저작자표시-비영리-변경금지 2.0 대한민국

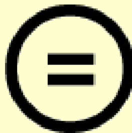
이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 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

dCollection

조선대학교
CHOSUN UNIVERSITY

[UCI]I804:24011-200000509001

August 2021
Doctoral Degree Thesis

Evolutionary Synthesis of Reliable Digital Circuits

Graduate School of Chosun University

Department of Computer Engineering

Umar Afzaal

조선대학교
CHOSUN UNIVERSITY

Evolutionary Synthesis of Reliable Digital Circuits

진화기반 신뢰성 높은 회로 합성 연구

August 27, 2021

Graduate School of Chosun University

Department of Computer Engineering

Umar Afzaal

조선대학교
CHOSUN UNIVERSITY

Evolutionary Synthesis of Reliable Digital Circuits

Advisor: Prof. Lee, Jeong-A

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

April 2021

Graduate School of Chosun University

Department of Computer Engineering

Umar Afzaal

조선대학교
CHOSUN UNIVERSITY

아프잘 우마 박사학위논문을 인준함

위원장 조선대학교 교수 **모상만** (인)

위원 조선대학교 교수 **신석주** (인)

위원 조선대학교 교수 **정호엽** (인)

위원 한림대학교 교수 **이정근** (인)

위원 조선대학교 교수 **이정아** (인)

2021 년 06 월

조선대학교 대학원

조선대학교
CHOSUN UNIVERSITY

Dedicated to parents who love unconditionally.

조선대학교
CHOSUN UNIVERSITY

ACKNOWLEDGEMENTS

All praise is due to the Lord of the Worlds alone. This thesis made possible, my heartfelt regard 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. In sincere acknowledgement, the outstanding companionship of the Muslim community, and in gratefulness, especially the generous hospitality, Mr. Muhammad Adnan, Miss Zobia Irshad, Mr. Zahid Hussain, Mr. Abdus Sami Hassan and Mr. Usman Afzaal. Thank you everyone.

조선대학교
CHOSUN UNIVERSITY

TABLE OF CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS **iii**

ABSTRACT **vii**

한글요약 **ix**

I. INTRODUCTION **1**

A. Contributions 2

B. Thesis Organization 3

II. RELATED WORK **4**

A. Error-masking Schema 4

B. Shared-logic Redundancy 7

C. Platform-specific 8

III. FAULT-RESILIENCE **11**

A. Preliminaries 11

B. Need for Fault-resilience 13

IV. PROPOSED METHODOLOGY **16**

A. Circuit Encoding in the Chromosome 16

B. Constraining the Cartesian Graph 19

1. Population Initialization 21

C. Search Algorithm 22

D. Fitness Function 23

E. Implementation 23

조선대학교
CHOSUN UNIVERSITY

V. EXPERIMENTAL RESULTS **26**
A. Synthesis Results 27
1. Overall Comparison 28
2. Effect of the Proposed Graph Constraints 29
B. Scaling SYFR 30
C. Building Larger Arithmetic Circuits 32
D. Data-Aware Synthesis 33
E. Effect of P_{fault} on Circuit Reliability 39

VI. ReCkt: A Library of Reliable Adders and Multipliers **41**

VII. SUMMARY AND CONCLUSIONS **42**

PUBLICATIONS **44**

BIBLIOGRAPHY **45**

APPENDIX A: ReCkt Library **51**

조선대학교
CHOSUN UNIVERSITY

LIST OF ABBREVIATIONS AND ACRONYMS

CMOS	Complementary metal–oxide–semiconductor
SEU	Single-event upset
SET	Single-event transient
PO	Primary output
TMR	Triple modular redundancy
ATMR	Approximate triple modular redundancy
FATMR	Full approximate triple modular redundancy
DWC	Duplication with comparison
TSC	Totally self-checking circuits
CED	Concurrent error detection
SOP	Sum of products
POS	Products of sum
FPGA	Field-programmable gate arrays
FMR	Fault masking ratio
P_{fault}	Fault-observation probability
CLB	Configurable logic blocks
PE	Processing element
LUT	Look-up table
CNN	Convolutional neural network
CGP	Cartesian genetic programming
CS	Cuckoo search
CS-GRN	Cuckoo search with genetic replacement of abandoned nests
CNF	Conjunctive normal form
SAT	Satisfiability
PDP	Power-delay product
EP	Error probability

조선대학교
CHOSUN UNIVERSITY

LIST OF FIGURES

1 The ATMR configuration uses explicitly redundant modules ,i.e., without logic sharing. 6
2 (a) Half adder circuit implemented using XOR and AND gates. (b) The reduced fault-set for the half adder circuit from Fig. 2(a). 12
3 (a) Half adder circuit implemented using XOR, NAND and AND gates. (b) The reduced fault-set for the half adder circuit from Fig. 3(a). 12
4 A configuration of the CNN architecture LeNet-5. 13
5 Performance of the network for 100 injected faults. (a) Baseline adder (Berkely-ABC) and (b) Fault-resilient adder (SYFR). . . . 15
6 Design flow for the proposed SYFR approach. 17
7 A Full adder represented in a Cartesian graph with parameters G={AND^0,OR^1,XOR^2,NOT^3}, $n_a = 2$, $n_i = 3$, $n_o = 2$, $n_c = 7$, $n_r = 1$. The chromosome with underlined node functions and highlighted outputs: ($\underline{2}$,0,2)($\underline{0}$,0,1)($\underline{2}$,1,3)($\underline{0}$,2,3)($\underline{1}$,4,6)($\underline{3}$,7,5)($\underline{1}$,8,7)(**5**,**7**) . . 18
8 Cascading additional circuitry to enhance the fault-masking property of the baseline. The auxiliary circuit acts as a wire and does not change the intended logic function g. 21
9 P_{fault} spread of the evolved solutions for addr4u, addr8u and mult4u at different values of n_c. 27
10 P_{fault} curves of the test circuits: addr4u, addr8u and mult4u. . . 28
11 Overall comparison: Mean P_{fault} curves for different n_c values. Mean calculated for addr4u, addr8u and mult4u. 29

조선대학교
CHOSUN UNIVERSITY

12 P_{fault} curves of the test circuits: addr4u, addr8u and mult4u. The evolution was carried out in absence of the graph contraints proposed in IV-B. 29
13 P_{fault} curves of the test circuits: c432, c1355 and c880. 31
14 (a) Mean P_{fault} curves for 50 000 vectors. (b) Improvement EP as a function of area overhead and (c) Relationship between improvements in EP and P_{fault} 31
15 (a) Sample 8-bit grayscale image to be sharpened. (b) Filtered version of the same image and (c) the sharper image obtained by adding the original image to its filtered version. 34
16 Percentage of occurence of the 8-bit pixel values for the images contained in the dataset. Note that the input images were scaled to range (0-127) to get 8-bit sum at adder outputs. 35
17 P_{fault} trace plots of addr8u for data-aware synthesis. 36
18 Number of images processed erroneously by the adders. 37
19 Pixel values processed incorrectly by the adders. The color intensity represents the error magnitude. 38
20 Reliability curves for an arbitrary circuit at different values of P_{fault}. 39
21 Pareto-optimal addr4u circuits with respect to (i) P_{fault} and (ii) area, power, delay and PDP. 51
22 Pareto-optimal addr8u circuits with respect to (i) P_{fault} and (ii) area, power, delay and PDP. 52
23 Pareto-optimal addr8s circuits with respect to (i) P_{fault} and (ii) area, power, delay and PDP. 53
24 Pareto-optimal mult4u circuits with respect to (i) P_{fault} and (ii) area, power, delay and PDP. 54

조선대학교
CHOSUN UNIVERSITY

LIST OF TABLES

1 Comparison with related works. 9
2 Distribution of the faults detectable by all test cases for the half adder circuit from Fig. 2(a). 12
3 Distribution of the faults detectable by all test cases for the half adder circuit from Fig. 3(a). 12
4 Adder Profiles. 14
5 Algorithm and fitness function parameters. 24
6 Test circuits profiles. 26
7 P_{fault} improvement [%] of the best evolved circuits compared to the baseline at various Cartesian graph columns n_c (higher is better) 27
8 P_{fault} values of the 8-bit ripple-carry adders and their constituent 4-bit adders. 33

조선대학교
CHOSUN UNIVERSITY

ABSTRACT

Evolutionary Synthesis of Reliable Digital Circuits

Umar Afzaal

Advisor: Prof. Lee, Jeong A

Department of Computer Engineering

Graduate School of Chosun University

In the event of an upset, fault-resilient circuits maintain correct functionality allowing the system to remain fully operational or at least operate with a graceful degradation. Every circuit has a certain level of inherent resilience to faults. Often times, this inherent resilience to faults is insufficient for the given application. This is because conventional synthesis tools generally only focus on optimizing a circuit with respect to area, power or timing budgets. There is a wide range of applications where faulty circuit behavior can lead to fatal results. Fault injection analyses are reported and show that even a single fault can be critical to the desired circuit operation in such applications.

To which end, in this thesis, we present SYFR, an evolutionary method for automated synthesis of increased fault-resilience digital circuits suitable for fine-grained use. Tests results for synthesis of up to 60 input circuits with SYFR are reported. SYFR can be repeatedly applied to a circuit to obtain various design tradeoffs between fault- resilience and implementation costs. SYFR can also be flexibly applied to build circuits which are selectively fault-resilient, i.e., their tolerance to faults is workload-aware. In addition, a novel population seeding mechanism to reduce the design space is introduced and experimentally validated.

조선대학교
CHOSUN UNIVERSITY

In summary, it is shown that SYFR can be considered a competitive synthesis methodology for constructing fault-resilient circuits.

조선대학교
CHOSUN UNIVERSITY

한 글 요 약

진화기반 신뢰성 높은 회로 합성 연구

아프잘 우마
지도 교수: 이정아
컴퓨터공학과
대학원, 조선대학교

신뢰성 높은 회로는, 오류가 발생한 경우에도 회로의 올바른 기능을 유지하여 시스템이 정상적인 작동 상태를 유지하거나 최소한의 단계별 오류 조치로 작동할 수 있게 한다. 모든 회로에는 회로 고유의 오류 허용 내결함성이 있다. 그러나 이러한 오류에 대한 회로 고유의 복원력은 주어진 응용분야에 대부분 충분하지 않다. 이는 기존 회로 합성 도구가 면적, 전력소모, 실행시간을 고려한 회로 최적화에 초점을 일반적으로 맞추고 있기 때문이다. 오류에 의한 회로 동작이 치명적인 결과로 이어질 수 있는 응용 분야는 다양하다. 이러한 응용분야에서는, 오류 모사 입력에 의한 분석을 통하여, 하나의 오류인 경우에도 회로 작동에 매우 심각한 영향을 끼칠 수 있음을 보일 수 있다.

본 논문에서, 신뢰성 높은 회로를 자동적으로 합성하는 방법론으로, 소규모 단위의 로직 변환을 고려하는 진화 기반의 SYFR (SYnthesis of Fault-Resilient circuits) 방법론을 제안한다. 본 방법을 이용하여 신뢰성 높은 회로를 직접적으로 합성한 결과는 60개까지의 로직 입력을 가질 수 있음을 보였다. SYFR 기법은, 분할된 회로에 반복적으로 적용하는 방식을 통하여, 대형 회로에 적용할 수 있는데, 회로의 신뢰도와 회로 구현 비용 조율을 통하여 다양한 설계공간 탐색이 가능하다. 그리고, SYFR 방법론은 오류 발생시 고장 복원력이 있는, 신뢰성 높은 설계 회로 합성에 워크로드 기반으로 선택적으로 유연하게 적용할

조선대학교
CHOSUN UNIVERSITY

수 있다. 본 논문에서 신뢰성 높은 회로의 설계 공간을 줄이기 위한 모집단 씨앗 메커니즘을 새롭게 제안하였으며, 실험결과로 이의 효용성을 보였다. 본 논문은 SYFR 방법론이 신뢰성 높은 회로를 합성하기 위한 경쟁력 있는 방법론으로 간주될 수 있음을 보여준다.

조선대학교
CHOSUN UNIVERSITY

I. INTRODUCTION

The pursuit of performance at lower power consumption has translated into an almost never ending quest for very deep sub-micron circuits. CMOS scaling is the modern approach for operating circuits at higher frequencies, but with a reduced energy consumption [Khoshavi, Ashraf, and DeMara, 2014]. Unfortunately, this has risen circuit reliability concerns since technology scaling accelerates the transistor wearout process degrading the circuit reliability. Nano-scale circuits due to their internal capacitances, and small noise margins resulting from reduced voltage supplies are easily susceptible to environmental radiation, and electrical, electromagnetic, and other noise in general [Lala, 2001]. This means that the probability of a fault occuring in such circuits is higher than ever. Thus, not only mission-critical applications (such as in space or avionics) which already enforce the use of fault-tolerance strategies, but other reliability-oriented applications (e.g., self-driving cars, banking etc.) implemented at modern technology nodes may also require some form of fault-mitigation.

Taxonomically, a soft error is known as a single-event upset (SEU) if it occurs as a transient voltage perturbation at a memory element due to charge transferred by high-energy particles. Otherwise, an upset which causes a transient pulse in combinatorial logic paths is known as a single-event transient (or SET). If an SET is registered at a memory element, it can indirectly manifest as an SEU. Alternatively, in pure combinatorial logic, it is possible that an SET propagates all the way to the primary outputs (POs) causing output errors. Thus, these soft-errors are synonymous with functional errors. For a reliabile operation, soft errors must be mitigated through appropriate fault-mitigation techniques.

Soft errors can be mitigated through the use of masking effects. These include

조선대학교
CHOSUN UNIVERSITY

timing masking, electrical masking or logical masking. The most common forms of soft error mitigation are based on logical masking through hardware redundancy. Triple modular redundancy (TMR) and Duplication with comparison (DWC) are prime examples of logic masking. Theoretically, these techniques guarantee 100 % error-masking or detection capability. However, the incurred area overheads could be more than 200 %. In practice, not all applications require full protection and there is a need for flexible design approaches to find an optimal balance between performance requirements and hardware costs [Polian and Hayes, 2010].

TMR and related redundancy schemes cannot be easily inserted at intermediate nodes for fine-grained fault-tolerance. For intermediate usage, logic masking enabled through shared-logic redundancy is most suitable. The current state of literature on fault-resilient circuits for intermediate use suggests the lack of a flexible approach for building such circuits in a scalable manner. Accordingly, the main contributions of this article are as follows:

A. Contributions

1. We present SYFR (SYnthesis of Fault-Resilient Circuits) for gate-level evolutionary synthesis of fault-resilient digital circuits that are suitable for protection of critical intermediate nodes. SYFR is based on the principles of evolutionary optimization. It is able to exploit a given circuit to create a multitude of increased fault-resilience variants of it at various tradeoffs between fault-resilience and hardware costs.

2. We present a technique for reducing the search space of the optimization problem which significantly enhances the synthesis process's effectiveness.

조선대학교 CHOSUN UNIVERSITY

3. We successfully demonstrate scalability of the proposed SYFR as we report synthesis of up to 60 input fault-resilient circuits.
4. For arithmetic circuits specifically, we show that it is possible to use smaller fault-resilient circuits built with SYFR as building blocks to create larger circuits which are also fault-resilient.
5. We demonstrate the fine-grained use of fault-tolerant circuits produced with SYFR in two examples on neural networks and cryptography.

B. Thesis Organization

The rest of this thesis is organized as follows. In chapter II, we review related literature on fault-resilient circuits and build motivation for the proposed method. In chapter III, we introduce profiling of circuits for fault-resilience and demonstrate the need for fault-resilient circuits in certain applications. Then, key elements and our implementation of the proposed approach are thoroughly discussed in chapter IV. In chapter V, the experimental framework and relevant results are presented. In addition, a library of fault-resilient circuits collected from the experiments performed in this thesis is introduced in chapter VI. Finally, a brief summary and our conclusions on this work are given in chapter VII.

조선대학교
CHOSUN UNIVERSITY

II. RELATED WORK

Fault-resilient circuits can be broadly categorized into two types according to fundamentals of their design: (i) Those which require a higher-level entity or explicit redundancy without logic sharing, and (ii) those which are based on the principle of logic sharing. Consequently, the applicability of these circuits also differ. In this chapter, we review related work from the literature in the context of the aforementioned classification.

A. Error-masking Schema

Of course, adding fault-tolerance capabilities to a circuit requires additional circuitry. The most easily implemented fault-tolerance methods are the classical passive redundancy architectures, such as the well-known TMR [Afzaal and Lee, 2018] which works by comparing three copies of a circuit and produces outputs via consensus voting. TMR is easy to implement, is very reliable, and able to provide a near 100 % fault-mitigation. However, the two major drawbacks which discourage a user from selecting TMR are: (i) it has a high implementation cost, and (ii) it cannot be applied at intermediate nodes since it creates a single point of failure at the consensus junction and therefore TMR must be applied at system level, i.e., the entire system must be replicated.

We will leave out duplication with comparison [Quinn et al., 2016], totally self-checking (TSC) circuits [Garvie and Husbands, 2019] and related redundancy-based methods from our discussion since they primarily belong to the class of concurrent error detection (CED) methods, i.e., the circuit becomes fault-aware but not necessarily fault-resilient. Here, we must mention that some works exist in the literature where a CED method is used to notify the system

조선대학교
CHOSUN UNIVERSITY

of a fault condition which then uses this notification to activate some fault-mitigation mechanism. For example, S. Ostanin et. al. [Ostanin, Kirienko, and Lavrov, 2015] proposed a fault-tolerance scheme based on TSC circuits where one of the modules is self-checking while the other one is a simplex module. The system then switches over to the spare module if the TSC module reports itself as faulty. A similar work on hybrid fault-tolerance architectures (based on CED) is also reported in [Afzaal and Lee, 2020].

Sánchez-Clemente et. al., Arifeen et. al and a few others have reported works on approximate TMR (or ATMR) [Sanchez-Clemente et al., 2016; Sanchez-Clemente, Entrena, and García-Valderas, 2014; Arifeen et al., 2018; Hassan et al., 2018; Gomes et al., 2015]. The idea of ATMR is to trade full coverage offered by TMR with hardware costs. ATMR is a fairly new paradigm for logic masking using approximate logic circuits. If a logic circuit Q implements a logic function g, its approximation $\acute{Q}$ is a logic circuit that implements a slightly different logic function $\acute{g}$. Then, $\acute{Q}$ can be used for masking or detecting errors wherever it overlaps Q in the input space. Based on this principle, an ATMR can be constructed by populating the higher-level entity or error-masking schema with the original circuit Q and the two approximate circuits $Q1$ and $Q2$. Similar to TMR, an ATMR requires forming consensus between the three copies through the use of a voter. The error-masking schema of ATMR is shown in Fig. 1.

For a circuit Q, Sánchez-Clemente et. al. [Sanchez-Clemente et al., 2016] presented three techniques for generating the under- and over-approximate versions $Q1$ and $Q2$. The approximations are generated from a unate description of Q using either of the following: i) line-testability estimations, ii) dynamic probability propagation and iii) evolutionary design. Except for the evolutionary design, scalability was demonstrated for both the line-testing and probability-

조선대학교
CHOSUN UNIVERSITY

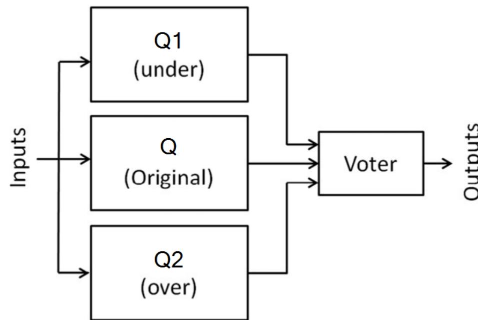


Figure 1: The ATMR configuration uses explicitly redundant modules ,i.e., without logic sharing.

based approaches. However, the evolutionary approach could provide radically different solutions that were hard to explore by the other two methods.

In contrast, Arifeen et. al. [Hassan et al., 2018] developed what is known as a full ATMR or FATMR where all three modules are approximate. The approximations are performed through logic optimization by means of prime implicant expansion and reduction. The result are three approximate modules suitable for use in a FATMR configuration. In [Arifeen et al., 2018], the authors noticed that the entire input space is not equally important as some portion of the input space is more vulnerable to errors. This allowed to drastically reduce the search space for generating approximate modules. The works of Arifeen et. al. [Arifeen et al., 2018; Hassan et al., 2018; Arifeen et al., 2016] and Sánchez-Clemente et. al. [Sanchez-Clemente et al., 2016; Sanchez-Clemente et al., 2012; Sanchez-Clemente, Entrena, and García-Valderas, 2014] are the state-of-the-art in ATMR methodologies. An in-depth survey on the current state of ATMR is available in [Arifeen, Hassan, and Lee, 2020].

The works reported in this section require the use of an error-masking schema which makes them infeasible for intermediate usage. For example, if TMR (or

조선대학교
CHOSUN UNIVERSITY

ATMR) is applied fine-grained, it must be applied as a multiple partition where the voters are kept decentralized until the POs (see Fig. 1 in [Afzaal and Lee, 2018]). Finally, consensus can be formed using a voting circuit made out of reliable components. Alternatively, if only a critical intermediate node requires fault-tolerance, then TMR must use hardened voting circuits [Afzaal and Lee, 2018; Arifeen, Hassan, and Lee, 2019]. This is infeasible if the circuit has multiple output lines because each output line will require a separate hardened voter. This introduces complexity and defeats the purpose of a partial TMR which is to reduce hardware overheads.

B. Shared-logic Redundancy

Inserting fault-resilient circuits at intermediate nodes incurs reduced hardware costs. It is an attractive option when it is known what nodes of a system are fault-critical and/or fault-susceptible. Fault-resilient circuits for such fine-grained use cases are based on shared logic redundancy. P. Balasubramanian et. al. [Balasubramanian and Naayagi, 2017] address the topic of inserting shared redundant logic to improve the fault-resilience of combinational circuits. The method starts with an SOP (sum of products) or POS (product of sums) form of a combinational circuit which is implemented using a standard cell library. The resulting circuit which is composed of simple and complex logic gates is then analyzed to build truth-cum-fault enumeration tables for all the internal nodes. The tables reveal the faulty combinations of input vectors and faults, i.e., a combination of a fault which causes a primary output error and the corresponding input vector.

Based on such faulty and non-faulty combinations, a metric called fault

조선대학교
CHOSUN UNIVERSITY

masking ratio or FMR is calculated where a higher FMR means the circuit has better fault-tolerance. To improve FMR of the circuit, each node is then analyzed and an appropriate gate is inserted such that the native functionality of the circuit is maintained. Since the new circuit is different, the truth-cum-fault enumeration tables need to be rebuild and FMR calculated again. This effectively makes the problem an optimization problem. Nevertheless, the methodology was only demonstrated for a single 4-input simple combinational circuit and its scalability was not established. Since it is a node-by-node circuit analysis approach, it will be difficult to apply it to large circuits. Only 4 variants of the logic function under test were designed and no data was presented to determine how many circuits with varying degrees of FMR can be designed with the method. A population-based search algorithm will be more suited to synthesize circuits with varying degrees of fault-tolerance to create a high-quality pareto-front. This means more tradeoffs between fault-tolerance and hardware costs will be available to the user.

C. Platform-specific

Other than the works mentioned above, various technology-specific fault-mitigation strategies have been proposed, such as, for field-programmable gate arrays (FPGAs) (see survey [Cheatham, Emmert, and Baumgart, 2006]). Sánchez-Clemente et. al. proposed the use of ATMR in FPGAs using the line-substitution method mentioned before [Sánchez-Clemente, Entrena, and García-Valderas, 2016]. The approach was tested extensively for the B13 benchmark from the ITC’99 set. In the context of evolutionary approaches, the authors in [Garvie and Thompson, 2004] proposed an evolutionary repair method for TMR implemented in FPGAs. In the event of damage to one of the modules,

조선대학교
CHOSUN UNIVERSITY

Table 1: Comparison with related works.

Feature	TMR	P.Bala et. al.	S.Ostanin et. al.	Arifeen et. al.	Sánchez-Clemente et. al.			This work
					Line-testability	*Probability-based*	*Evolutionary design*	(Evolutionary optimization)
Platform-independent	✓	✓	✓	✓	✓	✓	✓	✓
Workload-aware fault-tolerance	×	✓	✓	✓	✓	✓	✓	✓
Scalable	✓	×	×	✓	✓	✓	×	✓
High-quality pareto front	×	×	×	×	×	×	✓	✓
No error-masking schema	×	✓	×	×	×	×	×	✓
Intermediate nodes	×	✓	×	×	×	×	×	✓

the remaining two modules are used as sources of golden data for the fitness function. For small benchmark circuits, a significant improvement in reliability was reported. Salvador et. al. [Salvador et al., 2011] presented an evolvable hardware system for fault-tolerance in FPGAs. The proposed system combines an evolutionary algorithm with the runtime partial dynamic reconfiguration capability offered by modern FPGAs. For testing purposes, fault models were used at two levels of abstraction: i) configuration logic blocks (CLB) and ii) processing elements (PE) or look-up tables (LUT). Since most of the features required by the proposed system are already available on FPGAs, the overhead incurred for self-healing was very low.

Given the scope of this work, we are particulary interested in platform-independent fault-tolerance methods because of their wider applicability. In summary, a methodology for building fault-resilient circuits that offers the following features is desired.

- It should be platform independent.
- It should offer capability for synthesis of circuits whose tolerance to faults targets a typical workload if available.
- It should be scalable.
- It should be able to produce a high-quality pareto front.

조선대학교
CHOSUN UNIVERSITY

- It should avoid the use of an error-masking schema for logic masking in order for the circuits to be replacable at intermediate nodes.

In the context of these desirable, a comparison of the proposed method with related works from the literature is carried out in Table 1.

조선대학교
CHOSUN UNIVERSITY

III. FAULT-RESILIENCE

A. Preliminaries

For fault-resilience profiling of circuits, we must first develop an appropriate definition. In simple terms, the fault-resilience of a circuit can be defined as: *The ability of a circuit to mitigate the effects of faults.* Specifically, we characterize a circuit for fault-resilience according to the following definitions:

Definition 1 *Given a circuit and prescribed input vector and fault sets, the ratio of the number of faults observable at the primary outputs to the cardinality of the fault set called fault probability denoted by P_{fault} gives, in the event of a fault occuring in the circuit, the probability of the fault propagating to the primary outputs.*

Definition 2 *Circuit A is said to be more fault-resilient than B if, A has a lower P_{fault} than B.*

In the following, we illustrate the calculation of P_{fault} for the half adder circuit shown in Fig. 2a. For comparing the fault-tolerance of two circuits in terms of P_{fault}, it is sufficient to use a collapsed fault-set. The collapsed fault-set for the half adder circuit is also given in Fig. 2b. Fault-collapsing [Agrawal, Prasad, and Atre, 2003] is a technique which selects a single fault for a number of equivalent faults and subsequently only the dominating faults from these equivalence collapsed faults. As a result, the total number of faults to process is reduced and, for our purposes, P_{fault} can be computed more quickly. In this case, the faults $(\mathrm{A} \rightarrow \mathrm{C}\ /0)$ and $(\mathrm{B} \rightarrow \mathrm{C}\ /0)$ are redundant and therefore not considered. Note that the fault $(\mathrm{A}\ /0)$ is to be read as the node *A* stuck-at 0.

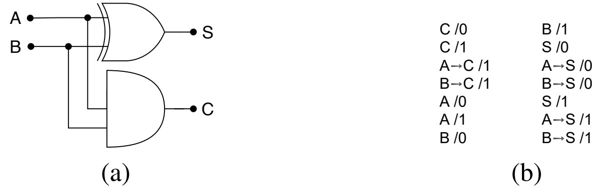


Figure 2: (a) Half adder circuit implemented using XOR and AND gates. (b) The reduced fault-set for the half adder circuit from Fig. 2(a).

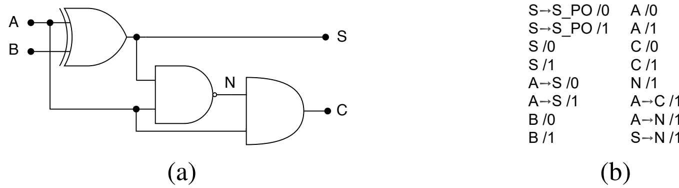


Figure 3: (a) Half adder circuit implemented using XOR, NAND and AND gates. (b) The reduced fault-set for the half adder circuit from Fig. 3(a).

Table 2: Distribution of the faults detectable by all test cases for the half adder circuit from Fig. 2(a).

A	B	C /0	C /1	A→C /1	B→C /1	A /0	A /1	B /0	B /1	S /0	A→S /0	B→S /0	S /1	A→S /1	B→S /1	Count
0	0		*				*		*				*	*	*	6
0	1		*	*			*	*		*		*		*		7
1	0		*		*	*			*	*	*				*	7
1	1	*				*		*			*	*	*			6

Table 3: Distribution of the faults detectable by all test cases for the half adder circuit from Fig. 3(a).

A	B	S→S_PO /0	S→S_PO /1	S /0	S /1	A→S /0	A→S /1	B /0	B /1	A /0	A /1	C /0	C /1	N /1	A→C /1	A→N /1	S→N /1	Count
0	0		*		*		*		*		*		*		*			7
0	1	*		*			*	*			*		*		*			7
1	0	*		*		*			*	*			*	*				7
1	1		*		*	*		*		*		*					*	7

Table 2 lists the distribution of the faults detectable by all possible test cases for the half adder circuit of Fig. 2. We can see that all 14 faults are detectable considering the four test cases. P_{fault} is therefore computed as: $(\frac{14}{14} = 1.0)$. Figure 3 shows an alternate implementation of the half adder along with its collapsed fault-set. Table 3 lists the distribution of the faults detectable for this half adder

조선대학교
CHOSUN UNIVERSITY

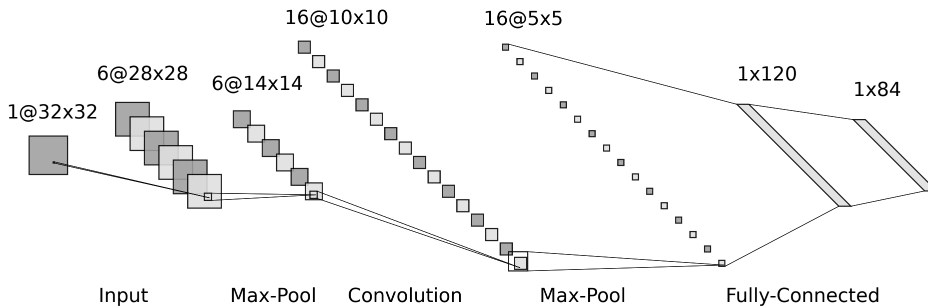


Figure 4: A configuration of the CNN architecture LeNet-5.

circuit. We see that the fault $(A \rightarrow N\ /1)$ is logically masked such that it is not detected by any of the four tests. The P_{fault} for this circuit is computed as: $(\frac{15}{16} = 0.937)$ which is lower compared to P_{fault} of the previous circuit. Thus, the half adder circuit of Fig. 3 is more fault-resilient compared to the half adder shown in Fig. 2. The utility of P_{fault} definition will be recognized in the following section (III-B) and later in chapter V.

B. Need for Fault-resilience

In this section, we highlight the impact of critical faults and that even a single fault can result in considerable output error unacceptable for the application at hand.

Figure 4 shows a configuration of the classic CNN architecture LeNet-5 [LeCun et al., 2015]. We trained the CNN on the MNIST dataset [Deng, 2012] for its 60 000 training images to derive the weights. We then performed a sensitivity analysis of the network through first-order Taylor expansion of the cost function. This allows us to rank the filters in each layer according to how much they contribute to the output. Expansive details on our sensitivity analysis of the LeNet-5 are available in [Hassan, Arifeen, and Lee, 2020]. The analysis showed

조선대학교
CHOSUN UNIVERSITY

Table 4: Adder Profiles.

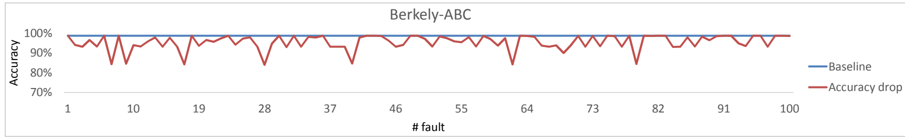
Circuit	Synthesis	PI	PO	Pfault	Area	Level	Gates	Delay (ns)	Power (uW)
signed 8-bit adder	Berkely-ABC	16	9	1.0	53.66	15	39	16.8	235.1
signed 8-bit adder	SYFR	16	9	0.524	114.46	19	76	35.2	642.6

that one of the six filters implementing the first 6@28x28 convolution layer ranks as the most critical. We then implemented the 8-bit signed addition operation of this filter with an adder synthesized using the open verilog synthesis suite Yosys [Wolf, Glaser, and Kepler, 2013] and Berkely-ABC [Brayton and Mishchenko, 2010] for a library of all the basic 2-input gates. The profile of the baseline adder is listed in Table 4.

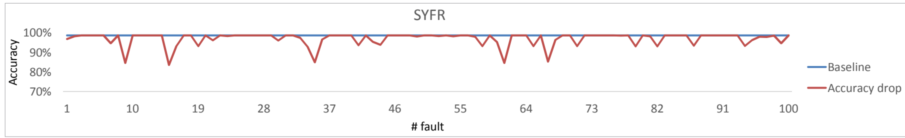
When tested with MNIST database's test data consisting of 10 000 images, the accuracy of the network implementing the accurate adder was 98.65 %. We then injected the baseline adder with a single fault. When tested again, the accuracy of the network dropped to an unacceptable 51.39 %. This is the same CNN in which it is possible to even prune multiple adders from low-ranking filters without any considerable drop in the classification accuracy. However, here is a portion of the CNN which cannot tolerate faulty operation even in the capacity of a single fault. Such critical nodes are good candidates for fine-grained fault-resilient circuits.

We evaluated the performance of the baseline adder against a fault-resilient adder generated by SYFR through fault-injection tests. Note that the adder constructed with SYFR was selected ad-hoc. It is also listed in Table 4. For the evaluation, all the filters were implemented with the adder under test. A filter was selected arbitrarily and a fault was injected in the adder at a random node. The CNN was then tested with the 10 000 test images. The network output accuracy was recorded and the whole process was repeated 100 times. The results of this

조선대학교
CHOSUN UNIVERSITY



(a)



(b)

Figure 5: Performance of the network for 100 injected faults. (a) Baseline adder (Berkely-ABC) and (b) Fault-resilient adder (SYFR).

evaluation are shown in Fig. 5. It is clear from these plots that the adder with better P_{fault} has lower accuracy drops since its plot is less rocky. This gives credibility to the P_{fault} definition used by SYFR.

In conclusion, this example clearly stresses the need for fault-resilient circuits which are easily usable at intermediate nodes without creating any single points of failures unlike TMR which when applied intermediately creates a single point of failure during consensus.

조선대학교
CHOSUN UNIVERSITY

IV. PROPOSED METHODOLOGY

The SYFR design flow as divided into several steps is shown in Fig. 6. In this chapter, we explain key elements of the proposed SYFR approach. In particular, we describe the circuit representation used, the underlying search algorithm for automatic synthesis, and formulate a fast fitness function for our problem. We also describe our proposal on how to tune the search process for reducing the search space size of our problem and therefore improve the speed and efficiency of SYFR. Lastly, we will describe our implementation of the proposed SYFR and what tools were used at each step.

A. Circuit Encoding in the Chromosome

The choice of circuit abstraction are the digital logic gates which have also been adopted in multiple works on synthesis of boolean logic [Vasicek and Sekanina, 2014; Kazarlis, Kalomiros, and Kalaitzis, 2016; Venkataramani et al., 2012; Venkataramani, Roy, and Raghunathan, 2013; Scarabottolo, Ansaloni, and Pozzi, 2018; Schlachter et al., 2017; Arifeen et al., 2016; Arifeen et al., 2018; Hassan et al., 2018; Gomes et al., 2015; Sanchez-Clemente, Entrena, and García-Valderas, 2014; Sanchez-Clemente et al., 2012]. Our choice is based on the convenience of working with this level of abstraction and because it can also be easily related to the low-level circuit implementation. To which end, instead of using the number of gates as a metric to estimate area, we will use relative areas of gates from the following dictionary (BUFF 0.0, NOT 0.67, NAND 1.0, NOR 1.0, AND 1.33, OR 1.33, XOR 2.0, XNOR 1.66) calculated by [Vasicek and Sekanina, 2014] for MCNC library [Sentovich et al., 1992], $V_{dd} = 5$ V and 20 MHz relative to that of a single NAND gate. Thus, in the following text, the *area* of a circuit will be

조선대학교
CHOSUN UNIVERSITY

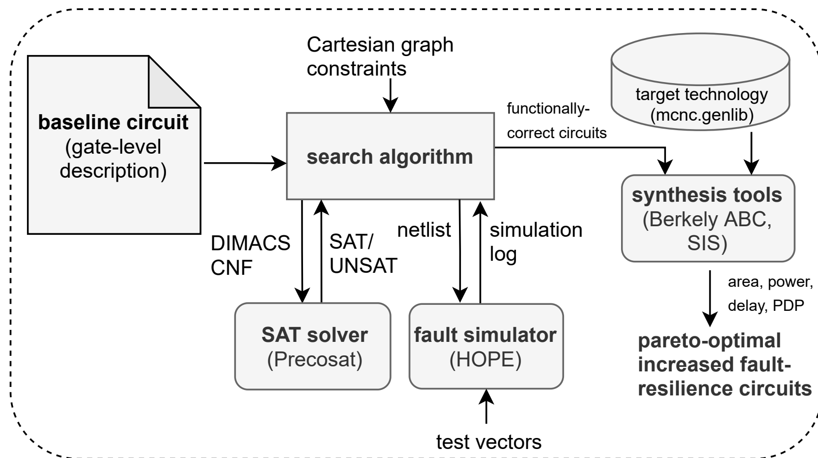


Figure 6: Design flow for the proposed SYFR approach.

given by the sum of the relative areas of gates of the circuit.

Cartesian graph representation from the popular Cartesian genetic programming or CGP [Miller and Harding, 2008] is used in this work for candidate circuit representation. In this representation, an array of n_c columns x n_r rows of programmable nodes models an n_i-input, n_o-output combinational circuit. The gate library G contains the available n_a-input node functions. The primary inputs and all the nodes of the graph are uniquely numbered starting from 0. Another parameter l controls the levels-back connectivity of the encoded graph. Setting l=1 allows a node to get its inputs from a node in the column immdediately preceeding it. If l=n_c is set, a node can get its inputs from any nodes in the columns to its left. Varying the parameters n_c, n_r and l therefore results in various graph topologies.

For processing purposes, the circuits represented in a Cartesian graph are encoded as an ordered list of integers known as a *chromosome*. Given the arity $n_a = 2$, the chromosome contains $n_c.n_r$ triplets (ϕ, N_1, N_2) for a node N_i where ϕ is a processing element from G while N_1 and N_2 are n_a addresses of nodes

조선대학교
CHOSUN UNIVERSITY

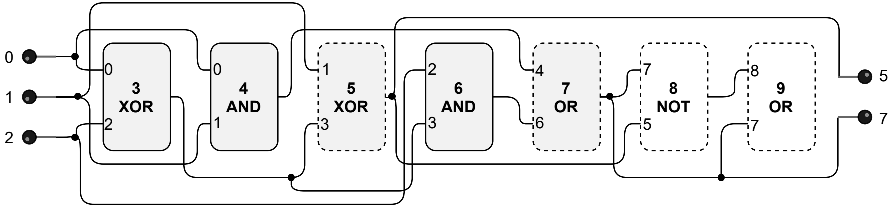


Figure 7: A Full adder represented in a Cartesian graph with parameters G={AND^0,OR^1,XOR^2,NOT^3}, $n_a = 2$, $n_i = 3$, $n_o = 2$, $n_c = 7$, $n_r = 1$.
The chromosome with underlined node functions and highlighted outputs: $(\underline{2},0,2)(\underline{0},0,1)(\underline{2},1,3)(\underline{0},2,3)(\underline{1},4,6)(\underline{3},7,5)(\underline{1},8,7)(\mathbf{5},\mathbf{7})$

from which the node N_i gets its inputs. The tail of the chromosome contains n_o integers which specify the nodes to which the POs are connected. Then, the total number of integers in the chromosome are given by $n_c.n_r(n_a+1)+n_o$. Note that we fixed the graph shape at n_cx1 because this shape allows all the different circuit topologies possible under the given graph constraints. For example, it is possible to encode a a 2x2 circuit topology in a 4x1 grid.

An example of a full adder circuit encoded in a Cartesian graph is shown in Fig. 7. We note that the NOT gate does not obey the arity constraint $n_a = 2$ and therefore during the decoding process, out of the two node inputs N_1 and N_2, the input N_2 is simply ignored. Finally, note that it is also possible to include complex gates such as AOI21 and OAI21 in the gate library G. We however did not consider complex gates for two reasons: (i) The fault simulator HOPE that we are using cannot directly process complex gates (albeit a workaround is possible) and, (ii) to be comparable to similar works on gate-level automatic synthesis of digital circuits such as [Vasicek and Sekanina, 2014] and [Kazarlis, Kalomiros, and Kalaitzis, 2016] which did not use complex gates.

조선대학교
CHOSUN UNIVERSITY

B. Constraining the Cartesian Graph

The evolutionary design of non-trivial circuits such as w-bit multipliers is very difficult since their fitness landscapes are quite rugged [Miller, Job, and Vassilev, 2000]. And then to search for a circuit which is not only functionally correct but also exhibits an improved P_{fault} over the baseline is even more challenging. Therefore, the initial population must be seeded appropriately (evolutionary optimization). Here, the baseline is defined as: *the target circuit synthesized by conventional synthesis tools*. The intention is to synthesize circuits functionally-equivalent to the baseline but more fault-resilient ,i.e., circuits with a lower P_{fault} than baseline.

The most straightforward way of seeding the initial population is to simply add the baseline to the population as n^{th} member. Since the seeded circuit (baseline) will be functionally-correct, it will be sorted to the top and from there, the evolutionary search operators will disseminate the baseline's genetic material to the rest of the population. There is, however, another way of approaching this problem where the fact that we are only looking for functionally-correct circuits can serve as an advantage. This is explained in the following paragraphs.

In most cases, if a circuit A has lower P_{fault} than another circuit B, A will also incur a higher implementation cost (e.g., area, power etc.). Thus, an improvement in fault-resilience comes at the cost of an increase in the hardware resources. With this understanding, and with the premise that seeking increased fault-resilience is of prime importance to the designer, we are not attempting to build circuits that are both more fault-resilient and cost-efficient compared to the baseline. And to that end, the genetic material of the baseline is seeded in all members of the population and kept intact. This is achieved through enforcing additional

조선대학교
CHOSUN UNIVERSITY

constraints on the Cartesian graph.

If $g_{baseline}$ gives the gate count of the baseline and therefore the minimum number of nodes required to encode the baseline, then the nodes occupied by the baseline from N_0 to $N_{g_{baseline}-1}$ given $n_c.n_r > g_{baseline}$ are designated as non-programmable. While, the remaining nodes from $N_{g_{baseline}}$ to $N_{n_c.n_r-1}$ are made available to the search process for improving P_{fault}. This results in an auxiliary circuit which cascades to the baseline and improves P_{fault} of the final circuit. If the baseline implements the logic function $g(x) = y$, then the auxiliary circuit logically acts as a wire or a buffer $h(y) = y$ as shown in Fig. 8. Thus, the desired logic function remains unchanged. Consider the full adder encoded in a 7x1 shaped Cartesian graph from Fig. 7. Under the idea mentioned above

1. Only the colorless nodes (8 and 9) are programmable, i.e. the search process is only allowed to modify the logic function of and/or connections to these nodes.

2. The graph outputs can either be taken from the programmable nodes (8 and 9) or from the colored nodes with dotted boundaries which are supposed to be the POs of the seed (nodes 5 and 7 for the encoded full adder in Fig. 7).

3. This last constraint applies to connections amongst the nodes. The input connections to the colored nodes (3,4,5,6 and 7) as encoded cannot change. The outgoing connections from the solid colored nodes (3,4 and 6) also cannot change. Inputs to a programmable node can be taken from either the colored nodes with dotted boundaries (5 and 7) or from another programmable node while also maintaining the levels-back l and no-feedback constraints mentioned in IV-A.

조선대학교
CHOSUN UNIVERSITY

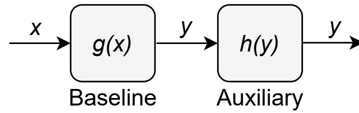


Figure 8: Cascading additional circuitry to enhance the fault-masking property of the baseline. The auxiliary circuit acts as a wire and does not change the intended logic function g.

These graph constraints then serve two purposes: (i) Since the genetic material of the baseline remains intact in all members of the population, it becomes easy to meet the functional-equivalency objective, and (ii) a considerable reduction in the size of the search space is achieved. Since SYFR only has to focus on finding the best auxiliary circuits for logic-masking, the design space exploration becomes easier.

1. Population Initialization

The first generation of population is initialized as follows. The graph size is selected such that the condition $n_c.n_r > g_{baseline}$ is satisfied and each member of the population is seeded with the baseline. The nodes from N_0 to $N_{g_{baseline}-1}$ encode genes of the baseline while the rest of the nodes from $N_{g_{baseline}}$ to $N_{n_c.n_r-1}$ are randomly initialized. The graph POs are connected to the POs of the baseline and thus all members of the initial population are actually fully working solutions. The population is evolved under the graph constraints introduced above. The evolutionary process is not allowed to modify genes of the baseline. Instead, the process focuses on the programmable nodes in search of suitable auxiliary logic that helps improve P_{fault} of the final circuit while maintaining the intended logic function.

조선대학교
CHOSUN UNIVERSITY

C. Search Algorithm

The search is conducted using an advanced variant of the Cuckoo Search (CS) algorithm [Yang and Deb, 2009] known as Cuckoo Search with Genetically Replaced Nests (Cuckoo-GRN) [Oliveira, Oliveira, and Affonso, 2018]. Cuckoo-GRN combines the benefits of genetic algorithms (GA) into the CS algorithm. In Cuckoo-GRN, instead of replacing the worst nests with randomly generated nests, the worst nests are genetically replaced. We applied it to our problem as follows

1. An initial population of size n is created according to IV-B1.
2. The fitness function f is called for each candidate circuit.
3. h genes (or integers) of a circuit selected at random undergo point mutations without violating the constraints defined in IV-B. The random step length h is drawn from a Lévy distribution: Lévy $\sim \; h = t^{-\lambda}, (1 < \lambda \leq 3)$. If the mutated circuit exhibits better fitness than its parent, it replaces its parent in the population. The process is repeated $n.p_a$ times, $(0 < p_a < 1)$.
4. The population is sorted in ascending order of fitness values.
5. $n.p_a$ number of worst candidate circuits are replaced by new circuits created using crossover and mutation. It is a single-point crossover where one of the parents is always the current best candidate while the other parent is selected using a 3-way tournament from the rest of the population. The crossover also maintains the graph constraints (IV-B). The children are further mutated using Lévy flight. Finally, the children replace the worst circuits in the population.

조선대학교 CHOSUN UNIVERSITY

6. Steps 3–5 are repeated until the termination criteria is satisfied.

D. Fitness Function

We intend to achieve two objectives: (i) to synthesize circuits functionally equivalent to the baseline, and (ii) improve their P_{fault} as much as possible. Thus, it is a dual-objective optimization problem. We use a consolidated fast fitness function formed by the weighted sum of the individual objective values. The fitness is defined as a function to be minimized as

$$f = \alpha(SAT) + (1-\alpha)(P_{fault}) \tag{1}$$

In this definition, SAT represents the functional equivalence objective. SAT is a boolean variable as it can only take two values (0 or 1). $SAT = 0$ means the candidate circuit is functionally equivalent to the baseline and vice versa. P_{fault} represents the fault-resilience objective whereas α controls the weight distribution between the two objectives. A successful candidate will have $SAT = 0$ in addition to a lower value of P_{fault} compared to the baseline.

E. Implementation

In this section, we describe our implementation of SYFR and choice of tools thereof. With reference to Fig. 6, we start with a gate-level decription of the target circuit synthesized with conventional synthesis tools, the Yosys synthesis suite [Wolf, Glaser, and Kepler, 2013] and Berkely-ABC [Brayton and Mishchenko, 2010]. This circuit serves as the baseline and is also used to initialize the population. The software is written in the python programming language. We also

조선대학교
CHOSUN UNIVERSITY

Table 5: Algorithm and fitness function parameters.

Parameter	Value
n	20
p	0.3
k_{max}	100 000
α	0.99

use the PyEDA python package [Drake, 2015] for some of the boolean algebra in our implementation. The CS algorithm parameter values used in our setup are listed in Table 5. These values are used as recommended in the literature [Yang and Deb, 2009]. The fitness function parameter α is set at 0.99 so as to very heavily favor the functional-correctness objective.

It is important to be able to calculate the fitness function in as little time as possible. The first objective *SAT* is obtained by creating a *miter* circuit from the candidate and reference circuits for formal equivalence checking. The miter is converted to its conjunctive normal form (CNF) using Tseitin's transformation formulas from [Manich and Figueras, 1997] and the resulting CNF formula is passed as a DIMACS CNF [Prestwich, 2009] to a SAT-solver (precosat [*PrecoSAT*]). This Boolean satisfiability- based equivalence checking approach for formal verification of circuits powered by modern SAT solvers can deliver results quickly including large circuits. More information on the said technique can be found in [Vasicek and Sekanina, 2012] where it is described in greater detail.

Next, we require a tool to process and extract the faults list of the given circuit. The tool should also give the fault observability counts at the POs which are required for calculating P_{fault}. For this purpose, we used the parallel fault simulator HOPE [Lee and Ha, 1996] which proved sufficiently fast to be

조선대학교
CHOSUN UNIVERSITY

incorporated into the search process. During the simulation, we collect and store all the unique functionally-correct circuits. Actually, two circuits can have the same values of P_{fault} but different hardware costs (due to different hardware configurations) and vice versa. Accordingly, to detect circuits with unique hardware configurations, we distinguish circuits based on three parameters: P_{fault}, relative area (IV-A) and, the number of logic levels. This allows us to collect a larger spread of circuits.

After the search algorithm terminates at the given criteria, we process all of the circuits collected for power consumption, delay and PDP. The delay is calculated using Berkely-ABC for the MCNC library. For power consumption, we use the SIS software [Sentovich et al., 1992] which has provision for estimating the dynamic power consumption of a circuit valid for the MCNC library at $V_{dd} = 5$ V and 20 MHz. We then process all the circuits collected for pareto-optimality with respect to two objectives: (i) P_{fault} and (ii) area, power, delay or PDP. For non-dominated sorting, we used the pareto.py program available from [Woodruff and Herman, 2013] which performs an epsilon-nondominated sort [Deb, Mohan, and Mishra, 2005] on the input circuits for the given objectives. Note that all the computations described in this work were performed on a desktop-class Linux machine equipped with an Intel i5-9500 @ 3.00GHz (6 cores) processor and 16GB of RAM.

조선대학교
CHOSUN UNIVERSITY

V. EXPERIMENTAL RESULTS

The test circuits consist of 7 circuits whose hardware profiles are listed in Table 6. All of these circuits were synthesized by conventional synthesis tools (Yosys, Berkely-ABC) and serve as the baseline for SYFR. Of the 7 circuits, the performance of SYFR is studied extensively for addr4u, addr8u and mult4u in V-A. The circuits c432, c1355 and c880 belong to the ISCAS set. They are used to demonstrate the technique's scalability in V-B. Finally, addr4ur is used for a demonstrative purpose in V-C.

The goal of SYFR is to produce functionally-correct circuits showing the lowest possible P_{fault} for a given graph size. SYFR was applied to addr4u, addr8u and mult4u for four graph sizes or the number of columns n_c: $(1.25)g_{baseline}$x1, $(1.5)g_{baseline}$x1, $(1.75)g_{baseline}$x1 and $(2.0)g_{baseline}$x1. At each graph size, 3 independent SYFR runs were conducted, and in total, 12 independent SYFR runs were conducted for each circuit. The search algorithm stops when the predefined number of generations k_{max} is reached. From the description of SYFR in IV-B, it is clear that the best solution in any generation will always be functionally-correct. Thus, instead of reporting fitness values, we only report P_{fault} values in the following.

Table 6: Test circuits profiles.

Circuit	Description	PI	PO	Pfault	Area	Level	Gates	Delay (ns)	Power (uW)
addr4u	unsigned 4-bit adder	8	5	1.0	24.33	7	18	7.9	97.0
addr4ur	unsigned 4-bit adder (carry-in)	9	5	1.0	28.0	9	20	9.9	122.5
addr8u	unsigned 8-bit adder	16	9	1.0	54.65	15	45	15.9	282.2
mult4u	unsigned 4-bit multiplier	8	8	0.996	89.98	17	75	22.8	524.0
c432	27-channel interrupt controller	36	7	0.995	161.76	27	171	31.2	782.5
c1355	32-bit SEC circuit	41	32	1.0	295.63	12	191	17.7	1231.2
c880	8-bit ALU	60	26	0.999	329.68	21	317	23.6	1576.8

조선대학교 CHOSUN UNIVERSITY

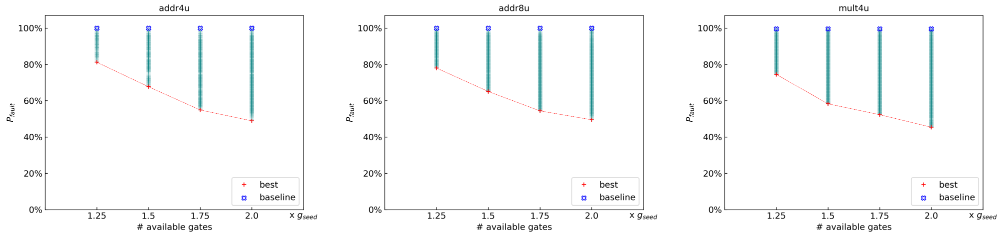


Figure 9: P_{fault} spread of the evolved solutions for addr4u, addr8u and mult4u at different values of n_c.

A. Synthesis Results

Figure 9 shows the P_{fault} values of all the evolved circuits in all runs at the end of evolution. All of the evolved circuits reported are functionally-correct. The baseline is also shown in these figures. Figure 9 shows that the P_{fault} spread becomes larger as the number of components available to SYFR increase. In other words, the number of circuits generated by SYFR increases. The spreads are also dense and connected indicating that SYFR produced circuits at all different levels of P_{fault} from the baseline to the best evolved.

Table 7 lists the best, worst and mean improvements in P_{fault} for the best evolved circuits for all runs at each graph size tested. As n_c is increased, SYFR reports more improvement in P_{fault}. On average, P_{fault} improved by 73.6 % when the number of columns n_c increased from $(1.25)g_{baseline}$ to $(1.5)g_{baseline}$, 32.25 %, from $(1.5)g_{baseline}$ to $(1.75)g_{baseline}$, and only 11 %, from $(1.75)g_{baseline}$ to

Table 7: P_{fault} improvement [%] of the best evolved circuits compared to the baseline at various Cartesian graph columns n_c (higher is better)

	$n_c = (1.25)g_{baseline}$			$n_c = (1.5)g_{baseline}$			$n_c = (1.75)g_{baseline}$			$n_c = (2.0)g_{baseline}$		
	best	worst	mean	best	worst	mean	best	worst	mean	best	worst	mean
addr4u	18.8	15.7	16.7 ± 1.5	32.3	28.9	30.0 ± 1.6	45.1	43.7	44.6 ± 0.7	51.1	44.2	48.2 ± 2.9
addr8u	22.0	18.4	19.9 ± 1.5	34.9	32.0	33.7 ± 1.3	45.6	42.7	44.0 ± 1.2	50.5	46.4	48.7 ± 1.7
mult4u	25.5	19.5	23.1 ± 2.6	41.7	38.5	39.7 ± 1.4	47.7	45.2	46.7 ± 1.1	54.6	51.2	53.4 ± 1.6

조선대학교
CHOSUN UNIVERSITY

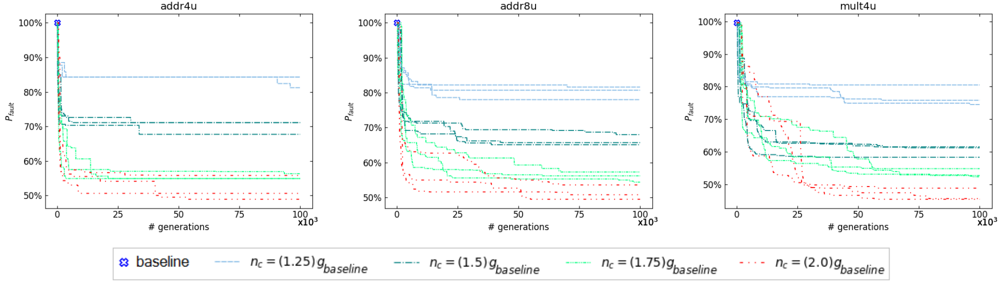


Figure 10: P_{fault} curves of the test circuits: addr4u, addr8u and mult4u.

$(2.0)g_{baseline}$. Thus, as the graph size is increased, the gains in P_{fault} improvement start to reach the point of diminishing returns. This is also observed in the decreasing slope of the dotted line (red) connecting the best evolved circuits in Fig. 9.

The P_{fault} convergence curves are plotted in Fig. 10. Almost all of the runs report a steep decrease in P_{fault}. This means that SYFR has a fast convergence rate. We also observe that the bulk of reduction in P_{fault} was achieved before 50k generations. Furthermore, for the same number of columns n_c, the P_{fault} curves converge to very close values. This is also indicated by the standard deviations reported in Table 7. This is valuable for practice because it means that a single SYFR run almost always provides an optimal-quality solution.

1. Overall Comparison

The curves for mean improvement in P_{fault} for the test results of addr4u, addr8u and mult4u are shown in Fig. 11. Again, for all the number of columns n_c tested, there is a quick improvement in P_{fault}. The curves then taper off and remain quite stable until k_{max} is exhausted. It is reasonable to conclude that typically $k_{max} = 50000$ generations should be enough to obtain high-quality solutions for

조선대학교
CHOSUN UNIVERSITY

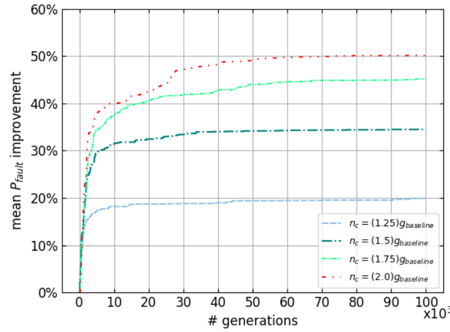


Figure 11: Overall comparison: Mean P_{fault} curves for different n_c values. Mean calculated for addr4u, addr8u and mult4u.

most circuits. Given n_c and a target P_{fault}, the data presented in Fig. 11 can be used to roughly estimate the number of generations required to reach the target P_{fault}.

2. Effect of the Proposed Graph Constraints

The effect of the Cartesian graph constraints proposed in IV-B was studied. A population initialization different to the one described in IV-B1 was tested as follows. The baseline was simply added to the initial population as the nth

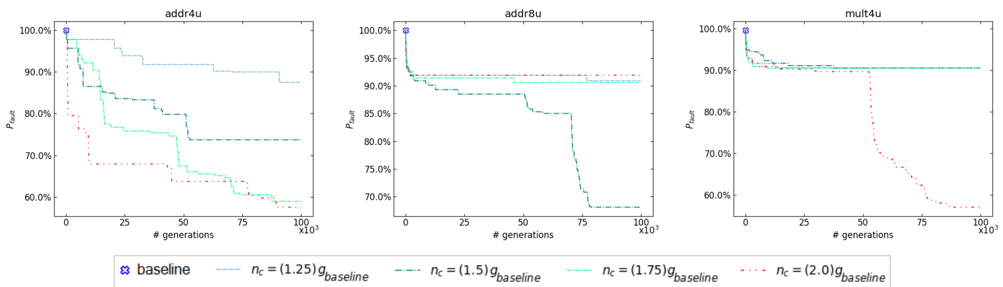


Figure 12: P_{fault} curves of the test circuits: addr4u, addr8u and mult4u. The evolution was carried out in absence of the graph contraints proposed in IV-B.

조선대학교
CHOSUN UNIVERSITY

member and the proposed graph constraints (IV-B) were lifted. The rest of the setup was kept the same as described in V-A. The convergence curves for the objective P_{fault} are shown in Fig. 12.

By comparing Fig. 12 to Fig. 10, one can clearly observe degradation in P_{fault} curves for the corresponding number of columns n_c. It is also clear that the best evolved circuits have worse P_{fault} compared to Fig. 10. This confirms the reduction in search space created by the technique proposed in IV-B. As SYFR only focuses on finding the auxiliary circuit for fault-masking, the task of design space exploration is made easy.

B. Scaling SYFR

In the experiments above (V-A), P_{fault} of a circuit was calculated for all of the input space. However, to scale SYFR, a random set of test vectors must be used for P_{fault} estimation during synthesis. This is synonymous to the use of 10 000 random test vectors for estimating line testabilities in the line-approximation method for generating approximate circuits by Sánchez-Clemente et. al. [Sanchez-Clemente et al., 2016]. To qualify the error-masking schemas constructed, the authors designed a metric called the error probability. For an error-masking schema under test, a fault simulation with 50 000 randomly generated test vectors was performed by means of the fault simulator HOPE [Lee and Ha, 1996]. The fault list used was the same list previously used for line-testability analysis. All the faults contained in the fault list were tested for each input vector. Considering all the faults equally likely, the total error probability (EP) was then computed as the average number of faults detected per input vector, divided by the size of the fault list.

조선대학교
CHOSUN UNIVERSITY

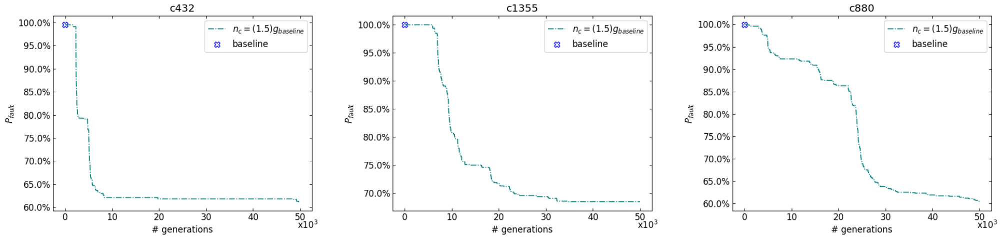
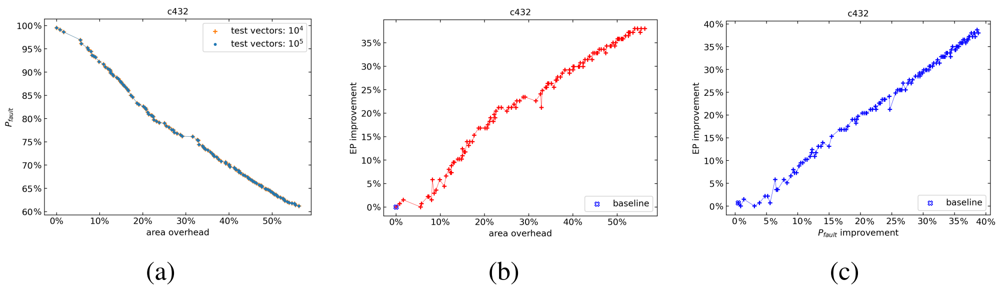


Figure 13: P_{fault} curves of the test circuits: c432, c1355 and c880.



(a) (b) (c)

Figure 14: (a) Mean P_{fault} curves for 50 000 vectors. (b) Improvement EP as a function of area overhead and (c) Relationship between improvements in EP and P_{fault}

We also chose to use 10 000 randomly generated test vectors to estimate P_{fault} during evolution. At the end of the evolution, the best evolved circuits were tested extensively to qualify our choice. This approach for scaling SYFR was tested for the c432, c1355 and c880 test circuits listed in Table 6. In V-A, we observed that SYFR performs consistently and that a single evolutionary run should be sufficient. Moreover, the best tradeoffs between hardware costs and fault-tolerance were achieved at $n_c = (1.75)g_{baseline}$. Therefore we decided to apply SYFR to the test circuits at $n_c = (1.75)g_{baseline}$ for $k_{max} = 50000$ as suggested in V-A1. The P_{fault} convergence curves are shown in Figure 13. Clearly, the reduction in P_{fault} achieved at the end of the run compares with the results presented for the smaller test circuits in V-A (Fig. 10).

조선대학교
CHOSUN UNIVERSITY

After the evolution, the P_{fault} of the best evolved circuits (c432) was recalculated as follows. For each evolved circuit, 100 P_{fault} values were calculated by means of fault simulation with HOPE. Each time 50 000 new random test vectors were generated for the fault simulation. The final P_{fault} was given as the average of the 100 P_{fault} values. Figure 14(a) shows almost no difference in the P_{fault} values calculated during evolution and the recalculated ones. Thus, the use of 10 000 random test vectors during evolution should be reasonable.

For the best evolved circuits (c432), we also calculated the EP metric defined in Sánchez-Clemente et. al. [Sanchez-Clemente et al., 2016]. The improvements in EP as a function of area overhead are presented in Fig. 14(b). The relationship between the two is almost linear. The improvement in EP as a function of improvement in P_{fault} is plotted in Fig. 14(c). The relationship between these two metrics is also linear. In conclusion, this fault-resilient analysis in terms of EP in addition to the P_{fault} metric used in SYFR gives credibility to the logic-masking abilities of the circuits evolved with SYFR.

C. Building Larger Arithmetic Circuits

Smaller adders and multipliers can be used as building blocks for implementing larger adders and multipliers. In this section, we test the hypothesis that whether the larger circuit also exhibits improved P_{fault} if its constituent circuits are fault-resilient. Since it is possible to implement an 8-bit ripple-carry adder simply by concatenating two 4-bit adders, we ad-hoc selected multiple addr4u and addr4ru adders with varying P_{fault} values and concatenated them. The circuit addr4ru (Table 6) is also a 4-bit adder but with a carry-in which is required for

조선대학교
CHOSUN UNIVERSITY

Table 8: P_{fault} values of the 8-bit ripple-carry adders and their constituent 4-bit adders.

addr4u	addr4ru	addr8u
0.949	0.945	0.947
0.902	0.909	0.905
0.849	0.844	0.845
0.802	0.806	0.802
0.754	0.750	0.750
0.706	0.703	0.702
0.654	0.650	0.650
0.601	0.606	0.601

concatenation. The P_{fault} values of the 4-bit adders used for concatenation and the resulting 8-bit adders are listed in Table 8. Indeed from Table 8, we observe that the 8-bit adders implemented have P_{fault} values comparable to their constituent adders. Thus, a combination of fault-resilient circuits will also produce a fault-resilient circuit.

D. Data-Aware Synthesis

With the proposed SYFR, it is also possible to selectively strengthen a circuit according to a typical workload. That is to say, when calculating P_{fault} of the circuit, only select input vectors which are deemed important are considered. Here, this idea is demonstrated via an image sharpening example. Consider an application where images of MRI scans of the brain classified as positive or negative for Alzheimer's disease are to be used for training a classifier. The images however need to be sharpened before being processed by the classifier. The sharpening process includes the application of a high pass filter to the image. The kernel for a high pass filter used in our example is given below

조선대학교
CHOSUN UNIVERSITY

$$K = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

We obtain the sharpened image $Q(i,j)$ as: $Q(i,j) = P_1(i,j) + P_2(i,j)$, where, $P_1(i,j)$ is the original image and $P_2(i,j)$ is the filtered image obtained from convolving $P_1(i,j)$ with the kernel K. Figure 15 shows an example case where the original image and its filtered version are added to produce a sharper image. The 8-bit adder performing this addition is our circuit of interest here. We tried three different adder circuits for these image additions: the baseline unsigned 8-bit addr8u from Table 6, an unsigned fault-resilient 8-bit adder synthesized using SYFR (for all 2^{16} inputs) and an unsigned selectively fault-resilient 8-bit adder synthesized for a typical workload (select vectors).

The dataset used was obtained from kaggle [https://www.kaggle.com/navoneel/; Online; accessed 27-January-2021] and it contains 253 images of MRI scans of the brain. For data-aware synthesis, different approaches to analyze the

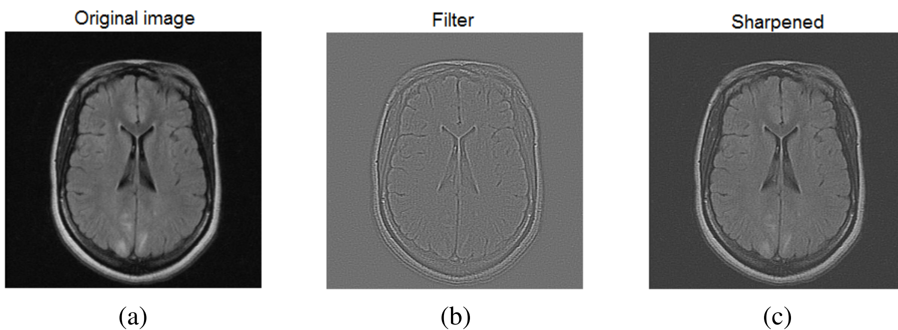


Figure 15: (a) Sample 8-bit grayscale image to be sharpened. (b) Filtered version of the same image and (c) the sharper image obtained by adding the original image to its filtered version.

조선대학교
CHOSUN UNIVERSITY

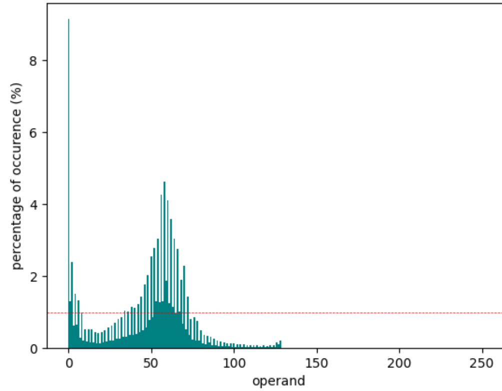
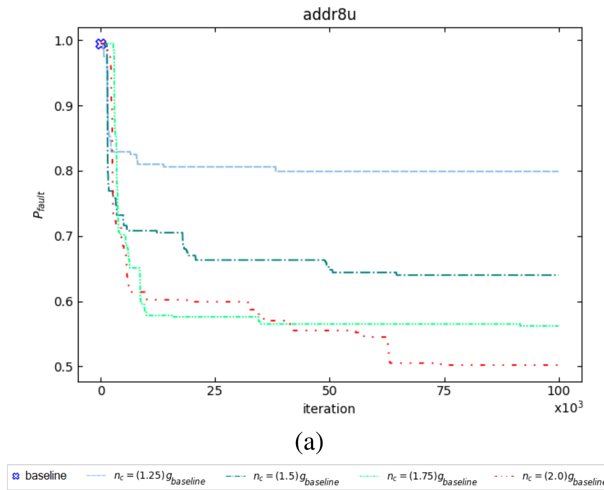


Figure 16: Percentage of occurence of the 8-bit pixel values for the images contained in the dataset. Note that the input images were scaled to range (0-127) to get 8-bit sum at adder outputs.

data to process can be taken. However, it is important to be able to collect the information required for data-aware synthesis in a fast manner since the whole point of data-aware synthesis is to reduce time to synthesis. In our case, we focused on what operands (pixel values) were most likely to be processed by the adder. This is a relatively straight forward approach but very fast to process and effective. First, we analyzed the frequency of occurence of pixel values (i and j) for all the P_1 and P_2 images to be added. The percentage of occurence plot of the operands for these 8-bit grayscale images is shown in Fig. 16.

Based on the percentage of occurence plot, we selected operands with more than 1 % percentage of occurence as important (the horizontal red line in Fig. 16). Then, all input patterns containing these select operands were considered important. This translated into 15,360 input patterns out of the total 2^{16} for the 8-bit addition which is a 77 % reduction in the input space. Thus, we only considered these 15,360 vectors for calculating P_{fault} of the adder circuits during synthesis with SYFR.

조선대학교
CHOSUN UNIVERSITY



(a)

Figure 17: P_{fault} trace plots of addr8u for data-aware synthesis.

The above mentioned process only took us about 15 seconds and we can safely assume that even if the dataset contained a large number of such 8-bit grayscale images, e.g., 10 000, it would still only take less than 10 minutes to obtain the occurence frequencies of the pixel values. In case the data to process is unknown, one could analyze a sufficient amount of dataflow at the adder inputs to confidently estimate the likelihood of each operand and then decide on what operands should be selected. Note that for the same circuit, P_{fault} value will vary if the prescribed test vectors are different. For example, the baseline addr8u from Table 6 has a P_{fault} value of 0.995 instead of the listed 1.0 when considering these select input vectors.

SYFR was applied with the selected input patterns for addr8u. This is done by passing only the selected patterns to the fault simulator which then returns fault detectability counts for the patterns provided. We are primarily looking to improve time to synthesis. The P_{fault} curves in Fig. 17 show a faster convergence compared to the curves in Fig. 10 for corresponding values of n_c.

조선대학교
CHOSUN UNIVERSITY

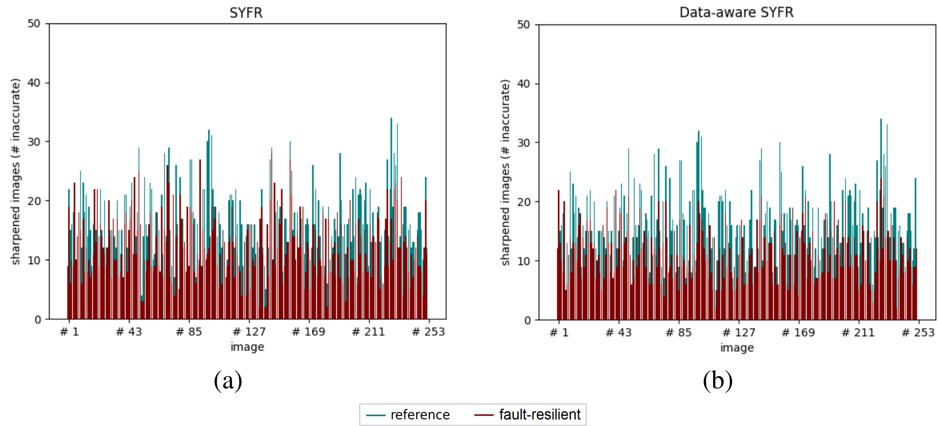


Figure 18: Number of images processed erroneously by the adders.

Next, to assess the quality of data-aware synthesis, we performed fault-injection tests and evaluated the performance of two 8-bit adders both of which had a P_{fault} value of 0.70 (arbitrary selection) where one of them was synthesized for all 2^{16} vectors while the other was picked out of the circuits collected from the data-aware synthesis runs. For each image, we added it to its filtered version 100 times using the adder under test while the adder was injected with a fault at a randomly selected node. We then compared the sharpened output image with the reference image. If there was any difference between the pixel values of the two images, we counted it as an error. For example, image #1 was processed incorrectly 6 times out of 100 by addr8u. Thus, we obtained these error counts for all 253 images of the dataset which are plotted in Fig. 18.

From these plots, we can clearly observe lower error counts in case of these 30 % more fault-resilient adders (red) establishing their better performance. We also see that both adders perform comparably for the given dataset, but the data-aware adder is slightly better and has the lower time to synthesis advantage. In total, the baseline addr8u processed 4 011 images incorrectly while the adders from

조선대학교
CHOSUN UNIVERSITY

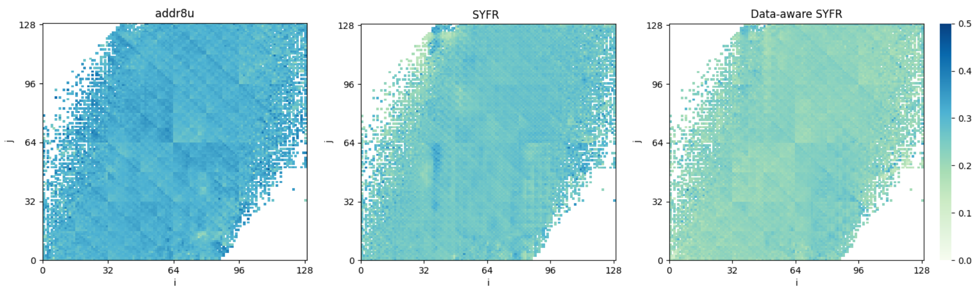


Figure 19: Pixel values processed incorrectly by the adders. The color intensity represents the error magnitude.

general and data-aware synthesis processed 3,130 and 2,860 images incorrectly, respectively.

Figure 18 only reveals information about the number of erroneous images and not anything about the number of erroneous pixels overall. The output pixel values of all the operands that were processed by each of the three adders were checked against the reference value and the results are shown in the form of heatmaps in Fig. 19. For an operand combination, its erroneous count is divided by its occurence count. In the case of addr8u for example, the pixel combination $(i, j) = (128, 128)$ appeared 1,900 times (occurence count) and it was processed incorrectly 510 times due to the faults injected in the adder. Therefore, 510 divided by 1,900 gives 0.27 and this is what the color intensities in Fig. 19 represent.

In Fig. 19, we see that the heatmaps of the two fault-resilient adders have clearly lighter color intensities with the data-aware fault-tolerant adder (rightmost) being arguably better than the other fault-tolerant adder (middle). The operands beyond 128 were emitted from these figures since they are absent in the dataset under test as Fig. 16 shows. Similarly, the white spaces in these heatmaps are also operand combinations which are absent in the images processed.

조선대학교
CHOSUN UNIVERSITY

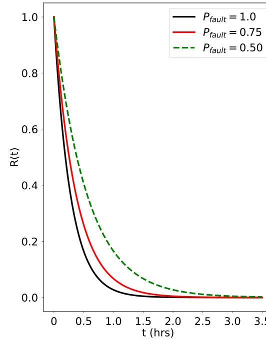


Figure 20: Reliability curves for an arbitrary circuit at different values of P_{fault}.

E. Effect of P_{fault} on Circuit Reliability

Reliability-oriented applications require reliable systems and a reliable system is built from reliable components. SYFR enables synthesis of reliable circuits which can be used to implement reliable systems and subsystems. In this section, we will observe the trends in circuit reliability at different values of P_{fault} through a simple example.

The reliability $R(t)$ of a single module is given by exponential failure law [Dubrova, 2008] as $e^{-\lambda t}$. When P_{fault} is accounted for, the reliability of a circuit is given by $e^{-P_{fault}\lambda t}$ on the assumption that P_{fault} and the failure rate λ are constant. The estimation of λ depends upon the type of technology being used, the circuit size, and some other factors (e.g., environmental). Assuming an arbitrary value of $\lambda = 0.001$ for a circuit, the reliability curves for different values of P_{fault} against time t are presented in Fig. 20.

From Fig. 20, it is clear that a circuit with a lower value of P_{fault} is more reliable. We observe that the gap between the different reliability curves is greater initially which then reduces with time i.e., the reliability of all systems decreases with time. Eventually, a fault-resilient circuit will not be more reliable than a

조선대학교
CHOSUN UNIVERSITY

non-fault-resilient circuit as its reliability reaches zero. Thus, with P_{fault} being considered in the reliability model, a user can estimate as to what P_{fault} value will be required to maintain the reliability of a given circuit within the required bounds until mission duration.

조선대학교
CHOSUN UNIVERSITY

VI. ReCkt: A Library of Reliable Adders and Multipliers

For benchmarking purposes, all the non-dominated adders and multipliers collected from the synthesis runs (V.A) are available online under the MIT license. The library ReCkt [Afzaal, 2021] contains all these circuits in Verilog HDL format. Each file contains the verilog netlist of the circuit along with a description of its hardware costs and P_{fault} values. The file name identifies the type of arithmetic circuit and its bit-width. The circuits available are non-dominated with respect to P_{fault} and area, power, delay, or PDP, the scatter plots of which are attached in Appendix A.

조선대학교
CHOSUN UNIVERSITY

VII. SUMMARY AND CONCLUSIONS

The results presented in this paper certainly place the proposed approach as a competitive methodology for designing fault-resilient circuits. SYFR is a complete systematic method for automated synthesis of fault-resilient circuits at the gate-level without considering technology-specifics. Since SYFR is based on evolutionary computation, it is able to provide a multitude of tradeoffs in comparison to the conventional synthesis tools. The circuits evolved can easily be inserted at intermediate nodes without creating any single points of failure.

The superior performance of fault-resilient circuits at intermediate nodes was demonstrated through an application on artificial neural networks. Results give credibility to the fault-resilience metric P_{fault} used by SYFR to characterize a circuit's tolerance to faults. The majority of SYFR's synthesis success is attributed to the proposed strategy for constraining the Cartesian graph. The graph constraints made the evolution much easier as confirmed by experimental results. It was also demonstrated that smaller fault-resilient arithmetic circuits can be stacked to build large adders and multipliers which are also fault-resilient.

Another benefit of SYFR was that it does not require any special computational resources to execute. A modern desktop-class computer can easily implement SYFR since all of our experiments were conducted on a desktop-class machine.

For future research, we plan to investigate ways to utilize circuits from ReCkt library to build larger arithmetic circuits in an optimal way. We also intend to investigate the evolutionary synthesis of error-resilient circuits. That is to say, there are different error metrics which are application-oriented. For example, PSNR is a metric which is used to gauge the quality of an image. Our goal

조선대학교
CHOSUN UNIVERSITY

would be to apply evolutionary circuit synthesis in such a way that the circuits are evolved for improved PSNR performance in fault conditions.

조선대학교
CHOSUN UNIVERSITY

PUBLICATIONS

1. Umar Afzaal, Abdus Sami Hassan, Muhammad Usman and Jeong A. Lee. ”On the Evolutionary Synthesis of Fault-resilient Arithmetic Circuits”. IEEE Transactions on Evolutionary Computation. (2021) (Under review at the time of thesis submission)

2. Umar Afzaal, Abdus Sami Hassan, and Jeong-A. Lee. ”Improved error detection performance of logic implication checking in FPGA circuits.” Microprocessors and Microsystems 78 (2020): 103179.

3. Umar Afzaal, and Jeong-A. Lee. ”Trading the Reliability of Approximate TMR in FPGAs with the Cost of Mitigation.” 2020 23rd Euromicro Conference on Digital System Design (DSD). IEEE, 2020.

4. Abdus Sami Hassan, Umar Afzaal, Tooba Arifeen, and Jeong A. Lee. ”Input-Aware Implication Selection Scheme Utilizing ATPG for Efficient Concurrent Error Detection.” MDPI Electronics 7, no. 10 (2018): 258.

5. Inayat Ullah, Zahid Ullah, Umar Afzaal, and Jeong-A. Lee. ”DURE: An energy-and resource-efficient TCAM architecture for FPGAs with dynamic updates.” IEEE Transactions on Very Large Scale Integration (VLSI) Systems 27, no. 6 (2019): 1298-1307.

6. Umar Afzaal, and Jeong-A. Lee. ”Low-cost Hardware Redundancy for Fault-mitigation in Power constrained IoT Systems.” 2020 International Conference on Information and Communication Technology Convergence (ICTC). IEEE, 2020.

조선대학교
CHOSUN UNIVERSITY

BIBLIOGRAPHY

Afzaal, Umar (2021). *ReCkt: A library of reliable adders and multipliers*. https://github.com/umar-afzaal/ReCkt. [Online; accessed 18-February-2021].

Afzaal, Umar and Jeong-A Lee (2018). "A Self-checking TMR Voter for Increased Reliability Consensus Voting in FPGAs". In: *IEEE Transactions on Nuclear Science* 65.5, pp. 1133–1139.

— (2020). "Low-cost Hardware Redundancy for Fault-mitigation in Power-constrained IoT Systems". In: *2020 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, pp. 60–62.

Agrawal, Vishwani D, AVSS Prasad, and Madhusudan V Atre (2003). "Fault Collapsing via Functional Dominance". In: *ITC*. Citeseer, pp. 274–280.

Arifeen, Tooba, Abdus Sami Hassan, and Jeong-A Lee (2019). "A fault tolerant voter for approximate triple modular redundancy". In: *Electronics* 8.3, p. 332.

— (2020). "Approximate triple modular redundancy: A survey". In: *IEEE Access* 8, pp. 139851–139867.

Arifeen, Tooba et al. (2016). "Probing Approximate TMR in Error Resilient Applications for Better Design Tradeoffs". In: *2016 Euromicro Conference on Digital System Design (DSD)*. IEEE, pp. 637–640.

— (2018). "Input Vulnerability-aware Approximate Triple Modular Redundancy: Higher Fault Coverage, Improved Search Space, and Reduced Area Overhead". In: *Electronics Letters* 54.15, pp. 934–936.

Balasubramanian, P and RT Naayagi (2017). "Redundant logic insertion and fault tolerance improvement in combinational circuits". In: *2017 International Conference on Circuits, System and Simulation (ICCSS)*. IEEE, pp. 6–13.

조선대학교
CHOSUN UNIVERSITY

Brayton, Robert and Alan Mishchenko (2010). "ABC: An academic industrial-strength verification tool". In: *International Conference on Computer Aided Verification*. Springer, pp. 24–40.

Cheatham, Jason A, John M Emmert, and Stan Baumgart (2006). "A survey of fault tolerant methodologies for FPGAs". In: *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 11.2, pp. 501–533.

Deb, Kalyanmoy, Manikanth Mohan, and Shikhar Mishra (2005). "Evaluating the ε-domination based multi-objective evolutionary algorithm for a quick computation of Pareto-optimal solutions". In: *Evolutionary computation* 13.4, pp. 501–525.

Deng, Li (2012). "The mnist database of handwritten digit images for machine learning research [best of the web]". In: *IEEE Signal Processing Magazine* 29.6, pp. 141–142.

Drake, Chris (2015). "Pyeda: Data structures and algorithms for electronic design automation". In: *Proc. 14th Python in science conference (SciPy)*, pp. 26–31.

Dubrova, Elena (2008). "Fault Tolerant Design: An Introduction". In: *Department of Microelectronics and Information Technology, Royal Institute of Technology, Stockholm, Sweden*, pp. 22–23.

Garvie, Michael and Phil Husbands (2019). "Automatic Synthesis of Totally Self-Checking Circuits". In: *arXiv preprint arXiv:1901.07023*.

Garvie, Miguel and Adrian Thompson (2004). "Scrubbing away transients and jiggling around the permanent: Long survival of FPGA systems through evolutionary self-repair". In: *Proceedings. 10th IEEE International On-Line Testing Symposium*. IEEE, pp. 155–160.

조선대학교
CHOSUN UNIVERSITY

Gomes, Iuri AC et al. (2015). “Using Only Redundant Modules with Approximate Logic to Reduce Drastically Area Overhead in TMR”. In: *2015 16th Latin-American Test Symposium (LATS)*. IEEE, pp. 1–6.

Hassan, Abdus Sami, Tooba Arifeen, and Jeong-A Lee (2020). “Data Footprint Reduction in DNN Inference by Sensitivity-Controlled Approximations with Online Arithmetic”. In: *2020 23rd Euromicro Conference on Digital System Design (DSD)*. IEEE, pp. 534–541.

Hassan, Abdus Sami et al. (2018). “Generation Methodology for Good-Enough Approximate Modules of ATMR”. In: *Journal of Electronic Testing* 34.6, pp. 651–665.

Kazarlis, SA, J Kalomiros, and V Kalaitzis (2016). “A Cartesian Genetic Programming Approach for evolving Optimal Digital Circuits”. In: *Journal of Engineering Science & Technology Review* 9.5.

Khoshavi, Navid, Rizwan A Ashraf, and Ronald F DeMara (2014). “Applicability of Power-gating Strategies for Aging Mitigation of CMOS Logic Paths”. In: *Circuits and Systems (MWSCAS), 2014 IEEE 57th International Midwest Symposium on*. IEEE, pp. 929–932.

Lala, Parag K (2001). *Self-checking and fault-tolerant digital design*. Morgan Kaufmann.

LeCun, Yann et al. (2015). “LeNet-5, convolutional neural networks”. In: *URL: http://yann. lecun. com/exdb/lenet* 20.5, p. 14.

Lee, Hyung Ki and Dong Sam Ha (1996). “HOPE: An Efficient Parallel Fault Simulator for Synchronous Sequential Circuits”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 15.9, pp. 1048–1058.

조선대학교
CHOSUN UNIVERSITY

Manich, Salvador and Joan Figueras (1997). “Maximizing the Weighted Switching Activity in Combinational CMOS Circuits under the Variable Delay Model”. In: *Proceedings European Design and Test Conference. ED & TC 97*. IEEE, pp. 597–602.

Miller, Julian F, Dominic Job, and Vesselin K Vassilev (2000). “Principles in the evolutionary design of digital circuits—Part II”. In: *Genetic programming and evolvable machines* 1.3, pp. 259–288.

Miller, Julian Francis and Simon L Harding (2008). “Cartesian Genetic Programming”. In: *Proceedings of the 10th annual conference companion on Genetic and evolutionary computation*, pp. 2701–2726.

Oliveira, Victoria YM de, Rodrigo MS de Oliveira, and Carolina M Affonso (2018). “Cuckoo Search approach enhanced with genetic replacement of abandoned nests applied to optimal allocation of distributed generation units”. In: *IET Generation, Transmission & Distribution* 12.13, pp. 3353–3362.

Ostanin, Sergey, Irina Kirienko, and V Lavrov (2015). “A fault-tolerant combinational circuit design”. In: *2015 IEEE East-West Design & Test Symposium (EWDTS)*. IEEE, pp. 1–4.

Polian, Ilia and John P Hayes (2010). “Selective hardening: Toward cost-effective error tolerance”. In: *IEEE Design & Test of Computers* 28.3, pp. 54–63.

PrecoSAT. http://fmv.jku.at/precosat/. [Online; accessed 22-July-2021].

Prestwich, Steven David (2009). “CNF Encodings.” In: *Handbook of satisfiability* 185, pp. 75–97.

Quinn, Heather et al. (2016). “Robust duplication with comparison methods in microcontrollers”. In: *IEEE Transactions on Nuclear Science* 64.1, pp. 338–345.

조선대학교
CHOSUN UNIVERSITY

Salvador, Ruben et al. (2011). “Fault tolerance analysis and self-healing strategy of autonomous, evolvable hardware systems”. In: *2011 International Conference on Reconfigurable Computing and FPGAs*. IEEE, pp. 164–169.

Sanchez-Clemente, Antonio, Luis Entrena, and Mario García-Valderas (2014). “Error Masking with Approximate Logic Circuits Using Dynamic Probability Estimations”. In: *2014 IEEE 20th International On-Line Testing Symposium (IOLTS)*. IEEE, pp. 134–139.

Sanchez-Clemente, Antonio et al. (2012). “Logic Masking for SET Mitigation Using Approximate Logic Circuits”. In: *2012 IEEE 18th International On-Line Testing Symposium (IOLTS)*. IEEE, pp. 176–181.

Sanchez-Clemente, Antonio J et al. (2016). “Error mitigation using approximate logic circuits: A comparison of probabilistic and evolutionary approaches”. In: *IEEE Transactions on Reliability* 65.4, pp. 1871–1883.

Sánchez-Clemente, Antonio José, L Entrena, and Mario García-Valderas (2016). “Partial TMR in FPGAs Using Approximate Logic Circuits”. In: *IEEE Transactions on Nuclear Science* 63.4, pp. 2233–2240.

Scarabottolo, Ilaria, Giovanni Ansaloni, and Laura Pozzi (2018). “Circuit Carving: A Methodology for the Design of Approximate Hardware”. In: *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, pp. 545–550.

Schlachter, Jeremy et al. (2017). “Design and Applications of Approximate Circuits by Gate-level Pruning”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25.5, pp. 1694–1702.

Sentovich, Ellen M et al. (1992). “SIS: A system for sequential circuit synthesis”. In:

조선대학교
CHOSUN UNIVERSITY

Vasicek, Zdenek and Lukas Sekanina (2012). “On Area Minimization of Complex Combinational Circuits Using Cartesian Genetic Programming”. In: *2012 IEEE Congress on Evolutionary Computation*. IEEE, pp. 1–8.

— (2014). “Evolutionary Approach to Approximate Digital Circuits Design”. In: *IEEE Transactions on Evolutionary Computation* 19.3, pp. 432–444.

Venkataramani, Swagath, Kaushik Roy, and Anand Raghunathan (2013). “Substitute-and-Simplify: A Unified Design Paradigm for Approximate and Quality Configurable Circuits”. In: *2013 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, pp. 1367–1372.

Venkataramani, Swagath et al. (2012). “SALSA: Systematic Logic Synthesis of Approximate Circuits”. In: *DAC Design Automation Conference 2012*. IEEE, pp. 796–801.

Wolf, Clifford, Johann Glaser, and Johannes Kepler (2013). “Yosys-a free Verilog synthesis suite”. In: *Proceedings of the 21st Austrian Workshop on Microelectronics (Austrochip)*.

Woodruff, Matthew and Jon Herman (2013). *pareto.py: a epsilon-nondomination sorting routine*. https://github.com/matthewjwoodruff/pareto.py. [Online; accessed 27-January-2021].

Yang, Xin-She and Suash Deb (2009). “Cuckoo Search via Lévy Flights”. In: *2009 World congress on nature & biologically inspired computing (NaBIC)*. IEEE, pp. 210–214.

조선대학교
CHOSUN UNIVERSITY

APPENDIX A: ReCkt Library

From the ReCkt library of reliable adders and multipliers [Afzaal, 2021].

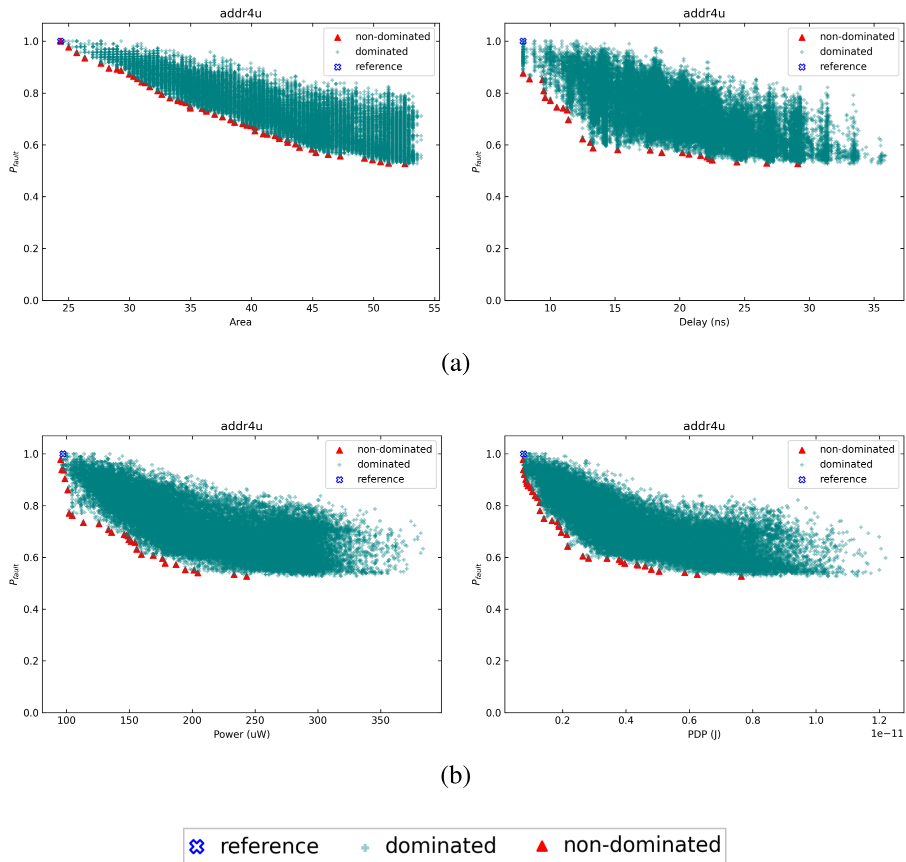
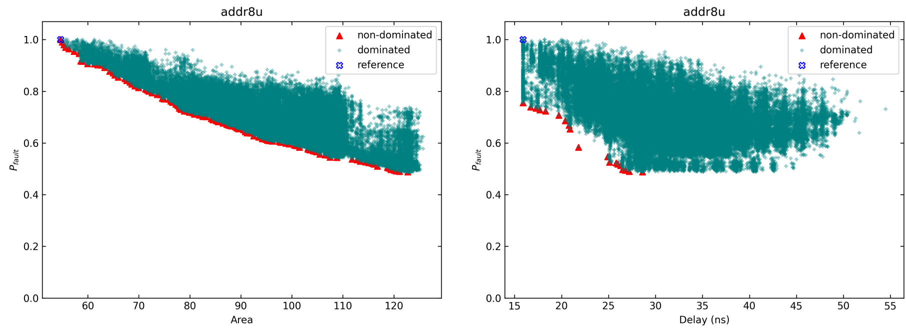


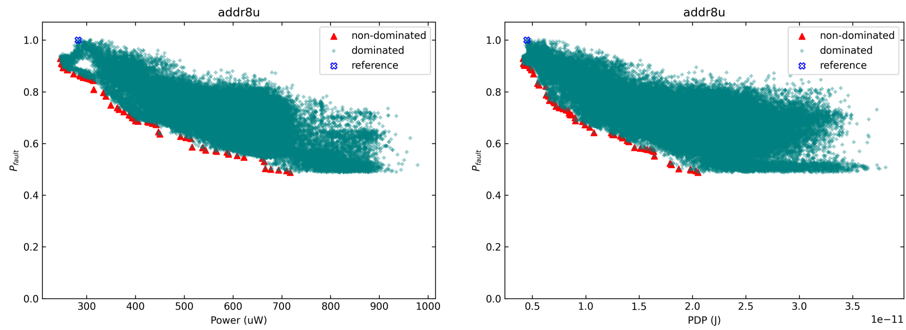
Figure 21: Pareto-optimal addr4u circuits with respect to (i) P_{fault} and (ii) area, power, delay and PDP.

조선대학교
CHOSUN UNIVERSITY

From the ReCkt library of reliable adders and multipliers [Afzaal, 2021].



(a)



(b)

Figure 22: Pareto-optimal addr8u circuits with respect to (i) P_{fault} and (ii) area, power, delay and PDP.

조선대학교
CHOSUN UNIVERSITY

From the ReCkt library of reliable adders and multipliers [Afzaal, 2021].

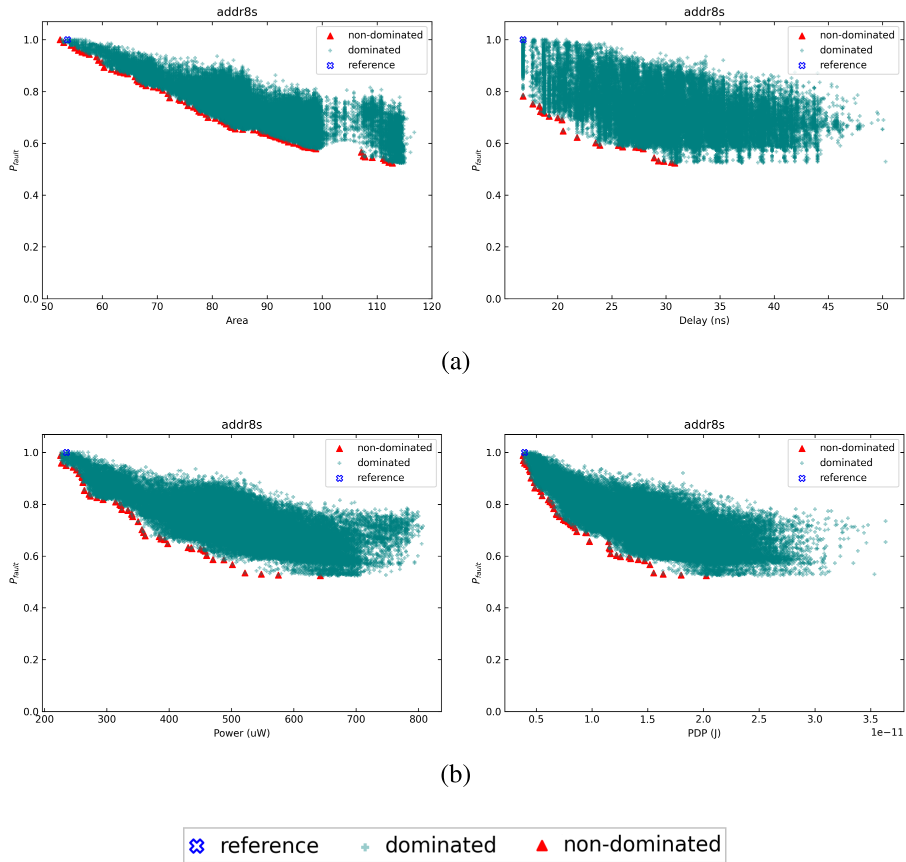


Figure 23: Pareto-optimal addr8s circuits with respect to (i) P_{fault} and (ii) area, power, delay and PDP.

조선대학교
CHOSUN UNIVERSITY

From the ReCkt library of reliable adders and multipliers [Afzaal, 2021].

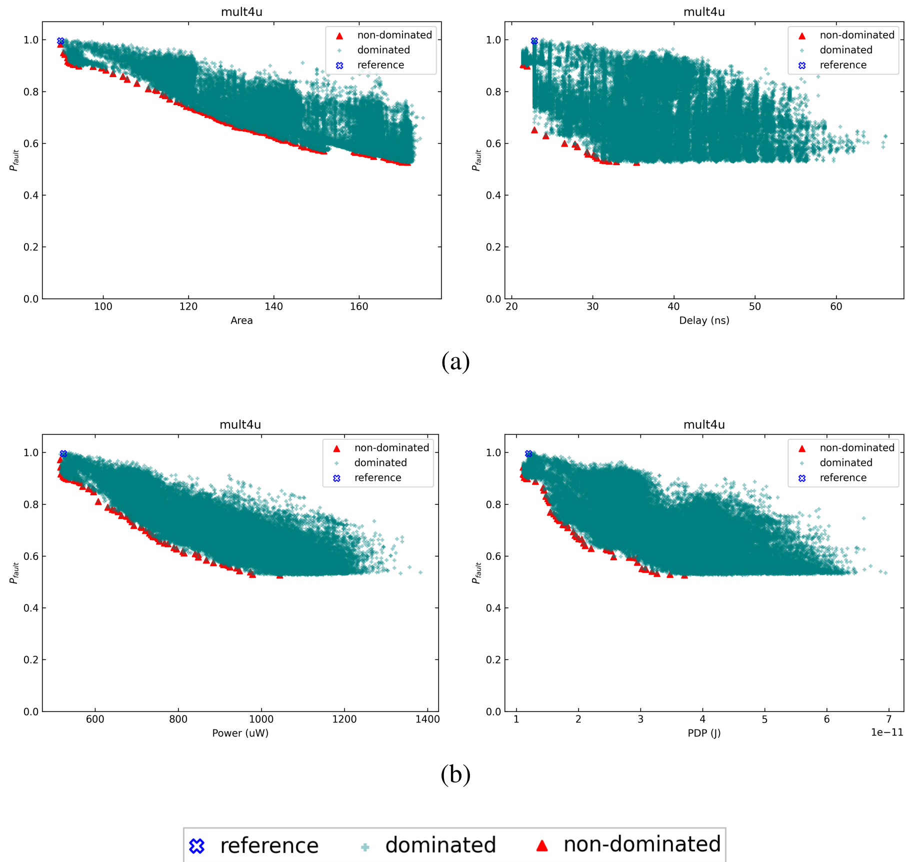


Figure 24: Pareto-optimal mult4u circuits with respect to (i) P_{fault} and (ii) area, power, delay and PDP.