest number of errors noticeable at the primary outputs). During process of producing 32,000 pseudo random vectors for assigning detectability to every implication, some of vectors could be nominated from the portion of input space, which is less susceptible to errors. The, implication selection will be biased then, regarding those implications which perform well for the arbitrarily selected 32,000 input vectors. This means that, implication which is being reflected as having the maximum detectability for the fault may not be the best choice for that fault.

Instead of using 32,000 random vectors in comprehension of prime implications, we explore use of an identified input space [33] that is utmost vulnerable to errors. Input-aware vectors in compact form can be attained for every fault using a ATPG like ATALANTA [35] which is a freely available ATPG tool built on fan-out oriented (FAN) algorithm. For larger circuits, FAN algorithm confines an ATPG's search space to lower its computation time [36]. Using ATALANTA's test initiation system, we can produce all possible test patterns

for every fault, as shown in the Figure 5, to classify the most vulnerable areas of circuit. ATALANTA generates all the test vectors in a compressed form after doing fault simulations on every node of the circuit which can also be treated in their condensed form.

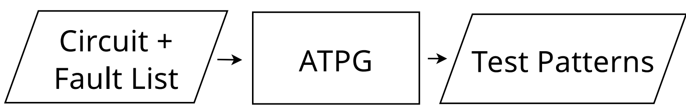


Figure 5: Proposed flow to obtain input-aware vectors.

Vectors that make the errors linked with a fault noticeable at the primary outputs (POs) are produced using ATALANTA, the ATPG tool. For every probable fault in circuit, we consider these input patterns for the more precise selection of implications. This method thus permits us to highlight only those input vectors for every fault that make all errors due to this fault visible at the POs which aids in an accurate detectability evaluation for every implication.

Table 1 indicates how implications are chosen using the example case of a 3 input circuit. The PO is identified as O. We need to select one implication out of subsequent two implications, $I1 : A(1) \rightarrow O(0)$ and $I2 : B(1) \rightarrow O(0)$, where B and A are somewhat two wires in circuit. Lets assume that stuck-at-1 (s.a.1) fault on some inner wire transmits to the output O by the input vectors $000, 011, 110$. Our target is to select a implication which has highest detectability in this situation. Among $I1$ and $I2$, $I1$ is capable to detect the given fault when the related error is noticeable at the output, while $I2$ cannot detect the error noticeable at 110, as $I2$ is not enabled on this input vector. Obviously, $I1$ is desirable for the assumed s.a.1 fault. The hitch lies in only exploiting a random set of vectors in ranking the implications. With an arbitrary set of vectors, we may select $I2$ above $I1$. Which also acts well but does not have highest detectability for the respective

fault under consideration for the total input space. This is as, for circuits with a huge input space, use of pseudo random vectors does not assure that the bulk of fault-aware vectors are used in allotting the detectability to the implication.

Table 1: Example: Selection of implication in a 3 input circuit.

A	*B*	*O*
1	1	0
0	0	0
0	1	0
1	1	0
0	0	1
0	0	1
1	0	0
1	1	0

B. Proposed Methodology

The primary step in implications centered concurrent error detection is to essentially find invariant relationships amongst different wires of a circuit. As wholly considering these implications would result in massive hardware overheads, we need to only select utmost valued implications such that a high $P_{detection}$ is realized at least hardware costs. This marks implication selection the most vital step in implications enabled CED. Furthermore, the purpose of exercising implications in the first place is to aid CED at least hardware costs.

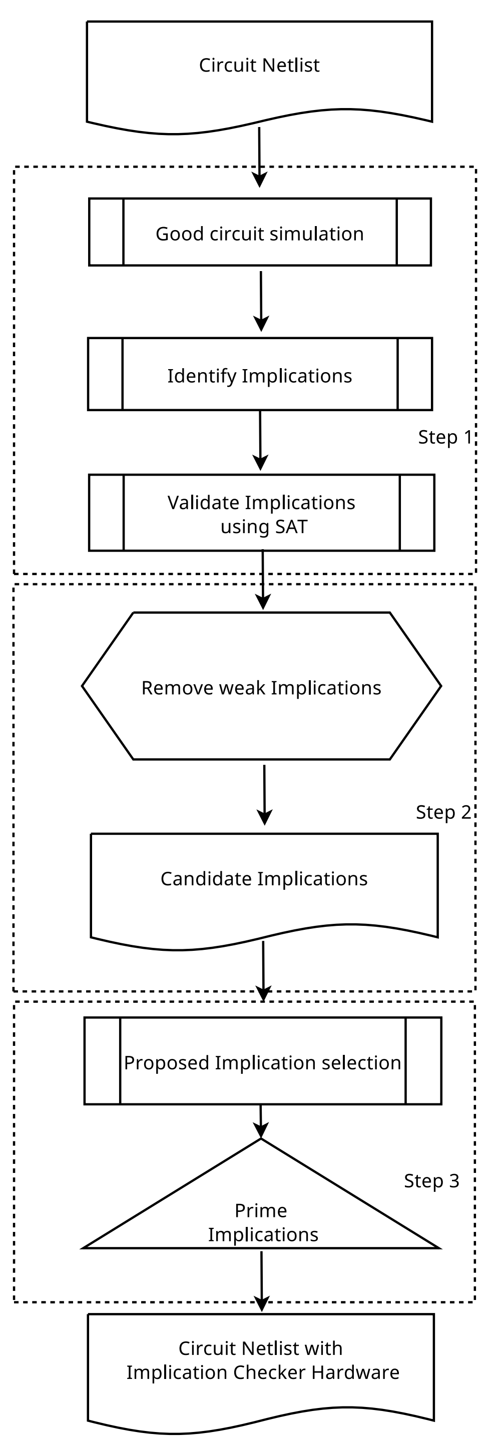


Figure 6: Flow chart for implication generation.

Collection @ chosun

The proposed system guarantees selection of most valued implications precisely through automatic test pattern generation gears rather than trusting on 32,000 pseudo random vectors centered circuit simulation. The course of our algorithm is presented in Figure 6 which comprises of following steps.

1. Recognize implication relations through good circuit simulation.
2. Eliminate potentially weak implications.
3. Prudently select the most valued implications from the residual ones such that the highest $P_{detection}$ attainable is ensured.

1. Implication Identification

Primarily, logic values of all wires in a circuit are logged in a good circuit simulation over 32,000 pseudo random input vectors. The simulation outputs are then processed to obtain all the *potential* implications. As the implications identified until this point are only *potential* as our good circuit simulation is not exhaustive, we must distinct valid implications from invalid implications. To windup, a formal method based on Boolean Satisfiability (SAT) is applied to list of potential implications.

The goal of this SAT based authentication is to check for any occasion from the input space where implication under test is void. If no such state exists, then implication under test is considered valid, i.e., no such input vector occurs for which implication relationship is defied. The circuit is changed into conjunctive normal form (CNF) [37] and every implication is tested by restraining the circuit CNF with its violating condition and checking in SAT engine whether there occurs an input state for which this CNF constrained with

the certain implication's violating condition is satisfiable. For instance, consider an implication $n1(1) \rightarrow n2(1)$, circuit CNF is compelled with its violating condition $n1 = 1$ and $n2 = 0$. If SAT solver fails to discover a satisfying solution, this implication is nonvoidable for any input and thus is a *valid* implication.

2. Weak Implications Removal

For elimination of weak implications, a probability strainer is applied to validated implications for speedily identifying and eliminating weak implications. In a weak implication, implicant wire has low probability of being excited with appropriate signal value for which implication relationship stands. Implications whose implicant node has probability fewer than 5% of having the appropriate logic signal according to simulation results are rejected. This onset value can be adjusted for every circuit, though, it is kept persistent for all circuits in order to speedily process list of established implications.

3. Implication Selection

For a certain set of input vectors, it is checked that whether the fault will disseminate to the output and how many times a definite implication can sense the error at the POs. A bit-flip fault on FPGA when articulated as an error can act just like a stuck-at fault. Therefore, *implication detection* is presented as a number which mirrors how many times the implication under test is defied when stuck-at fault is noticeable at the POs. The essential methodology is same as citealves2010cost which is to choose one implication for every possible stuck-at fault in circuit. This set of implications is termed as *prime implications*.

Algorithm 1 illustrates the proposed method for a more accurate selection

Algorithm 1 Algorithm for implication selection.

Input: network, imp_set , faults

Output: best_imps

```
1:  for fault in faults: do
2:      test_vectors = ATALANTA(network,fault)
3:      best_Detectability = 0
4:      for imp in imp_set: do
5:          new_Detectability = HOPE (network,fault,test_vectors,imp_set)
6:          if (new_Detectability > best_Detectability) then
7:              best_Detectability = new_Detectability
8:              best_imp = imp
9:          end if
10:     end for
11:     best_imps[fault] = (best_imp,best_Detectability)
12: end for
13: return best_imps
```

of prime implications. This algorithm processes all implications and chooses the best implication for every fault based on their implication detactabilities. In this algorithm, several specific data structures are used, which are given in Table 2. The implication selection procedure operates in an iterative way over every fault. As defined in Algorithm 1, for every fault, we reckon all those vectors which make the errors related with the given fault noticeable at the POs where, these vectors are called *test_vectors* for particular fault. These vectors can be acquired in compact wild-card patterns from ATPG like ATLANTA [35]. For instance, a wild card *11 signifies both 011 and 111 vectors. These wild card items for every fault are then used in fault simulation of given fault for calculating the

implication detectabilities *new_Detectability* exploiting HOPE [38]. HOPE can process *test_vectors* in their condensed pattern form. For every fault, it considers all implications and selects a sole implication that has the maximum detectability *best_Detectibility* for the particular fault. This implication identifier jointly with its number of detections is saved beside the fault identifier as *best_imps*.

Table 2: Data structure to store information.

Name	Type (Data Type)	Single Entity Size (Bytes)	Description
imp_set	List	72	List of all candidate implication
faults	List	72	List of all stuck-at faults
best_imps	Dictionary	288	Best Implications against corresponding faults
test_vectors	List	72	List of all the vectors where faults are observable at POs

The selected implications are organized in descending order rendering to their associated fault detection. The implications comprised in this well-ordered list are titled as the prime implications. Top-candidates from list are more probable to deliver better fault coverage than implications situated at the bottom of the list. Lastly, with the ordered list, it is now feasible to select a subclass of these implications starting from top that fulfills a certain hardware budget to incorporate in the checker logic.

Based on the similar methodology, test patterns produced through ATALANTA for every fault are then used for accurate estimation of the selected implications in the computation of $P_{detection}$. Algorithm 2 illuminates in detail our scheme for $P_{detection}$ calculation. Instead of employing 32,000 random vectors, only *test_vectors* in compressed shape are applied. The implication checker hardware is then added to netlist and this netlist along with fault list and test vectors for each fault are postulated to HOPE in order to compute $P_{detection}$. In

Algorithm 2, *TP* and *TM* denote True Positive and True Miss, correspondingly.

Algorithm 2 Algorithm for implication testing.

Input: network , faults, test_vectors, best_imps

Output: $P_{detection}$

```
1:  TP = 0
2:  TM = 0
3:  for fault in faults: do
4:      test_vectors = ATALANTA(network,fault)
5:      detection = HOPE(network,fault,test_vectors,best_imps)
6:      if (detection == True) then
7:          TP = TP + 1
8:      else
9:          TM = TM + 1
10:     end if
11: end for
12: Calculate (P_detection)
13: return P_detection
```


Collection @ chosun

III. RESULTS AND ANALYSIS

A. Experimental Methodology for CED

The methods described formerly were applied to a group of combinatorial MCNC benchmarks. Implication centered CED can also be functional for sequential circuits [4], [13], though, we have limited ourselves to just combinational circuits in this work since the tenacity is only to equate amongst our algorithm with that in [13]. The main apprehension with respect to implication based CED on sequential logic is that of its applicability. Afterwards, the next apprehension is concerning achieving adequate $P_{detection}$ in sequential circuits. Both problems have already been assessed in [4], [13]. Implication centered CED can be applied to sequential circuits just by opening all the feedback registers and turn them into primary circuit inputs, efficiently changing the sequential circuit into a combinational circuit for processing resolutions [13].

We used FPGA-based netlists synthesized by means of the Xilinx Synthesizer Tool (XST) in the experimentations. The FPGA realized post-synthesis netlists are used so that the enactment of implication-based CED becomes more illustrative of realized circuits unlike netlists based in ingenuous logic gates, as used in [4], [13]. It should be distinguished that the results attained by implications enabled CED contrast depending upon the specified circuit implementation. In FPGAs only, there could be numerous circuit implementations according to type of gates in the particular library. Consequently, the $P_{detection}$ results would differ even among different FPGA types. Yet, the expected performance enhancement of our method comparative to [13] would stay applicable on any circuit implementation. The profiles of circuits used in this thesis-work comprising their primary inputs (PIs), primary

조선대학교
CHOSUN UNIVERSITY

Table 3: Circuit profiles.

Circuit	PI	PO	LUT
in5	24	14	70
in6	33	23	63
sao2	10	4	22
br1	12	8	29
br2	12	8	27
luc	8	27	46
alcom	15	38	38
al2	16	47	47
dk17	10	11	29
dk27	9	9	14
dk48	15	17	33
ibm	48	17	49
signet	39	8	66
t481	16	1	21
x6dn	38	5	93

outputs and the quantity of look-up tables (LUTs) disbursed on the Xilinx Virtex-5 XCVLX100T device are provided in Table 3.

Now, we describe how the algorithm rationalized in Section. III-E is realized. Foremost, Icarus Verilog [39] is employed for a good circuit simulation of the post-synthesis Xilinx verilog format netlist for a size of 32,000 random vectors. The logic values for every wire are logged in during this period of circuit simulation. A Python driver is used to explore for potential implications after the good circuit simulation output records. The potential implications are then established in the SAT engine. We have used PicoSAT [40] solver for implication authentication. The output of this stage is a list of confirmed implications. The implications whose implicant node have probability of less than 6% of taking the appropriate logic value essential for enabling the implication are detached

Collection @ chosun

according to good circuit simulation output. The output is list of candidate implications to choose prime implications from. The netlist is then transformed from a Xilinx verilog structure to BENCH format. During transformation, the names of wires in original LUT-based netlist are well-preserved so that merely faults on those wires will be handled in the following steps. Subsequently, for every stuck-at fault in circuit, ATALANTA is used to harvest the *test_vectors* in compressed pattern form. The fault simulator HOPE [38] is then provided with fault list, the test_vectors for every fault and the circuit netlist with checkers for every candidate implication. HOPE is capable to process the test_vectors in compressed form. The output of this phase is a simulation output for every fault containing report about the detectabilities of all candidate implication checkers for that fault. Therefore, the finest implication for every fault is elected based on this output from HOPE.

The best implications for every fault are called *prime* implications. Following, we insert the checkers for all these prime implications to original circuit netlist and send the modified netlist along with list of all the potential stuck-at faults in the circuit and their test_vectors to HOPE. Then we use HOPE's fault simulation output towards calculation of $P_{detection}$ attained by the prime implications.

We have associated our results with state-of-the-art implication selection algorithm from [13]. In our thesis-work, the algorithm from [13] was also comprehended using the same tools. The model in [13] varies against this thesis-work in that the implication selection phase uses 32,000 random vectors in its place of the exact test_vectors for every fault. All boolean algebra functions in this work were implemented using the Python pyEDA module [41].

Table 4: Execution times for the implication selection step.

Circuit	Proposed Time (min)	[13] Time (min)
in5	0.11	1.72
in6	0.07	1.77
sao2	0.01	0.08
br1	0.03	0.4
br2	0.02	0.36
luc	0.04	1.94
alcom	0.06	1.74
al2	0.05	2.25
dk17	0.04	0.52
dk27	0.01	0.2
dk48	0.04	1.43
ibm	1.77	0.59
signet	1.1	0.91
t481	0.04	0.08
x6dn	0.12	2.38

1. Execution Time Evaluation

The evaluation of total execution time in the selection of prime implications for the benchmark circuits studied is shown in Table 4. The simulations were executed on a linux system running a 3.300 Ghz quad-core with sixteen GB DDR3 memory. Methodical generation of condensed input patterns which are used to test the implications makes the performance of the proposed algorithm quicker; in fact, proposed algorithm takes 0.234 min on average compared to the 1.09 min by the technique in [13].

In Table 4, we notice that 2 test circuits *ibm* and *signet* explicitly have higher execution times in the our algorithm. This is owing to an enlarged number of test vectors produced by the ATPG. Since there are numerous faults that do not

Table 5: Compression rate for implication count.

Circuit	Validated	Selected Implications [Proposed]	Selected Implications [13]
in5	617	68	78
in6	472	69	79
sao2	62	8	11
br1	286	38	41
br2	240	35	27
luc	485	53	53
alcom	412	121	100
al2	534	94	83
dk17	356	39	40
dk27	152	11	27
dk48	603	39	49
ibm	86	41	42
signet	232	68	68
t481	86	11	11
x6dn	232	86	98

collapse into further faults, the ATPG yields a lengthier time in producing the test vectors for a greater number of faults.

2. Compression Rate Evaluation

Table 5 displays the results of implication reduction for the numerous benchmarks. Column two gives the total number of authenticated implications after processing potential implications in a SAT engine. Column three offers our total number of selected *prime* implications while Column four provides the number of *prime* implications selected by [13].

The compression rates fluctuate depending upon the benchmark circuit under consideration. On highest, the amount of selected implications will be identical

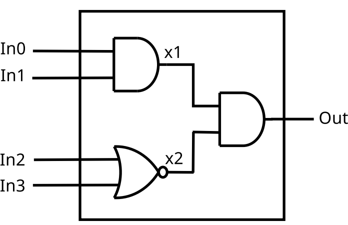


Figure 7: A set of wires covered in a LUT.

to the total number of potential stuck-at faults in circuit since only a sole implication deemed best for a particular fault is selected. Though, when the similar implication is selected for numerous faults, the selected implications total will be lesser than the highest possible.

Now, we can spot the difference among operating on a post-synthesis FPGA realized netlist. Compared to basic logic gates-based netlists in [13], where authenticated implications total is typically in thousands, the validated implications total on FPGA circuits is much lesser. This is since there are less wires in a LUT-based circuit netlist matched to logic-gate based netlists. As presented in Figure 7, wires $x1$ and $x2$ have been enclosed by the LUT and realized in memory thus decreasing the number of wires accessible to search candidate implications from.

The compression rate varies with specific benchmark, though on average a decent comparable reduction can be seen in proposed method. It can be seen that up-to 59.2% drop in implication total is achieved as in case of dk27.

3. $P_{detection}$ Evaluation

For every fault, its test_vectors are applied to circuit under test and the retort is categorized into one of the following 4 cases :

1. No Effect.

2. True Positive (TP): Faults transpire at the internal node and are noticed and visible at PO.

3. False Positive (FP): Faults transpire at the internal node and are noticed but never altered any PO.

4. True Miss (TM): Unnoticed and altered a PO.

The false positive retort is detected either when the checker hardware itself is disturbed by a fault or when an error transpires in the circuit under test but progresses logically masked and therefore is not seen at the PO. In the calculation of $P_{detection}$, we have likewise used the ratio (i.e., $TP/[TP+TM]$), as it has been used in [13].

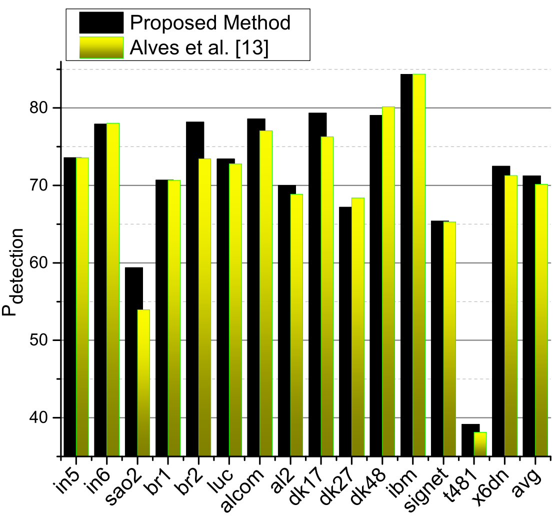


Figure 8: $P_{detection}$ plots for the proposed method and [13].

The $P_{detection}$ metric for our approach and for the approach in [13] are equated in Figure 8. When best implications are selected for every fault using

our method, it does not promise that $P_{detection}$ will increase radically, particularly in this circumstance of FPGA circuits where the total authenticated implications count is already a limited hundred in all of specified circuits. Still the technique based on the selection of implications from pseudo-random vectors [13] performs fine, since it comprises a heuristic step, the attained $P_{detection}$ cannot be deemed lossless. Conversely, in proposed method, it is certain based on the competence of the ATPG tool which produces the test_vectors for every fault, that the attained $P_{detection}$ could be optimum. This can be checked by observing for an instance where the method advised in [13] picks a number of prime implications less than our proposed technique and accomplishes a higher $P_{detection}$. By seeing Figure 8 and Table 5, we comprehend that no such instance occurs. In deed, in very cases, the $P_{detection}$ attained by our algorithm is either higher than [13] or it is almost the identical for both techniques. Furthermore, it was presented with a case study in [4] that even 1% degradation in $P_{detection}$ could trigger millions of unnoticed errors thus highlighting the importance of even minor gains in $P_{detection}$.

This reveals the improved accurateness of the proposed scheme. Note that the trivial decrease in $P_{detection}$ in some instances is due to accuracy of the yield of ATPG. For example, dk27 and dk48 have just 1.2% and 1.1% higher $P_{detection}$ for [13], correspondingly, than the proposed system. In FAN algorithm, tests are produced for every testable fault using backtracks. Due to a restricted number of tries in backtracking, the algorithm halts when a given threshold is surpassed for a fault. Thenceforth, such faults are indicated as aborted faults. For circuits such as br1, in6, dk48, dk48 and ibm, there are definite untestable faults due to which their exact test vectors are unobtainable. It was presented by [42], that the yield attained of faults that are tested using FAN is typically 95.74–99.52%. Yet, the average (avg) $P_{detection}$ of proposed methodology is improved

in comparison to [13]. Besides, FAN can be substituted with more competent test pattern generation algorithms for further reducing any loss on $P_{detection}$.

Therefore, the proposed approach is flexible and not merely it can achieve a improved average $P_{detection}$, it is assured to be optimal particular the accurateness of the ATPG employed.

4. Area and Delay Overheads

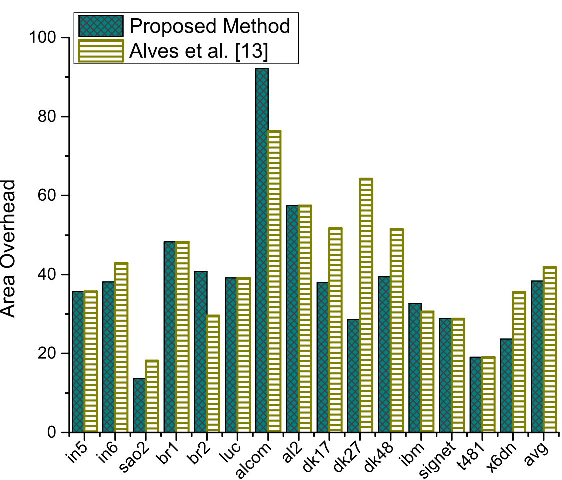


Figure 9: Area overhead comparison between the proposed method and [13].

All circuits were synthesized by means of the Xilinx XST tool for a Virtex-5 FPGA device. In Figure 9, we display the area overhead for every benchmark we have studied when the checkers for implications selected by both approaches are included. The area overhead is considered in terms of the number of LUTs taken on the device. On average our technique outperforms the method in [13]. We have attained a better average $P_{detection}$ while our average area overhead is also inferior than the scheme in [13]. We must state here that the focus of this thesis-work is

to reduce the loss on $P_{detection}$ and the same procedure can also be applied to the choice of essential implications in [4].

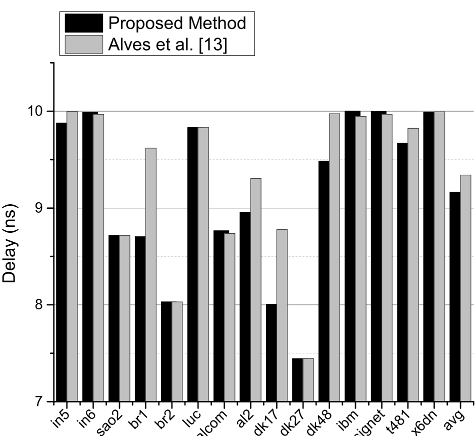


Figure 10: Circuit delays for the proposed method and the method in [13].

In Figure 10, the delay of the circuits following the addition of the checker circuitry is equated. We witness that in those cases where the technique in [13] has attained a higher circuit delay such as al2, br1, dk48 and dk17, the difference is very noteworthy. On the other hand, in circuits such as alcom, in6, signet and ibm, where our method attains a higher delay, the increase is only very small. Thus, we do get a lower circuit delay on average. It must still be noted that in this thesis-work, we do not reflect circuit delays as a restriction in the implication selection procedure, adequately only the detectabilities of implications are pondered. For instance, the circuit in6 shows a marginally higher delay with our method, yet the number of implications selected by proposed algorithm are considerably lower than that in [13]. This is as, despite a lesser implication count, there are more implications on critical path of the circuit causing an rise in the circuit delay compared to the technique in [13]. It is doable to contain the circuit delay as a

constraint in the implication selection phase so that a set of implications adjusted with regards to circuit delay is obtained as shown in delay-optimized implication selection results [13].

5. Relationship between $P_{detection}$ and Selected Implications

To perceive the implication selection impact on $P_{detection}$, we insinuate a new metric called impact-level (IL). We define IL as the fraction of $P_{detection}$ to selected prime implications count. Evidently, a higher value of IL is appropriate and translates into a improved implication selection. It can be comprehended from Figure 11 that the proposed scheme has a higher impact-level. In fact, in some circumstances, it is much improved than [13]. For dk27, with a substantial decrease in area overhead, we are able to increase IL by 41.10% compared to [13]. On average, IL of proposed method is enhanced for the given benchmark circuits.

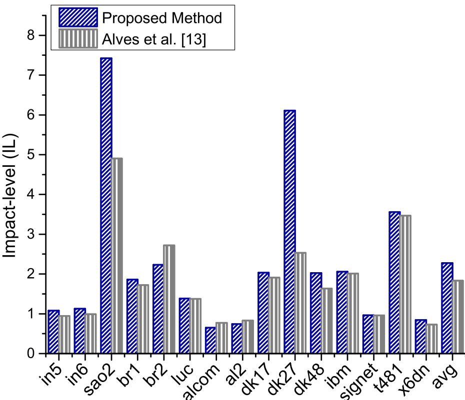


Figure 11: Relationship between $P_{detection}$ and implication-count.

IV. CONCLUSION

In this thesis, with the aim of providing higher reliabilty to mission ciritical applications, we have focussed on Concurrent Error Detection (CED) through implications. As implications are gate-level invariant relationships for a circuit, they can even exist in thousands. To opt for reduced set of valuable implications, the existing works do not reflect for every error in a circuit that can propogate to the outputs because of cogitating only a subgroup of the total input space of the circuit under test. Hence, the thesis targets the problem of selecting implications for achieving minimal loss on $P_{detection}$ for CED.

To select the most valuable implication candidates, we have proposed an algorithm that operates on an input-aware approach. The proposed methodology provides a better insight for a judicious grading of implications based on their error detectabilities estimated from Automatic Test Pattern Generation (ATPG) algorithms. ATPG helps in defining the number of errors identified by every implication, which permits an improved selection of a abridged valuable set of implications for attaining maximum $P_{detection}$.

The experimental results have validated the efficiency of the proposed approach. It achieves a minimal loss on $P_{detection}$ at reduced hardware overheads. We have achieved up to 41.10% improvement in terms of the proposed impact-level metric compared to state-of-the-art approach. The execution of our algorithm is also 4.5 times faster on average. Using the same fault-aware test vectors based approach, we are also able to estimate the $P_{detection}$ in a fast and more exact way.

PUBLICATIONS & PATENTS

A. Journals

1. T. Arifeen, A. S. Hassan, and J.-A. Lee, “A Fault Tolerant Voter for Approximate Triple Modular Redundancy”, *Electronics*, vol. 8, no. 3, p. 332, 2019.

2. A. S. Hassan, T. Arifeen, H. Moradian, *et al.*, “Generation Methodology for Good-Enough Approximate Modules of ATMR”, *Journal of Electronic Testing*, vol. 34, no. 6, pp. 651–665, 2018.

3. A. Hassan, U. Afzaal, T. Arifeen, *et al.*, “Input-Aware Implication Selection Scheme Utilizing ATPG for Efficient Concurrent Error Detection”, *Electronics*, vol. 7, no. 10, p. 258, 2018.

4. T. Arifeen, A. S. Hassan, H. Moradian, *et al.*, “Input vulnerability-aware approximate triple modular redundancy: higher fault coverage, improved search space, and reduced area overhead”, *Electronics Letters*, vol. 54, no. 15, pp. 934–936, 2018.

B. Conferences

1. T. Arifeen, A. S. Hassan, and J. A. Lee, “Error Correctable Approximate Multiplier with Area/Power Efficient Design Through Mixed CMOS/PTL”, in *2018 21st Euromicro Conference on Digital System Design (DSD)*, IEEE, 2018, pp. 190–195.

2. U. Afzaal, A. S. Hassan, T. Arifeen, *et al.*, “Effect of FPGA Circuit Implementation on Error Detection Using Logic Implication Checking”, in

2018 21st Euromicro Conference on Digital System Design (DSD), IEEE, 2018, pp. 196–200.

3. T. Arifeen, A. S. Hassan, H. Moradian, *et al.*, “Probing approximate TMR in error resilient applications for better design tradeoffs”, in *2016 Euromicro Conference on Digital System Design (DSD)*, IEEE, 2016, pp. 637–640.

4. T. Arifeen, A. S. Hassan, H. Moradian, *et al.*, “Area and Power Efficient Adders for Digital Processing Systems”, in *2017 27th Joint Conference on Information Communication (JCCI)*, 2017.

5. H. Moradian, T. Arifeen, A. S. Hassan, *et al.*, “Accuracy Analyzing of Approximate Adders Based on Input Data Range in IoT”, in *2017 27th Joint Conference on Information Communication (JCCI)*, 2017.

6. A. S. Hassan, A. Hassan, T. Arifeen, *et al.*, “FPGA Based Low Power Design of An FIR Filter Using Polyphase Decomposition”, in *2016 Korea Information Science Society (KISA)*, 2016.

C. Patents

1. J. A. Lee, T. Arifeen, A. S. Hassan, *et al.*, “Approximate adder consists of 4 transistors has TED of 4 and DSP integrated with the adder”, pat., Korea Patent 1020160137439, 2019. [Online]. Available: `http://link.kipris.or.kr/link/main/sharePage_EN.jsp?reg_key=gaXejDaQeuOtyJld32oYgA==&APPLNO=1020160137439`.

2. J. A. Lee, T. Arifeen, A. S. Hassan, *et al.*, “Apprpximate adder consists

of 18 transistors has TED of 2 and DSP integrated with the adder", pat., Korea Patent 1020160138097, 2019. [Online]. Available: http://link.kipris.or.kr/link/main/sharePage_EN.jsp?reg_key=gaXejDaQeu0tyJld32oYgA==&APPLNO=1020160138097.

3. J. A. Lee, T. Arifeen, A. S. Hassan, *et al.*, "Accurate adder consists of 14 transistors and DSP integrated with the adder", pat., Korea Patent 1020160135953, 2019. [Online]. Available: http://link.kipris.or.kr/link/main/sharePage_EN.jsp?reg_key=gaXejDaQeu0tyJld32oYgA==&APPLNO=1020160135953.

4. J. A. Lee, T. Arifeen, A. S. Hassan, *et al.*, "Apprpximate adder consists of 6 transistors has TED of 3 and DSP integrated with the adder", pat., Korea Patent 1020160137438, 2018. [Online]. Available: http://link.kipris.or.kr/link/main/sharePage_EN.jsp?reg_key=gaXejDaQeu0tyJld32oYgA==&APPLNO=1020160137438.

5. J. A. Lee, T. Arifeen, A. S. Hassan, *et al.*, "Systematic methodology for development of approximate circuits for triple modular redundancy modules and triple modular redundancy modules thereof", pat., Korea Patent 1020160107927, 2017. [Online]. Available: http://link.kipris.or.kr/link/main/sharePage_EN.jsp?reg_key=gaXejDaQeu0tyJld32oYgA==&APPLNO=1020160107927.

6. J. A. Lee, T. Arifeen, A. S. Hassan, *et al.*, "Accurate adder consists of 18 transistors and DSP integrated with the adder", pat., Korea Patent 1020160135758, 2018.

REFERENCES

[1] Wilson, “International technology roadmap for semiconductors (ITRS)”, *Semiconductor Industry Association*, 2013. (visited on 07/25/2018).

[2] H. Zhang, M. A. Kochte, M. E. Imhof, L. Bauer, H.-J. Wunderlich, and J. Henkel, “GUARD: Guaranteed reliability in dynamically reconfigurable systems”, in *Proceedings of the 51st Annual Design Automation Conference*, ACM, 2014, pp. 1–6.

[3] S. Mitra and E. J. McCluskey, “Which concurrent error detection scheme to choose?”, in *Test Conference, 2000. Proceedings. International*, IEEE, 2000, pp. 985–994.

[4] C.-H. Wang and T.-Y. Hsieh, “On Probability of Detection Lossless Concurrent Error Detection Based on Implications”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2017.

[5] N. Alves, Y. Shi, J. Dworak, R. I. Bahar, and K. Nepal, “Enhancing online error detection through area-efficient multi-site implications”, in *VLSI Test Symposium (VTS), 2011 IEEE 29th*, IEEE, 2011, pp. 241–246.

[6] U. Afzaal, A. S. Hassan, T. Arifeen, and J. A. Lee, “Effect of FPGA Circuit Implementation on Error Detection Using Logic Implication Checking”, in *Digital System Design (DSD), 2018 Euromicro Conference on*, IEEE, 2018.

[7] R. W. Hamming, “Error detecting and error correcting codes”, *Bell Labs Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.

[8] R. E. Lyons and W. Vanderkulk, "The use of triple-modular redundancy to improve computer reliability", *IBM Journal of Research and Development*, vol. 6, no. 2, pp. 200–209, 1962.

[9] T. Arifeen, A. S. Hassan, H. Moradian, and J. A. Lee, "Probing Approximate TMR in Error Resilient Applications for Better Design Tradeoffs", in *Digital System Design (DSD), 2016 Euromicro Conference on*, IEEE, 2016, pp. 637–640.

[10] I. A. Gomes, M. G. Martins, A. I. Reis, and F. L. Kastensmidt, "Exploring the use of approximate TMR to mask transient faults in logic with low area overhead", *Microelectronics Reliability*, vol. 55, no. 9-10, pp. 2072–2076, 2015.

[11] A. J. Sanchez-Clemente, L. Entrena, R. Hrbacek, and L. Sekanina, "Error Mitigation Using Approximate Logic Circuits: A Comparison of Probabilistic and Evolutionary Approaches", *IEEE Transactions on Reliability*, vol. 65, no. 4, pp. 1871–1883, 2016.

[12] I. A. Gomes, M. Martins, A. Reis, and F. L. Kastensmidt, "Using only redundant modules with approximate logic to reduce drastically area overhead in TMR", in *2015 16th Latin-American Test Symposium (LATS)*, IEEE, 2015, pp. 1–6.

[13] N. Alves, A. Buben, K. Nepal, J. Dworak, and R. I. Bahar, "A cost effective approach for online error detection using invariant relationships", *IEEE Transactions on computer-aided design of Integrated Circuits and Systems*, vol. 29, no. 5, pp. 788–801, 2010.

[14] A. J. Sánchez-Clemente, L Entrena, and M. García-Valderas, “Partial tmr in fpgas using approximate logic circuits”, *IEEE Transactions on Nuclear Science*, vol. 63, no. 4, pp. 2233–2240, 2016.

[15] K. Chen, J. Han, and F. Lombardi, “Two approximate voting schemes for reliable computing”, *IEEE Transactions on Computers*, vol. 66, no. 7, pp. 1227–1239, 2017.

[16] H. Xie, L. Chen, R. Liu, A. Evans, D. Alexandrescu, S.-J. Wen, and R. Wong, “New approaches for synthesis of redundant combinatorial logic for selective fault tolerance”, in *2014 IEEE 20th International On-Line Testing Symposium (IOLTS)*, IEEE, 2014, pp. 62–68.

[17] M. R. Choudhury and K. Mohanram, “Approximate logic circuits for low overhead, non-intrusive concurrent error detection”, in *Proceedings of the conference on Design, automation and test in Europe*, ACM, 2008, pp. 903–908.

[18] M. Augustin, M. Gössel, and R. Kraemer, “Reducing the area overhead of TMR-systems by protecting specific signals”, in *2010 IEEE 16th International On-Line Testing Symposium*, IEEE, 2010, pp. 268–273.

[19] M. Augustin, M. Gössel, and R. Kraemer, “Implementation of selective fault tolerance with conventional synthesis tools”, in *14th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems*, IEEE, 2011, pp. 213–218.

[20] K. Mohanram, E. S. Sogomonyan, M Gossel, and N. A. Touba, “Synthesis of low-cost parity-based partially self-checking circuits”, in *On-Line Testing Symposium, 2003. IOLTS 2003. 9th IEEE*, IEEE, 2003, pp. 35–40.

[21] Y. Liu and K. Wu, "Fault-duration and-location aware ced technique with runtime adaptability", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 3, pp. 507–515, 2014.

[22] J. Oh and M. Kaneko, "Automated selection of check variables for area-efficient soft-error tolerant datapath synthesis", in *Circuits and Systems (ISCAS), 2015 IEEE International Symposium on*, IEEE, 2015, pp. 49–52.

[23] T. Verdel and Y. Makris, "Duplication-based concurrent error detection in asynchronous circuits: Shortcomings and remedies", in *17th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2002. DFT 2002. Proceedings.*, IEEE, 2002, pp. 345–353.

[24] S. J. Piestrak and T. Nanya, "Towards totally self-checking delay-insensitive systems", in *Twenty-Fifth International Symposium on Fault-Tolerant Computing. Digest of Papers*, IEEE, 1995, pp. 228–237.

[25] V. Akella, N. H. Vaidya, and G. R. Redinbo, "Limitations of vlsi implementation of delay-insensitive codes", in *Proceedings of Annual Symposium on Fault Tolerant Computing*, IEEE, 1996, pp. 208–217.

[26] E. Grass, V. Bartlett, and I. Kale, "Completion-detection techniques for asynchronous circuits", *IEICE TRANSACTIONS on Information and Systems*, vol. 80, no. 3, pp. 344–350, 1997.

[27] S. J. Piestrak, "Membership test logic for delay-insensitive codes", in *Proceedings Fourth International Symposium on Advanced Research in Asynchronous Circuits and Systems*, IEEE, 1998, pp. 194–204.

[28] V. I. Varshavsky, *Self-Timed Control of Concurrent Processes: The Design of Aperiodic Logical Circuits in Computers and Discrete Systems*. Springer Science & Business Media, 2012, vol. 52.

[29] N. Kanopoulos, D. Pantzartzis, and F. R. Bartram, “Design of self-checking circuits using dcvs logic: A case study”, *IEEE Transactions on Computers*, vol. 41, no. 7, pp. 891–896, 1992.

[30] D. A. Rennels and H. Kim, “Concurrent error detection in self-timed vlsi”, in *Proceedings of IEEE 24th International Symposium on Fault-Tolerant Computing*, IEEE, 1994, pp. 96–105.

[31] D. Shang, A Bystrov, A. Yakovlev, and D. Koppad, “On-line testing of globally asynchronous circuits”, in *11th IEEE International On-Line Testing Symposium*, IEEE, 2005, pp. 135–140.

[32] S. Almukhaizim and Y. Makris, “Concurrent error detection methods for asynchronous burst-mode machines”, *IEEE Transactions on Computers*, vol. 56, no. 6, pp. 785–798, 2007.

[33] T. Arifeen, A. S. Hassan, H. Moradian, and J. A. Lee, “Input vulnerability-aware approximate triple modular redundancy: higher fault coverage, improved search space, and reduced area overhead”, English, *Electronics Letters*, vol. 54, 934–936(2), 15 2018, ISSN: 0013-5194.

[34] P. Balasubramanian and R. T. Naayagi, “Redundant logic insertion and fault tolerance improvement in combinational circuits”, in *2017 International Conference on Circuits, System and Simulation (ICCSS)*, 2017, pp. 6–13.

[35] H. Lee and D. Ha, "Atalanta: An efficient ATPG for combinational circuits", Technical Report, 93-12, Dep't of Electrical Eng., Virginia Polytechnic Institute and State University, Blacksburg, Virginia, Tech. Rep., 1993.

[36] T. Kirkland and M. R. Mercer, "Algorithms for automatic test-pattern generation", *IEEE Design & Test of Computers*, vol. 5, no. 3, pp. 43–55, 1988.

[37] R. A. Rutenbar, "The first eda mooc: Teaching design automation to planet earth", in *Proceedings of the 51st Annual Design Automation Conference*, ACM, 2014, pp. 1–6.

[38] H. K. Lee and D. S. Ha, "HOPE: An efficient parallel fault simulator for synchronous sequential circuits", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 9, pp. 1048–1058, 1996.

[39] S. Williams, "Icarus verilog", *On-line: http://iverilog. icarus. com*, 2006.

[40] A. Biere, "PicoSAT essentials", *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 4, pp. 75–97, 2008.

[41] C. Drake, "Pyeda: Data structures and algorithms for electronic design automation", in *Proceedings of the 14th Python in Science Conference (SciPy 2015)*, 2015, pp. 26–31.

[42] Fujiwara and Shimono, "On the Acceleration of Test Generation Algorithms", *IEEE Transactions on Computers*, vol. C-32, no. 12, pp. 1137–1144, 1983.

ACKNOWLEDGEMENTS

All praise is due to the Lord of the Worlds alone. This thesis made possible, my heartfelt appreciation for the constant reassurance and prayer of my parents who words cannot describe the level of patience with which they persevere my absence. The exceptional guidance and support from Professor Lee, my supervisor during the course of this degree.