

**Cookie-Cutter: Achieving Defect/Fault Tolerance For Large-Scale Systems with Highly  
Unreliable Components**

BY

SOUMYA BANERJEE

B.Tech., West Bengal University of Technology, 2011

M.S., University of Illinois at Chicago, 2016

THESIS

Submitted as partial fulfillment of the requirements  
for the degree of Doctor of Philosophy in Electrical and Computer Engineering  
in the Graduate College of the  
University of Illinois at Chicago, 2017

Chicago, Illinois

Defense Committee:

Wenjing Rao, Chair and Advisor

Miloš Žefran

Zhichun Zhu

Amit Ranjan Trivedi

John Lillis, Computer Science

Copyright by  
Soumya Banerjee  
2017

*To my parents...*

## ACKNOWLEDGMENTS

Over the last 6 years of my PhD program, I've had the opportunity to come into contact with some amazing people, many of whom positively influenced my research, directly and indirectly.

First of all, I would like to thank my adviser, Prof. Wenjing Rao, for her constant and unwavering support. When I started off with my PhD, my perception was that, as a PhD, my only job is to do research and obtain results. Prof. Rao taught me in addition to the aforementioned goal, how important it also is to present my research to the scientific community. Her emphasis on critical thinking for reviewing various papers has helped me in both research and other aspects of my life.

I would also like to thank Prof. Milos Zefran for agreeing to be a part of my PhD committee and helping me out of certain difficult situations I faced as a PhD student. I extend my gratitude to Prof. Zhichun Zhu for serving as a member of my PhD committee, as well as for the interesting and informative classes she taught on Computer Architecture. I thank Prof. Amit Trivedi and Prof. John Lillis for agreeing to serve in my PhD committee.

Working with Soroush, Jian and Paolo at UIC for the last 5 years have been an amazing experience. You guys are awesome!

As a PhD student, there comes numerous challenges, both academic as well as personal. So, thank you Ketaki, Riddha, Adita, Nirupam, Satabdi for being there for me during my most difficult times. Most importantly, thanks to Raika for cheering me up!

## **ACKNOWLEDGMENTS (Continued)**

Last but not the least, without my family's support, I couldn't have achieved this milestone. So, Ma (my mother), Baba (my father), Didi (my elder sister), Pakhi (my niece) and everyone in my family who has stood by me over the years: I dedicate my work to all of you!

SB

## **PREFACE**

This body of work is an intellectual property of Soumya Banerjee. The works discussed here was conducted at the University of Illinois at Chicago. The work has been presented and published in (Banerjee and Rao, 2016), (Banerjee and Rao, 2017a), (Banerjee and Rao, 2015), (Banerjee and Rao, 2017b). The research work was funded by NSF Grant CNS-1149661.

Soumya Banerjee  
September 6, 2017

## **CONTRIBUTION OF AUTHORS**

The contents of Chapter 2 was published in (Banerjee and Rao, 2016), (Banerjee and Rao, 2017a). Prof. Wenjing Rao was the PI for this part of the work. I was responsible for conceptualizing the idea, performing the experiments and writing the papers.

The contents of Chapter 3 was published in (Banerjee and Rao, 2015), (Banerjee and Rao, 2017b). Prof. Wenjing Rao was the PI for this part. I was responsible to coming up with the idea, formalizing the problem, proving all the lemmas and theorems, designing the algorithms, as well as performing the experiments and composing the papers.

## TABLE OF CONTENTS

<u>CHAPTER</u>		<u>PAGE</u>
<b>1</b>	<b>INTRODUCTION . . . . .</b>	<b>1</b>
1.1	Defect Tolerance and Design Space Exploration via Sparsity in Parallel Prefix Adders . . . . .	2
1.2	Formalization of Basic Replacement Chain Algebra to Arbitrary Many-Processor System . . . . .	3
1.3	Organization . . . . .	5
<b>2</b>	<b>GENERALIZATION OF KOGGE-STONE ADDER GROUP SEGREGATION PROPERTY . . . . .</b>	<b>7</b>
2.1	Introduction to Parallel Prefix Adders . . . . .	7
2.2	Kogge-Stone Adders and Group Segregation Property . . . . .	9
2.3	Defect Tolerant Parallel Prefix Adders . . . . .	13
2.3.1	Motivations and Previous Works . . . . .	13
2.3.2	The Proposed Defect Tolerant Design . . . . .	15
2.3.3	Experimental Results . . . . .	18
2.3.3.1	Reliability Analysis . . . . .	20
2.3.3.2	Expansion of Reliable PPA Design Space Choices . . . . .	22
2.3.3.3	Delay Variation in the Proposed Scheme . . . . .	24
2.4	Sparse Parallel Prefix Adders . . . . .	25
2.4.1	Motivations and Previous Works . . . . .	25
2.4.2	Design Approach for Sparse Parallel Prefix Adders . . . . .	27
2.4.2.1	Examples . . . . .	27
2.4.2.2	Group-Segregation designs for $k$ -Sparse PPA's . . . . .	29
2.4.2.3	Residual GP Restoration designs for $k$ -Sparse PPA's . . . . .	30
2.4.2.4	Overall Design . . . . .	31
2.4.3	Associated Metrics . . . . .	33
2.4.4	Experimental Results . . . . .	35
<b>3</b>	<b>FORMALIZATION OF REPLACEMENT CHAIN ALGEBRA IN ARBITRARY MANY-PROCESSOR SYSTEMS . . . . .</b>	<b>39</b>
3.1	Previous Works and Motivations . . . . .	39
3.2	Introduction to the Proposed 2-Layered Task-PE Model . . . . .	44
3.2.1	Formal Definition of the Task-PE Model . . . . .	45
3.3	Necessary and Sufficient Conditions for Guaranteed $k$ -FT . . . . .	48
3.3.1	Replacement Graph Changes after a Successful Repair: Replacement Chain Algebra . . . . .	48
3.3.2	Repair Chain Independence of Task-PE Model . . . . .	51

## TABLE OF CONTENTS (Continued)

<u>CHAPTER</u>		<u>PAGE</u>
3.3.3	Illustration of the Conditions . . . . .	55
3.3.4	Discussion . . . . .	57
3.4	Physical Implementation of Task-PE Model . . . . .	57
3.4.1	Repair Strategy and Reliability Analysis . . . . .	62
3.4.2	Reliability Enhancement . . . . .	63
3.4.3	Experimental Results . . . . .	66
3.4.3.1	Reliability Comparisons . . . . .	66
3.4.3.2	Reliability Enhancement Type 2: by Enlarged Coverage . . . . .	69
3.4.3.3	Hardware and Interconnect Overhead . . . . .	71
3.4.3.4	Average Lengths of Replacement Chains . . . . .	71
3.4.4	Discussion . . . . .	73
3.5	Future Work: Construction of $k$ Fault Tolerant Systems . . . . .	73
3.5.1	Design of Balanced System . . . . .	76
3.5.2	Design of Unbalanced System . . . . .	80
3.5.3	Experimental Results . . . . .	80
3.5.3.1	Reliability-Cost Trade-offs . . . . .	81
3.5.3.2	Replacement Chain Selection . . . . .	83
<b>4</b>	<b>CONCLUSIONS AND FUTURE WORKS . . . . .</b>	<b>86</b>
4.1	Generalization of Group Segregation Property in Kogge-Stone Adder . . . . .	87
4.2	Fault Tolerant Many-Processor Systems . . . . .	88
	<b>APPENDIX . . . . .</b>	<b>91</b>
	<b>CITED LITERATURE . . . . .</b>	<b>96</b>
	<b>VITA . . . . .</b>	<b>102</b>

## LIST OF TABLES

<u>TABLE</u>		<u>PAGE</u>
I	PARAMETERS FOR THE THREE DESIGNS SHOWN IN FIGURE 9, NORMALIZED TO THE 1-SPARSE DESIGN ©IEEE 2017 . . . . .	27
II	REDUNDANT INTERCONNECT COMPLEXITY COMPARISONS AMONG VARIOUS APPROACHES ©IEEE 2017 . . . . .	70
III	COST AND MEAN AVERAGE DISTANCE (MAD) FOR THE COV- ERAGE DISTRIBUTION CORRESPONDING TO THE SYSTEMS IN FIGURE 37 . . . . .	83

## LIST OF FIGURES

<b><u>FIGURE</u></b>		<b><u>PAGE</u></b>
1	The GP computation stages of various 16-bit PPA's ©IEEE 2017 . . . . .	8
2	The defect tolerant KSA design in (Ndai et al., 2007) ©IEEE 2016 . . . . .	9
3	Group Segregation Property of KSA and corresponding approaches for defect tolerant and sparse PPA design ©IEEE 2016 and ©IEEE 2017 . . . . .	10
4	A 16-bit Radix-3 KSA ©IEEE 2016 . . . . .	16
5	An example of the proposed adder design approach with 24-bits and 3 groups ©IEEE 2016 . . . . .	16
6	Reliability Analysis for the proposed design with 64-bit Brent-Kung Adder ©IEEE 2016 . . . . .	21
7	Reliability comparison between the proposed design versus the N-Adder approach with 64-bit Brent-Kung Adder ©IEEE 2016 . . . . .	22
8	Design trade-off points for various 64-bit adders at 0.1% defect rate ©IEEE 2016 . . . . .	23
9	Various designs for 32-bit LFA's with changing sparsity ©IEEE 2017 . . . . .	28
10	A variety of Group-Segregation Block structures ©IEEE 2017 . . . . .	29
11	Various Residual GP Restoration options ©IEEE 2017 . . . . .	31
12	Overall design structure of an $m$ -bit $k$ -Sparse PPA ©IEEE 2017 . . . . .	32
13	Representing an arbitrary PPA in a 2D space using Cartesian Coordinates, and estimation of interconnect lengths via half-perimeter method ©IEEE 2017 . . . . .	34
14	Various metrics evaluated on 64-bit Sparse PPA's with changing sparsity on various PPA types ©IEEE 2017 . . . . .	36
15	The 64-bit Adder Design space in power-delay domain ©IEEE 2017 . . . . .	37

## LIST OF FIGURES (Continued)

<u>FIGURE</u>		<u>PAGE</u>
16	A zoomed-in view of Figure 15 ©IEEE 2017 . . . . .	37
17	Typical approaches for implementing fault tolerance in systems with multiple PE's . . . . .	40
18	a) Task-PE relationships with replacement possibilities augmented for a system with 5 tasks and 7 PE's, b) its Replacement Graph . . . . .	46
19	After a repair, $p_2$ “inherits” the incoming replacements of $p_3$ , while $p_1$ “inherits” the incoming replacements of $p_2$ . . . . .	49
20	Illustration of Lemma 1: reversal of a replacement chain . . . . .	50
21	Illustration for Lemma 2, (a) Case 1: $p \in \mathbb{P}(\mathbb{T})$ , (b) Case 2: $p \in GW(\mathbb{T})$ , (c) Case 3: $p \notin \mathbb{P}(\mathbb{T})$ and $P \notin GW(\mathbb{T})$ . . . . .	52
22	Illustration of proof of main claim: (a) $1 \Rightarrow 2$ , (b) $3 \Rightarrow 1$ . . . . .	53
23	Illustration of the Repair Chain Independence: when a PE chooses an arbitrary chain . . . . .	55
24	Illustration of the Repair Chain Independence: effect on a PE when another PE chooses an arbitrary chain . . . . .	56
25	Effect of choosing of the shared spare by the faulty PE . . . . .	56
26	(a) Replacement of one PE by another: requiring the latter to be connected to former's neighbors; (b) An alternative implementation using switches and Routers . . . . .	58
27	The proposed Router-PE System with a fault tolerance illustrated . . . . .	59
28	The Router-PE system corresponding to the system in Figure 27, isomorphic to the Task-PE model . . . . .	60
29	a) The $4 \times 4$ 2D Array layout with the corresponding coordinates assigned, b) system with spares distributed around in the system with local coverages, c) Replacement relationships among the PE's from the configuration in (b) ©IEEE 2017 . . . . .	62

## LIST OF FIGURES (Continued)

<b>FIGURE</b>		<b>PAGE</b>
30	An example repair in a $4 \times 4$ Array implemented on a $5 \times 5$ PE Array a) after 3 faults are repaired, b) after the 4 <sup>th</sup> fault at $p_{24}$ is repaired through chain $p_{24} \leftarrow p_{19} \leftarrow p_{14} \leftarrow p_{15} \leftarrow p_{10} \leftarrow p_4$ ; and c) using alternate replacement chain of $p_{24} \leftarrow p_{18} \leftarrow p_{13} \leftarrow p_7$ . The changed replacements are also highlighted ©IEEE 2017 . . . . .	63
31	a) The $4 \times 4$ Router space divided into 2 $4 \times 2$ spaces, resulting in increase in total number of spares, b) type 2 reliability enhancement by allowing some performance degradation over (a), c) Reliability comparison between (a) and (b), d) same space divided into 4 $2 \times 2$ spaces, resulting in further increase in total number of spares, and d) type 2 reliability enhancement by allowing more performance degradation over (e), f) Reliability comparison between (d) and (e) ©IEEE 2017 . . . . .	64
32	Reliability comparison between the previous works and the proposed approach . . . . .	67
33	Reliability enhancement in the proposed approach by allowing Normalized Delay to increase ©IEEE 2017 . . . . .	69
34	Average lengths of Replacement Chains for various systems . . . . .	72
35	Systems with two different cost-reliability trade-offs (reliability obtained through simulation) . . . . .	74
36	Balanced and Unbalanced Replacement Graphs . . . . .	76
37	Reliability curves for different systems . . . . .	81
38	System delay increase for shortest vs random chains . . . . .	85

## List of Algorithms

1	Algorithm to design Balanced Replacement Graphs . . . . .	77
2	Algorithm to design Unbalanced Replacement Graphs . . . . .	78

## **LIST OF ABBREVIATIONS**

PPA	Parallel Prefix Adder
KSA	Kogge-Stone Adder
HCA	Han-Carlson Adder
LFA	Ladner-Fisher Adder
BJA	Brent-Kung Adder
RCA	Ripple-Carry Adder
PE	Processing Element
k-FT	k-Fault Tolerant
MAD	Mean Average Distance
UIC	University of Illinois at Chicago
EDAA	European Design and Automation Association
IEEE	Institute of Electrical and Electronics Engineers

## SUMMARY

The dissertation discusses about defect and fault tolerance in nano-scale systems. With fast shrinking device dimensions, the fabrication equipments have been unable to keep up with the required level of accuracy. As a result, the manufacturing defect rates are on the rise. On the other hand, smaller device dimensions have also resulted in devices being more susceptible to cosmic particle strikes as well as electromigration. Because of the reasons mentioned, defect and fault tolerance mechanisms are necessitated for new nano-scale systems.

The dissertation is divided into two parts: highly defect tolerant Parallel Prefix Adder (PPA) design (and beyond); and highly fault tolerant Many-Processor systems supporting on-the-fly repairs.

For the first part, we point out the unique “Group-Segregation” property of Kogge-Stone Adders (KSA), a particular type of PPA. This property enables the KSA to be divided into two disjoint groups, even and odd bits, such that each can perform the computations independently of each other. If one group is defective, the other group can compute the correct results for the defective group with marginal hardware and computation overhead. In this work, we show how such number of groups can be increased such that more defects can be tolerated. Moreover, we propose a technique to extend the Group-Segregation property to other PPA’s as well. The Group-Segregation property also paves the way for designing “sparse” PPA’s, where every  $k$ -th bit computes their results, while the rest of the bits rely on the former. We propose a generalized approach for designing arbitrary  $k$ -sparse PPA’s.

In the second part, we introduce a novel 2-Layered Task-PE model, which facilitates design of highly fault tolerant Many-Processor systems. Such a model contains Processing Elements at one layer

## **SUMMARY (Continued)**

for the execution of the Tasks at another layer. The faults are tolerated on-the-fly, after every fault occurrence through “chain of replacements”, where a faulty PE is replaced by a functional PE located nearby; the latter being replaced again by another PE; the chain of replacements continue until a spare is reached. In this dissertation, we show how such a model culminates in the Many-Processor system to be “replacement chain independent”, i.e., the reliability is unaffected by the choice of the replacement chain for repair. This independence in choosing the repair choice makes the repairs simple and lightweight. We show the design of such system with strictly local interconnects, making the system highly scalable.

## CHAPTER 1

### INTRODUCTION

The dissertation deals with defect and fault tolerance in nano-scale circuits and systems, specifically Parallel Prefix Adders and Many-Processor systems. Scaling down of sizes for current technologies following Moore's Law has resulted in prevalence of the reliability issues in the devices. With the rapid size scaling of devices, the manufacturing procedures have been unable to keep up the accuracy needed for fabricate such devices. As a result, manufacturing **defects** are becoming an important issue. (ITRS, 2015) predicts that the defect rates will increase from current  $10^{-7}$  up to  $10^{-1}$ . Manufacturing defects are not the only issue holding back the future devices. Smaller devices are also more susceptible to external interference, such as cosmic ray particle strikes (Zieglar, 1996) as well as internal issues like electromigration (Lienig, 2013). This does not only result in transient faults, but also has the potential to damage the devices in permanent **faults**.

To formalize, we consider two issues affecting the device reliability:

**Defects** are due to the manufacturing inaccuracies of the fabrication equipments.

**Faults** are due to the external interference of cosmic particle and internal issues, such as electromigration. They typically occur during normal runtime of the circuit/system. The effects can be transient or permanent.

The dissertation discusses about a defect tolerance approach for Parallel Prefix Adders and a fault tolerance approach in Many-Processor systems.

### 1.1 Defect Tolerance and Design Space Exploration via Sparsity in Parallel Prefix Adders

Parallel Prefix Adders (PPA) are a class of adders which provide a large variety of choices along performance-power-area paretal front. Kogge-Stone Adder (KSA) is a particular type of PPA, which is perhaps the fastest typical PPA, while using a large amount of hardware. Owing to the design, it performs significant redundant computation. The redundant computation is KSA can be explained by its *Group Segregation* property. In KSA, the bits are divided into two distinct groups: even bits and odd bits. The two bits compute their results independently of each other. Furthermore, a group can compute an additional copy of its result just by using the results of the other group. This *Group Segregation* property can be utilized in two ways:

1. Having a redundant copy of the result available is convenient when the design is used for defect and fault tolerance.
2. Removing all the computing elements of a group will result in half the adder computing the result, while the other half uses the results of the former.

Due to the desirable property of KSA, most of the works on reliable and sparse adders have concentrated on KSA. While there exist approaches suitable for other PPA's, they are mainly ad-hoc and are unsuitable for implementation in a general sense. Furthermore, with only two groups at their disposal, it can guaranteed tolerate only a single defect: one defect at any one of the groups. However, with increasing defect rates, it is unlikely that there will be only one defect circuit wide. To make sure the minimum number of tolerable defects is greater than 1, the dissertation proposes an approach to increase the number of groups to  $k$ . With  $k$  groups, as long as one of the groups is defect free, the adder functions.

Thus, it can tolerate guaranteed  $(k - 1)$  defects. Moreover, we propose a design methodology to extend the desirable KSA properties onto other PPA's as well, making every PPA able to tolerate defects.

Overall the proposed approach does not only tolerate more defects, but also extends the group segregation property of KSA onto other PPA's. This makes any arbitrary PPA being able to tolerate defects, not only KSA. The defect tolerance is done through post manufacturing reconfiguration.

If we remove all the computing elements of a group and let the other group perform all the computations, following the aforementioned observation 2, the resulting adder is a *2-sparse* version of KSA, which incidentally is a Han-Carlson Adder (HCA). In this dissertation, we extend the Group Segregation property of KSA onto other PPA's to design any arbitrary sparse PPA. We also increase the number of groups in a similar fashion to come up with  $k$ -sparse PPA's. Sparse adders provide the designers with more choices in design space to utilize the exact adder fitting the requirements.

## **1.2 Formalization of Basic Replacement Chain Algebra to Arbitrary Many-Processor System**

In the consumer market, future processors are expected to contain upto a thousand cores (Borkar, 2007). FPGA's, NoC's, GPU's already contain hundreds of identical unit computing elements. DSP processors have classically contained hundreds, if not thousands, of identical adders and multipliers. In spite of the high performance delivered by the processors, FPGA's, NoC's and GPU's, future reliability issues may keep their capabilities limited (Constantinescu, 2003). As an example, the Power Mac G5 machine at Virginia Tech, in the absence of any Error Correcting Codes (ECC) was so prone to crashing, that it used to crash even before booting (Geist, 2016). The very high fault rate was due to the cosmic particle strikes. Even though adding ECC could alleviate the problem to a certain extent, still ECC cannot detect and correct an unlimited number of faults.

Keeping the issue in mind, lots of work has been done on the issue of Many-Processor reliability. There have been works done on 2D, 3D Grid/hypercube, Tree, Ring topologies of Processors. Their main assumptions are: given a specific topology, how to retain the original topology when  $k$  Processors are removed through faults. However, two important things missing are: defect and fault tolerance approach for any arbitrary topology and lightweight on-the-fly repairs after every fault manifestation.

As pointed out, most of the previous works concentrated on specific topologies. While they delivered satisfactory reliability for those specific topologies, they cannot be used in other topologies.

On the other hand, lightweight on-the-fly repair capabilities is desired in any many-processor system. After a fault occurs, stopping the system, repairing it and eventually resuming the normal functionalities will result in inconvenience for the user.

In the proposed design technique, we first consider *Replacement Chain* based repair on a novel 2-layered Task-PE model, and derive the underlying properties through *Replacement Chain Algebra*. We will show that the proposed 2-layered model can be extended to work on any arbitrary topology. In repairs via Replacement Chain, a faulty Processing Element (PE) is replaced by a non-faulty functional one in its neighborhood, which in turn, is replaced by another PE. This chain goes on until a spare PE is reached. Using Replacement Chain Algebra, we formalize this process on 2-layered model. We show that such a repair mechanism is able to support lightweight on-the-fly repairs after every fault manifestation. Moreover, the system can guarantee  $k$  fault tolerances when certain conditions are fulfilled.

Overall, the proposed approach can deliver the following:

1. Reliability with guarantee.
2. Simple lightweight on-the-fly repairs.

3. Localized interconnects in any arbitrary topology, resulting in scalability.

### 1.3 Organization

Overall, the dissertation consists of two parts: Chapter 2 discusses about generalizing the Group Segregation property of KSA; Chapters 3 discusses about designing fault tolerant Many-Processor system following the 2-layered model supported by Replacement Chain Algebra. Effectively, the first part deals with extending an specific property of a circuit to others, while the second part deals with formalizing basic properties of any system.

Chapter 2, is divided into two distinct sections: defect tolerant PPA design and sparse PPA design. First we discuss about how PPA's function, followed by discussions about KSA and its Group Segregation property. Following this part, we discuss about generalized defect tolerant PPA design. Finally, we show how sparse adder designs can be generalized by using the Group Segregation property.

The second part is contained in Chapter 3, starting from discussions about the previous works in the fault tolerance in Many-Processor systems and also the motivation behind our proposed design model, followed by the introduction of the proposed 2-layered Task-PE model for designing fault tolerant Many-Processor systems. The conditions of guaranteed  $k$  fault tolerances in the proposed 2-layered model are discussed next. This chapter also proves that the repairs in the proposed model are indeed lightweight and simple, and do not need to follow any complex algorithms. Next, we provide a case study of designing the popular 2D Processor Array with fault tolerance capabilities using the proposed 2-layered model. We show that such a design is scalable in terms of the interconnects, as they are always constrained within local neighborhoods. As a result, even with more number of components, the interconnects are not long. Furthermore, the reliability delivered by the approach is compares favor-

ably to other similar approaches. Finally, we show generalized approaches for designing fault tolerant Many-Processor systems with arbitrary topology and placement of components as a future work. This approach also aims to reduce the interconnect lengths by making sure they are as local as possible. Specific placements and topologies (grid, hypercube etc) can act as a special case of the proposed design approach.

We conclude the overall dissertation in Chapter 4.

## CHAPTER 2

### GENERALIZATION OF KOGGE-STONE ADDER GROUP SEGREGATION PROPERTY

*This chapter has been partially published as (Banerjee and Rao, 2017a), (Banerjee and Rao, 2016). ©2017, 2016, IEEE.*

#### 2.1 Introduction to Parallel Prefix Adders

Parallel Prefix Adders (PPA's) are a class of adders which compute the sum of two  $n$ -bit numbers by considering two particular parameters: Generate and Propagate (GP). Generate indicates if a carry has been *generated* by a bit, or a *block* of bits. On the other hand, Propagate indicates if a bit or a *block* of bits can *propagate* a carry signal from the previous bits to the next.

PPA's (Parhami, 2010) use three steps to compute the sum bits:

1. **Bit-level Generate and Propagate:** for each input bit pair  $A_i$  and  $B_i$ , the bit-level Generate and Propagate signals  $(G_{i:i}, P_{i:i})$  are computed as:

$$(G_{i:i}, P_{i:i}) = (A_i \cdot B_i, A_i \oplus B_i)$$

2. **Block-level Generate and Propagate:** this stage derives  $(G_{i:0}, P_{i:0})$  for all the bits  $i$ . Such results can be computed via the following prefix formulation of sub-blocks:

$$(G_{i:j}, P_{i:j}) = (G_{i:k_1}, P_{i:k_1}) \circ (G_{k_1:k_2}, P_{k_1:k_2}) \circ (G_{k_2:k_3}, P_{k_2:k_3}) \circ \dots \circ (G_{k_n:j}, P_{k_n:j})$$

when  $i \geq k_1 \geq k_2 \geq k_3 \geq \dots \geq k_n \geq j$ .

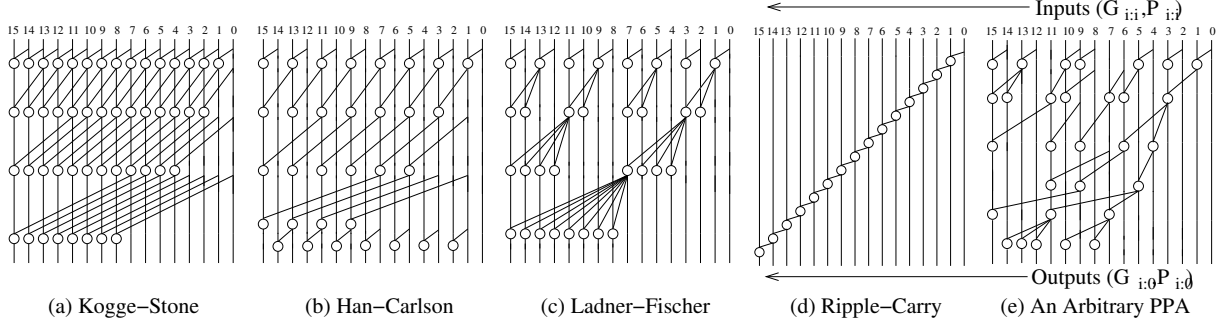


Figure 1: The GP computation stages of various 16-bit PPA's ©IEEE 2017

The “ $\circ$ ” operator is defined as:

$$(G_{i:k}, P_{i:k}) \circ (G_{k:j}, P_{k:j}) = (G_{i:k} + P_{i:k} \cdot G_{k:j}, P_{i:k} \cdot P_{k:j})$$

where  $i \geq k \geq j$ .

**3. Carry and Sum computation:** the  $G_{i:0}$  and  $P_{i:0}$  of all the bits are used to compute Carry  $C_i$  and Sum  $S_i$  in parallel:

$$C_i = G_{i-1:0}, S_i = P_{i:i} \oplus C_i$$

Obviously, the most critical part of a PPA is stage 2 in terms of area, power and performance. In this crucial stage, the Block-level Generate and Propagate  $(G_{i:0}, P_{i:0})$  is computed for every bit  $i$ , and each

of these computations relies on the preceding bits from  $(i - 1)$  to 0. Therefore the entire circuit consists of a large network of Generate and Propagate blocks. Since “ $\circ$ ” operator is associative, the block-level Generate and Propagate computations can be done in arbitrary order. Various PPAs thus use distinct architectures for this stage. Figure 1 shows this stage 2 structures of various PPA’s, including Kogge-Stone (KSA), Han-Carlson (HCA), Ladner-Fischer (LFA), Ripple-Carry (RCA) and an Arbitrary PPA, with each of the circles representing one “ $\circ$ ” operation, or a *GP Block*.

## 2.2 Kogge-Stone Adders and Group Segregation Property

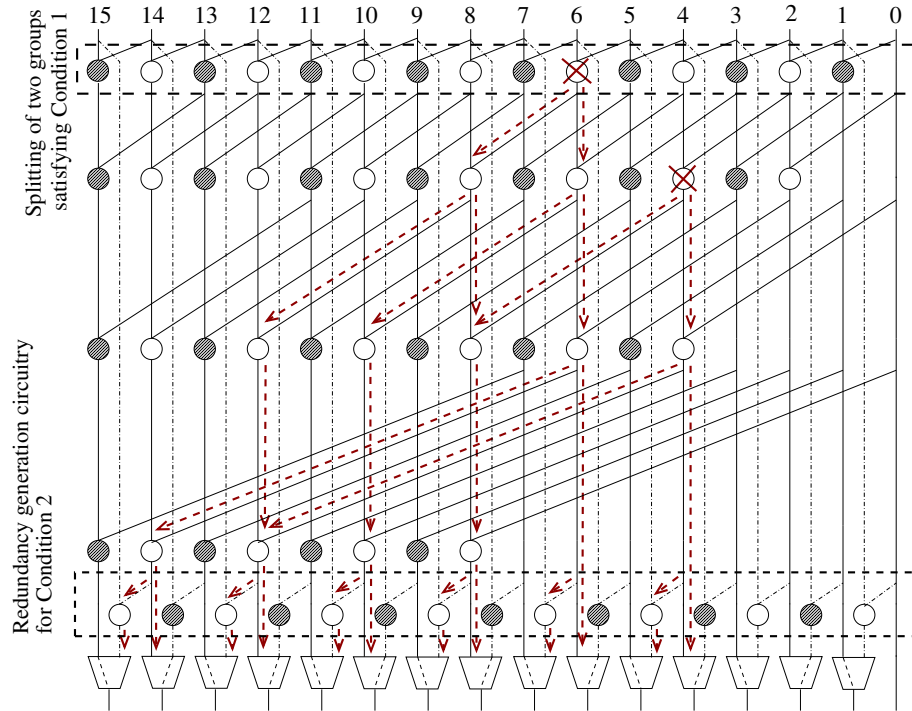


Figure 2: The defect tolerant KSA design in (Ndai et al., 2007) ©IEEE 2016

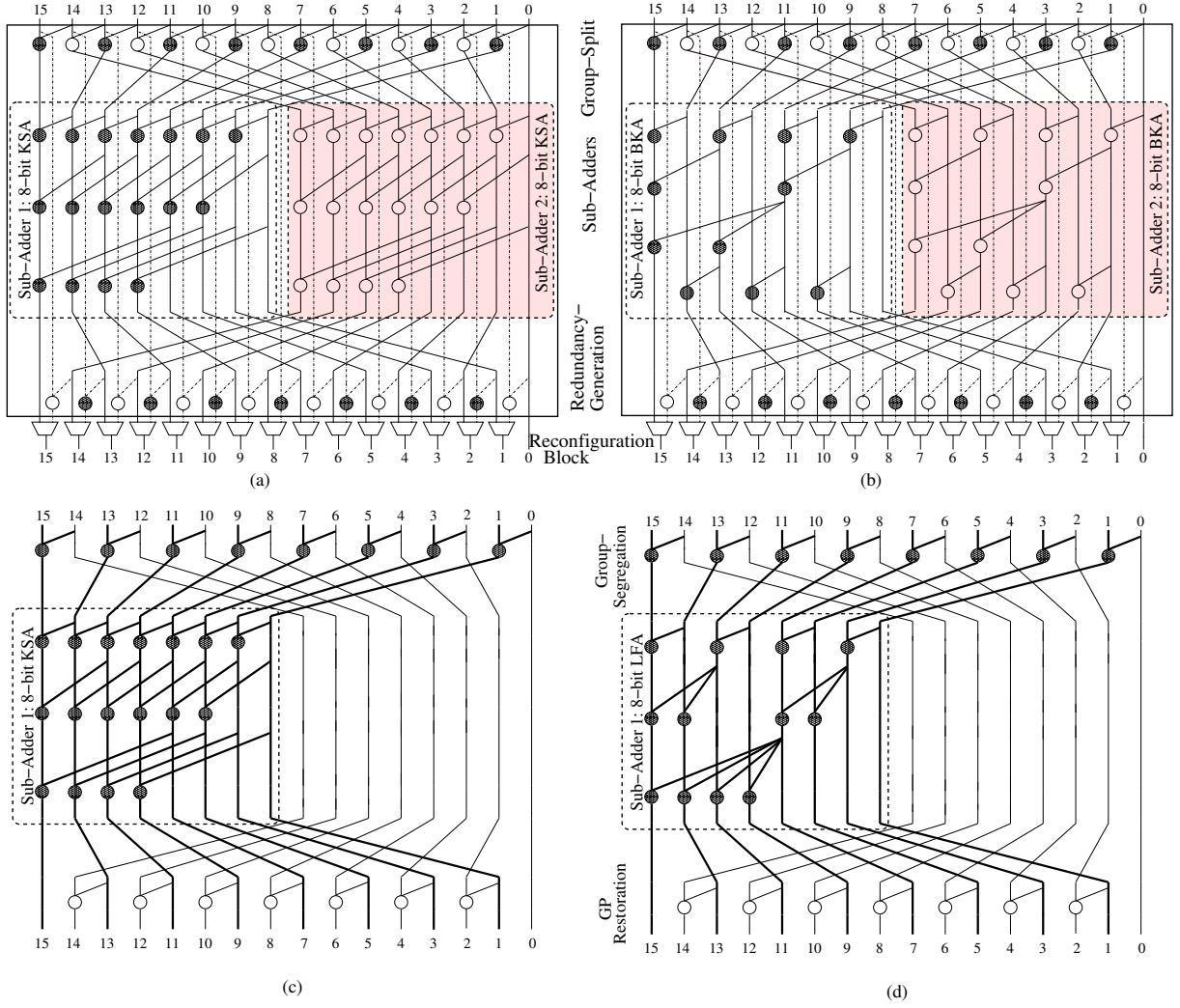


Figure 3: Group Segregation Property of KSA and corresponding approaches for defect tolerant and sparse PPA design ©IEEE 2016 and ©IEEE 2017

A typical (radix-2) 16-bit Kogge-Stone Adder (KSA) has its bits divided into two disjoint hardware groups: even (Group 0) and odd (Group 1) bits. As a result, any defect in one group will have their

erroneous results constrained within the same group, thus not contaminating the results of the other group.

Moreover, it turns out that each group can derive a redundant copy of the results of the other group, i.e., the corresponding GP signals with a small extra cost. Such a scheme is proposed in (Ndai et al., 2007), with its structure shown in Figure 2. This design ensures the correct result when all the defects are within one group. In such a case, the defect-free group is used to generate the results for both, with the additional row of extra GP Blocks at the bottom. For example, bit 14 has two copies of  $(G_{14:0}, P_{14:0})$  generated: one from the regular computation of the even group (which bit 14 belongs to), another by performing  $(G_{14:14}, P_{14:14}) \circ (G_{13:0}, P_{13:0})$ , obtained from bit 13, part of the odd group. When the defect is in the even group, as is shown in the figure, the first copy will be erroneous. However, the second copy, computed from the odd group, remains correct. This copy of the result is then selected by configuring the MUXes at the output end.

Essentially, the KSA structure satisfies two key conditions, which make it possible to achieve defect tolerance at such a low cost: **Condition 1:** the formation of *hardware disjoint groups*, each carries out half of the GP signal calculations independently; and **Condition 2:** the *replacing capacity* of one group's result for deriving a redundant copy of the other groups' results, thanks to the arrangement of interleaving bits into groups.

Extending the defect tolerance framework to other PPA's such as Brent-Kung (BKA), Ripple-Carry (RCA), shown in Figure 1, is not straightforward, as they do not easily satisfy the two conditions presented in KSA. For this reason, we need to take a careful look into the structure that makes such conditions hold. From Figure 2, it can be observed that Condition 1 for KSA is essentially enabled by the first

level of GP Blocks (denoted as *Group-Split*) only. Then, all the communications are strictly contained within each group. Therefore, this first level of GP Blocks of a KSA can in fact be extracted out, to satisfy Condition 1 for other general PPA structures. Condition 2, it turns out, is also not limited to the specific structure of KSA's. As long as the groups are formed by interleaving bits, and an additional level of GP Blocks (denoted as *Redundancy-Generation*) are added at the end, each group can guarantee to generate a redundant copy of results for all the other groups. The structure between those two levels do not really impose any constraints for either Condition 1 or Condition 2, thus are customizable.

Based on these two observations, Figure 3(a) illustrates a radix-2 KSA in a reorganized form, highlighting the three stages, where the two *Sub-Adder* structures in the middle take the form of two 8-bit KSAs. It is worth noting that, the Sub-Adders are only responsible for the computation of GP information for the corresponding bits of the same group. However, each is identical to the form of an 8-bit KSA, except that the interleaved bits (even and odd bits) are taken in each as inputs, instead of the consecutive ones as in a regular 8-bit KSA.

Thus, the two groups of a defect tolerant KSA, (as the two Sub-Adders) being independent from one another, can be replaced in block by another adder structure without affecting the functionality or the defect tolerance ability. As is shown in Figure 3(b), the corresponding Sub-Adders are replaced by Brent-Kung Adders (BKA's). Such a replacement can be done in general with any PPA structure, thus opening up new choices in defect tolerant designs.

On the other hand, due to the presence of two disjoint groups, and the convenience of computing the results of one group using the results of other, the hardwares for a group can simply be eliminated that make the group dependent on the other group of results. This creates the basic idea for *sparse adders*.

Figure 3(c) then shows that the 2-Sparse version of a KSA is achieved in three steps: 1) segregating even and odd bits by the first layer of GP operators; 2) eliminating the computations of one group (the even bits) as Non-Essential Bits; 3) derive the final results of the Non-Essential Bits by adding a layer of GP-blocks at the output end, from the results of Essential Bits.

From the above observations it is then possible to generalize a 2-Sparse KSA with other types of Sub-Adders, as is shown in the example of Figure 3(d). Note that as long as the first layer of GP-blocks (denoted as Group-Segregation) is present, two independent and mutually substitutable groups (even and odd bits) are formed. From then on, one group (the odd bits) can be designated as the Essential Bits, and it can use any form of PPA structure as a Sub-Adder. The associated GP operators for the other group will be eliminated as these even bits now constitute the Non-Essential Bits. Eventually their results will be derived from the Essential Bits via the last layer of GP-blocks at the output side (denoted as Residual GP Restoration).

## **2.3 Defect Tolerant Parallel Prefix Adders**

### **2.3.1 Motivations and Previous Works**

The scaling down of device dimensions into the nanometer range is likely to result in significantly higher defect rates during the manufacturing process of IC's (ITRS, 2015). With significantly increased defect rates, defect tolerance mechanisms are necessitated to guarantee a reasonable yield. Post manufacturing reconfiguration techniques to bypass defects are already applied in memory systems and FPGA's (Shi and Fuchs, 1992) (Hatori et al., 1993). However, such low-cost defect tolerance techniques rely heavily on the relative independence of operations of the homogeneous components, such as LUT and memory cells. Logic systems, on the other hand, usually constitute heterogeneous components

with strong dependencies among each other. This makes it hard to realize fine-grained, low-cost defect tolerance schemes for a high level of defect rate.

In the first part of this chapter, we focus on the design of highly defect tolerant PPA's. Multiple papers have addressed the reliability issues in general adders. For instance, defect tolerant designs are proposed in (Kahng and Kang, 2012), (Ye et al., 2013) where an adder can be configured for its approximate computation, according to the amount of accuracy required in an application. (Townsend et al., 2003) applies quadruple time redundancy for an adder to tolerate error occurrences. The work in (Johnson, 1989) applies Triple-Modular Redundancy with a voter to guarantee the correct result. (Ghosh et al., 2007) uses a mechanism to stretch the clock period to accommodate the critical delay paths, with minor performance sacrifice. Even though these designs have a relatively low hardware cost, they mostly assume a low level of defect occurrences, and cannot scale when defect rates are high.

Among the various adders, PPA provides a general form to represent a wide range of adder design choices (Parhami, 2010). Reliable PPA designs have mostly been done on the particular form of Kogge Stone Adders (KSA), due to its inherent regular structure and hardware redundancy. For performance purposes, the hardware of a typical KSA is divided into two disjoint groups of the even bits and the odd bits. This provides a natural way to make the defects *isolated*: errors caused by the defects in one group will not affect the results produced by the other group. Furthermore, the in-built redundancy in a KSA allows each group to be capable of generating the results for the other group, with a small hardware and time overhead. Based on these features, defect tolerance mechanisms are proposed in (Ndai et al., 2007) (Ghosh et al., 2008) for KSA's, where as long as one of the two groups is defect-free, all the computations are done using the defect-free half of the adder to ensure correctness.

This section of work presents a general approach for defect tolerant PPA that is no longer limited to the special case for KSA. The proposed approach is a scalable solution along two dimensions: the number of disjoint hardware groups (beyond two), and the variety of PPA structures that can be adopted (beyond KSA). This provides enhanced reliability, as well as opens up a large number of design choices for defect tolerant PPA designs.

### **2.3.2 The Proposed Defect Tolerant Design**

Under a high defect rate, chances are exceedingly low that all the defects will be concentrating within only one group out of two. Therefore, the number of groups needs to be increased to enhance the likelihood of having at least one defect free functional group. Such an expansion on the number of groups can be achieved by exploring higher radices (Gurkayna et al., 2000). The radix of a PPA is defined by how many maximum “o” operations are done inside a single GP Block. A radix-3 KSA example is shown in Figure 4, with 3 disjoint groups: Group 2 (bit Mod 3 = 2), Group 1 (bit Mod 3 = 1) and Group 0 (bit Mod 3 = 0). Defect tolerance can be achieved in this case, by letting each group generate two additional results for the other two groups. With every bit having 3 copies of the result, at least one defect free group will ensure the success of the defect tolerance mechanism. Overall, increasing the number of disjoint groups raises the level of defect tolerance.

Figure 5 shows a generalized example of the proposed design with 3 groups in a 24-bit PPA. The Group-Split stage resembles the first level of a radix-3 KSA, which guarantees that the computations of the 3 groups will be carried out by disjoint hardware. In the subsequent levels, the three groups (8-bit Sub-Adders) independently carry out the computations of GP signals. Their specific structure can be of any PPA design. In the end, at the Redundancy-Generation stage, two additional GP Blocks are

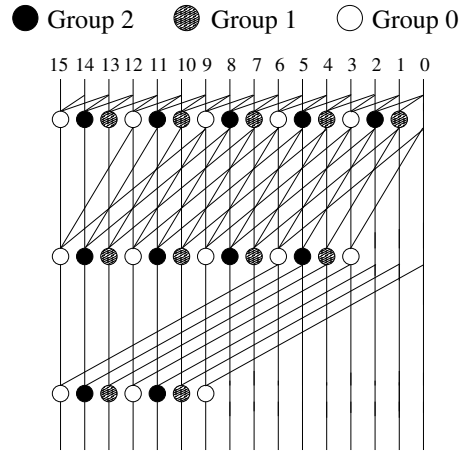


Figure 4: A 16-bit Radix-3 KSA ©IEEE 2016

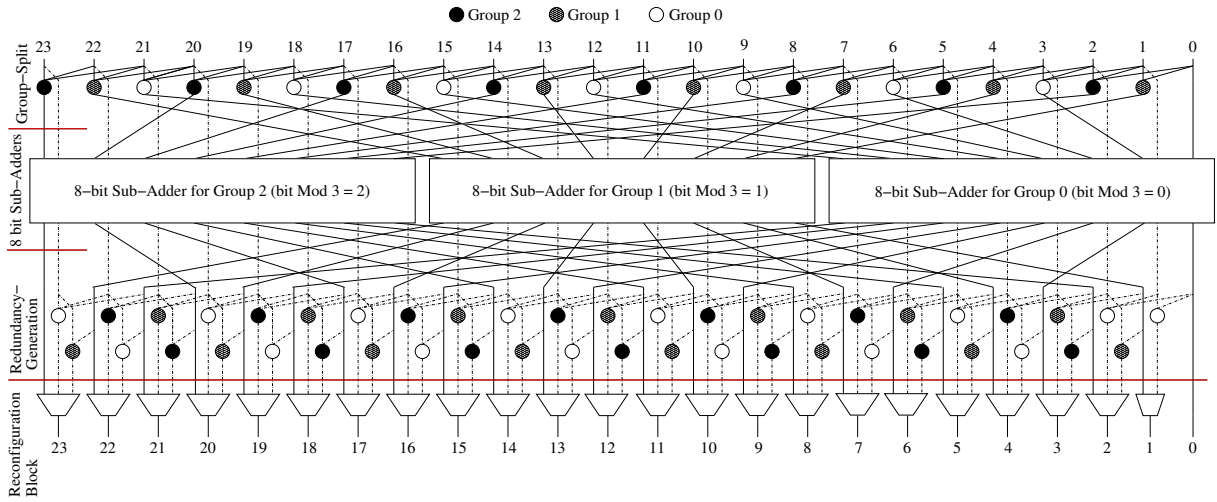


Figure 5: An example of the proposed adder design approach with 24-bits and 3 groups ©IEEE 2016

added to each bit for the redundant results for the other two groups. As an example, bit 12 (belonging to Group 0) generates a copy of its result from its own group's hardware. In addition, one redundant copy is derived by:  $(G_{12:12}, P_{12:12}) \circ (G_{11:0}, P_{11:0})$  from the result of Group 2, and another redundant copy  $(G_{12:12}, P_{12:12}) \circ (G_{11:11}, P_{11:11}) \circ (G_{10:0}, P_{10:0})$  is derived from the result of Group 1. As long as one group is defect-free, the correct result for every bit can be guaranteed by configuring the MUXes at the output end.

In the overall design, the extra hardware needed in the Group-Split stage and the final Redundancy-Generation stage is determined by the number of groups in the design. The type of Sub-Adders is determined independent of the group number. The entire defect tolerant PPA structure can thus be characterized by two parameters: the number of groups, and the type of the Sub-Adders.

We can formalize the design approach with  $k$ -groups by dividing the framework into the following multiple stages:

**Group-Split:** This stage takes the same form as the first level of a radix- $k$  KSA, where bit  $i$  interacts with bits  $(i-1)$ ,  $(i-2)$ ,  $(i-3)$ , till  $(i-k+1)$  to generate  $G_{i:i-k+1}$  and  $P_{i:i-k+1}$ . Thus each bit ( $i \geq (k-1)$ ) requires a  $k$ -input GP Block at this stage. This stage forms the  $k$  groups, which will be independent of each other with disjoint hardware.

**Sub-Adders:** Once the groups are formed, all the bits are computed via a corresponding Sub-Adder: bit  $i$  will be grouped with bits  $(i \pm k)$ ,  $(i \pm 2k)$  and so on, to be processed by the  $(i \bmod k)^{th}$  Sub-Adder. For  $k$  groups, there will be  $k$  Sub-Adders. At this stage, the Sub-Adders can be of any PPA design. For bit  $i$  in a group (Sub-Adder), the following operations are performed to compute  $G_{i:0}$  and  $P_{i:0}$ :

$$(G_{i:0}, P_{i:0}) = (G_{i:i-k+1}, P_{i:i-k+1}) \circ (G_{i-k:i-2k+1}, P_{i-k:i-2k+1}) \circ (G_{i-2k:i-3k+1}, P_{i-k:i-3k+1}) \circ \dots$$

$$\circ (G_{i-mk:0}, P_{i-mk:0})$$

In this equation,  $(i - mk)$  is any bit where  $(i - mk) < k$  and  $(i - mk) \geq 0$ . The bits  $i, (i - k), (i - 2k), \dots (i - mk)$  belong to the same group, such that  $(i - mk) < k$  and  $(i - mk) \geq 0$ , are part of the same group. The corresponding Sub-Adder is responsible for the computation of  $G_{i:0}$  and  $P_{i:0}$ .

**Redundancy-Generation:** At this stage, each bit generates  $(k - 1)$  additional results by utilizing the extra GP Blocks. Specifically, bit  $i$  performs the following operations to generate redundant copies of its results  $i_1, i_2, i_3$  to  $i_{k-1}$ :

$$i_1 = (G_{i:i}, P_{i:i}) \circ (G_{i-1:0}, P_{i-1:0})$$

$$i_2 = (G_{i:i}, P_{i:i}) \circ (G_{i-1:i-1}, P_{i-1:i-1}) \circ (G_{i-2:0}, P_{i-2:0})$$

...

$$i_{k-1} = (G_{i:i}, P_{i:i}) \circ (G_{i-1:i-1}, P_{i-1:i-1}) \circ \dots \circ (G_{i-k+2:i-k+2}, P_{i-k+2:i-k+2}) \circ (G_{i-k+1:0}, P_{i-k+1:0})$$

### 2.3.3 Experimental Results

We carried out the simulation for the proposed defect tolerant PPA design with various instances using a C program. All the analyses are done on 64-bit adders, considering defects at the transistor level. This model assumes that any defective transistor will result in erroneous behavior is on the pessimistic side, and covers most of the interconnect defects as well. We analyze the reliability of various PPA implementation at the Sub-Adder stage, and the associated hardware requirements, with

KSA, Han-Carlson (HCA), Ladner-Fischer (LFA), Brent-Kung (BKA) and Ripple-Carry (RCA) as the Sub-Adders.

The proposed PPA design is compared with a standard N-Adder defect tolerance approach: one out of  $N$  adders is selected by a MUX to be used for defect tolerance. Thus, as long as one of the adders is defect free, the defects can be tolerated. In this way, an N-Adder approach is comparable to an  $N$ -group configuration of the proposed approach, except that the redundancy is organized “externally” in an N-Adder approach, rather than “internally” as in the proposed approach. The N-Adder approach is similar in hardware cost to the N-Modular Redundancy (NMR) approach. However, an NMR approach requires the majority of the adders to function properly for the correct result to be voted out, to deal with dynamically occurring faults. An N-Adder approach, on the other hand, requires only a single adder to be defect free to bypass defects that have been diagnosed.

In both the proposed approach and the N-Adder design, a block of MUXes is needed to select the correct output. The correctness of such a reconfiguration block is crucial for the entire defect tolerance scheme. Thus we assume that in both the cases, it is guaranteed reliable without any defects, possibly by adopting highly reliable devices.

To maintain the generality in the proposed approach for devices with any dimension, we evaluate the quality of the proposed design approach with the following generic parameters:

**Reliability** is calculated by randomly generating a set of defective devices, and determining if the defect can be tolerated in the adder. This is done for 1000 times, and the reliability is determined by *the ratio of the successful cases to the total number of cases (1000) considered*.

**Performance** (and also **Delay**) is indicated by the *sum of fan-ins of the GP Blocks along the critical path*. From (Gurkayna et al., 2000), it is known that an  $n$ -input GP Block usually has less than  $n/m$  times more delay than an  $m$  input GP Block, when  $n > m$ . In this paper, we evaluate the performance of GP Blocks via the lower bound case of their number of inputs. Thus, we consider 2-input GP Block 1.5 times faster than 3-input GP Blocks and 2 times faster than 4-input GP Blocks, etc. In this way, the sum of fan-ins of the GP Blocks along the critical path indicates an upper limit of the delay, or the lower limit of performance. As an example, a 64-bit 1-group RCA has a fan-in sum of the GP-Blocks along the critical path equal to  $126 (= 63 \times 2)$ , whereas, a 64-bit 4-group RCA has the same fan-in sum of 34 (4 for Group-Split stage and 30 for Sub-Adder stage)+ additional degradation due to Redundancy-Generation stage. The possible values for additional degradation can be either 0 for no defect condition, or 2, 3, or 4 depending upon the number of defective groups. Thus, the fan-in sum along the critical path can be either of these values: 34, 36, 37, or 38.

Required **hardware** is measured by the total number of transistors required for CMOS implementation. Basically, an  $i$ -input GP Block will require  $(6 \times i) + 2$  number of transistors (including both PMOS and NMOS transistors).

In this section, by an adder with certain number of groups, we mean the type of adder used in the Sub-Adder section of the proposed structure. As an example, by a 64-bit RCA with 4 groups, we mean that the structure has an input of 64-bits, with 4 groups, each of type RCA of 16-bits.

### 2.3.3.1 Reliability Analysis

Figure 37 shows the reliability analysis for various 64-bit BKA's. Each reliability curve is plotted with defect rates at the transistor level, shown for 2 to 6 groups. Reliability (represented in percentage)

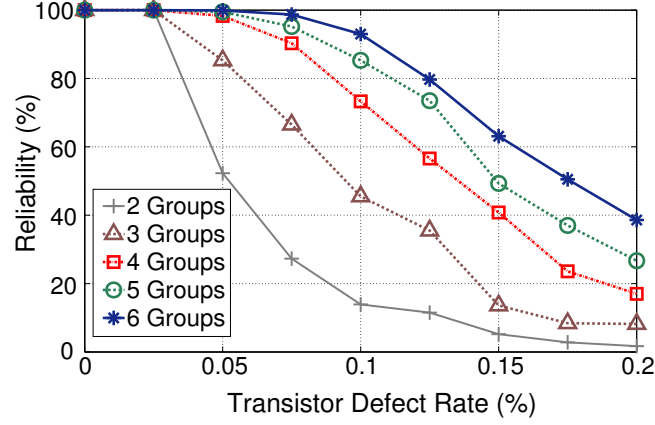


Figure 6: Reliability Analysis for the proposed design with 64-bit Brent-Kung Adder ©IEEE 2016

is given by the ratio of the number of instances where the defects are tolerated, to the total number of trials.

Figure 37 verifies the effect of increasing the number of groups, which consistently delivers enhanced reliability. As an example, for BKA with transistor level defect rates of 0.1%, 2 groups can only tolerate the defects for less than 20% of the time. Reliability is increased to more than 70%, when the number of groups is increased to 4. It can be further enhanced to more than 90% if the number of groups is 6. This trend of enhanced reliability due to increased number of groups is generally observed to be true for all types of Sub-Adders.

Figure 7 depicts the reliability comparison between N-Adder and the proposed approach for 64-bit BKA's. The proposed scheme is superior to the N-Adder approach in both cases of  $N = 3$  and 5. This is because for an N-Adder approach, a single defect is enough to deem an adder unusable. In the proposed approach, since the Redundancy-Generation stage is significantly large, any defect there will leave more

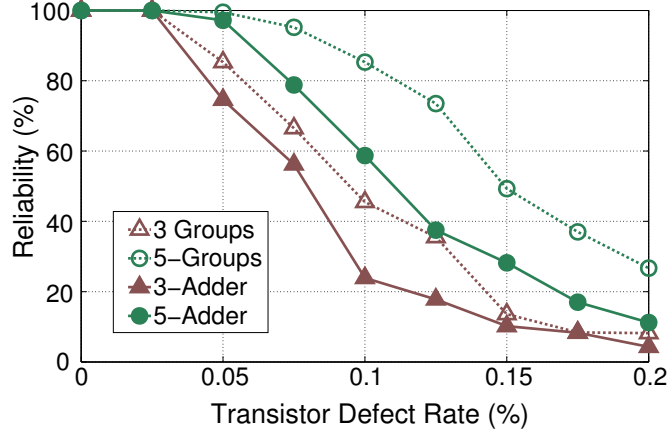


Figure 7: Reliability comparison between the proposed design versus the N-Adder approach with 64-bit Brent-Kung Adder ©IEEE 2016

room for the actual adder to perform without any error affecting it. We will show that the reliability gap between N-Adder (and thus NMR) and the proposed approach for higher performance adders to be even greater.

### 2.3.3.2 Expansion of Reliable PPA Design Space Choices

In Figure 8, various designs from the proposed approach is shown together with the N-Adder based approach to illustrate their positions in the design trade-off space. Comparison is done by considering three metrics: reliability, transistor number, and delay (indicated by the sum of fan-ins of the GP Blocks along the critical path).

Figure 8(a), (b) and (c) show the design of PPA's that can meet the reliability requirements of  $> 50\%$ ,  $> 70\%$ , and  $> 90\%$ , respectively, under a defect rate of  $0.1\%$ .

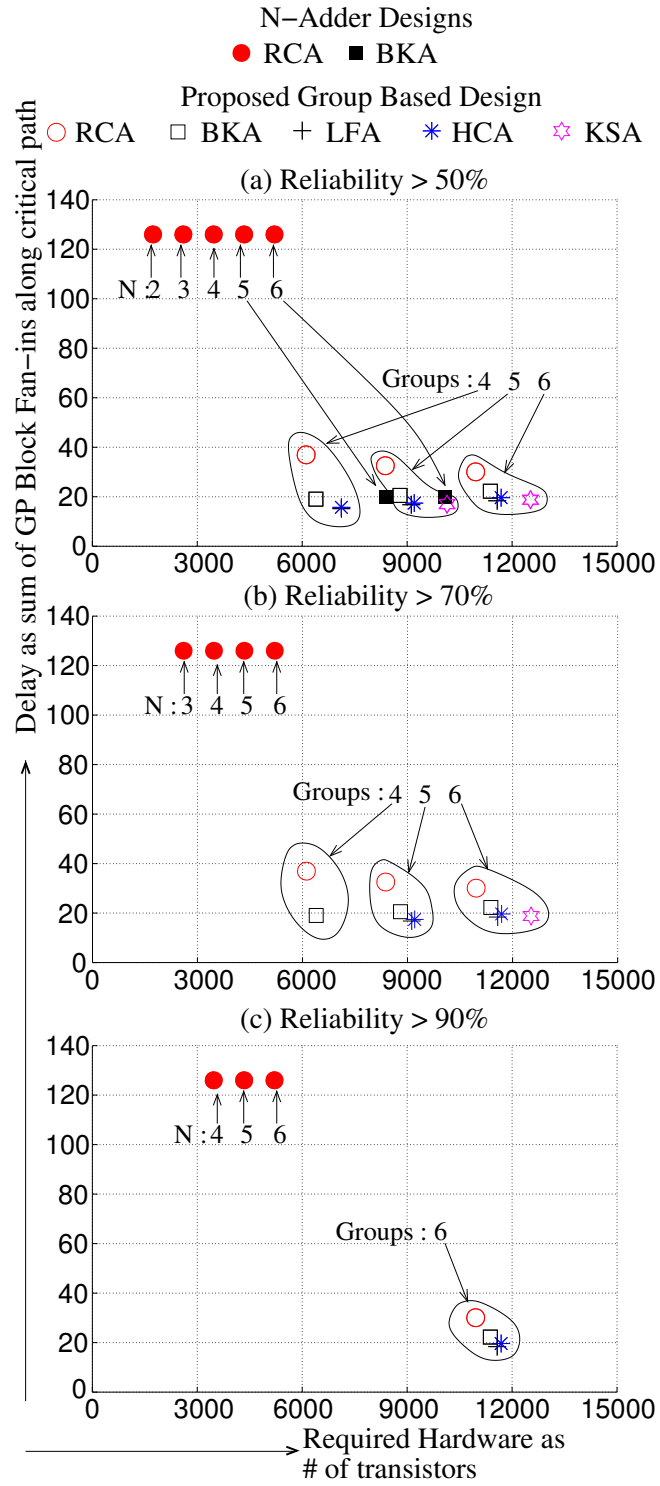


Figure 8: Design trade-off points for various 64-bit adders at 0.1% defect rate ©IEEE 2016

Overall, for the N-Adder approach, only simple types of adders such as RCA and BKA can meet the reliability requirements, while higher performance adder designs (such as KSA, HCA etc) fail to deliver even  $> 50\%$  reliability. In fact, no other N-Adder approaches other than RCA can deliver a reliability of  $> 70\%$ . The proposed scheme, however, can deliver a number of solutions with RCA, BKA, LFA, HCA and KSA that meet various reliability requirements. Overall, the low delay, high performance region is dominated by the proposed design approach, while the N-Adder approach occupies the complementary region where the delay is high and the hardware requirement is low. Overall, the proposed design approach expands the design space by delivering highly reliable PPA's of a variety of types for any given defect rate.

### **2.3.3.3 Delay Variation in the Proposed Scheme**

The proposed approach results in some performance variations over a non defect-tolerant single group PPA:

- 1) An  $n$ -bit  $k$ -group design with the same type of adder structure at the (parallelly operating) Sub-Adder stage will have each of the Sub-Adder to be  $(n/k)$  bits wide, and effectively reducing the delay due to parallelism, compared to the same adder structure of an original  $n$ -bit implementation.
- 2) With the increasing number of groups, the number of inputs to each GP Block in the Group-Split stage increases, thus adding delay to the critical path. In addition, the presence of higher input GP Blocks at the Redundancy-Generation stage also introduces extra delay, particularly for tolerating a higher number of defects.

For lower-performance adders (such as RCA), factor 1 is dominant due to reduction in delay in linear scale, while the performance degradation is negligible in comparison. It is evident from Figure

8, a 4-Adder approach has a delay of  $> 120$ , while the comparable 4-group approach has a delay of  $< 40$ , effectively delivering 3-fold performance enhancement. For higher performance adders, factor 2 is dominant, since the reduction in the total delay due to factor 1 occurs in a logarithmic scale. However, as is shown in Figure 8, the performance degradation for BKA, LFA, HCA and KSA are negligible with the increasing number of groups, in the proposed approach.

## 2.4 Sparse Parallel Prefix Adders

### 2.4.1 Motivations and Previous Works

With the advent of low-power high-performance design regime, it is important to choose the right component which delivers the most fine-grained optimal design trade-offs. Adders, being one of the most common and important components of any computing platform, are one of the most suitable candidates for such design space explorations. In this section, we provide a unified framework to achieve more design choices by considering *Sparsity* of adders.

A  $k$ -Sparse adder computes the Block-level GP results (stage 2 of a PPA) only from every  $k^{th}$  bit (denoted as **Essential Bits** thereafter), and use their results to derive results (of stage 2) for the remaining bits (denoted as **Non-Essential Bits**). Sparse adders are useful to explore supplementing designs for the existing PPA's, which differ slightly from the original adder in terms of power-performance-area metrics. Thus, exploration sparsity is important if small fine-tuning is needed in the design.

This section focuses on general Sparse PPA's as an extension of Group Segregation property of KSA discussed in the beginning of the current chapter. A framework is proposed to derive various sparse designs for any existing PPA structure. The resultant approach opens up a large set of available

PPA designs, with various combinations of structures and sparsities, such that a designer can choose among them along the Pareto-front.

Sparse PPA's have been proposed in (Aktan et al., 2015) (Matthew et al., 2003) to explore additional performance-power trade-offs. Sparse PPA's are constructed by allowing only a subset of bits (denoted as "Essential Bits" in this paper) of the entire adder to compute their own results, with the rest of the bits (denoted as "Non-Essential Bits") deriving their results from the Essential Bits. Such a "sparse" design enables more varieties of PPA structures and thus facilitating design space trade-off explorations.

In the field of general PPA design, (Liu et al., 2007) (Roy et al., 2014) propose ILP based approaches to obtain a PPA structure satisfying a given set of constraints such as area, power, and performance. These approaches are able to search within the huge design space of possible structures, to obtain a PPA that meets the specifications. However, the runtime complexity is high, and these approaches do not provide a ready set of PPA's for the designers to choose from.

Among the previous works: (Zlatanovici et al., 2009) proposes Sparse PPA's by considering the radix and lateral fanout at each stage of the adder; (Aktan et al., 2015) focuses on the the optimal sparseness of Kogge-Stone and Ladner-Fisher Adders to minimize power; (Matthew et al., 2003) proposed an alternative sparse Ladner-Fischer Adder structure; (Zeydel et al., 2010) evaluated the various parameters (structure, wire length, sparseness etc) to understand the power-performance trade-offs. Other examples of Sparse PPA's include Conditional Sum Adder (Sklansky, 1960) and Carry Select Adder (Bedrij, 1962). Overall, most of the previous work are either limited to a narrow scope or ad-hoc in nature. There lacks a systematic framework of constructing Sparse PPA's in general.

Sparsity	Number of GP Blocks	Interconnect Length	Power	Delay
1	1	1	1	1
2	0.7875	1.0404	0.7527	0.65
4	0.8125	1.0699	0.7793	0.5

TABLE I: PARAMETERS FOR THE THREE DESIGNS SHOWN IN FIGURE 9, NORMALIZED TO THE 1-SPARSE DESIGN ©IEEE 2017

#### 2.4.2 Design Approach for Sparse Parallel Prefix Adders

For a 2-Sparse design, the Group-Segregation and Residual GP Restoration are both straightforward in implementation, each consisting of one layer of GP blocks. However, extending from 2 to a  $k$ -Sparse design will significantly increase the overhead in these two building blocks. In fact, this is a necessary price to pay, when more than half of the bits are made Non-Essential. Moreover, a variety of design options can be explored to realize the functionalities of Group-Segregation and Residual GP Restoration.

##### 2.4.2.1 Examples

Figure 9 shows three examples of 32-bit LFA's with 1, 2, and 4 Sparse structures. One can easily observe the reduction of fan-outs with increasing sparsity of the same PPA structure. Table 1 shows the corresponding parameters, normalized with respect to the default 1-Sparse design. The improvements in the number of GP Blocks required, power and delay can be observed with increased sparsity in LFA.

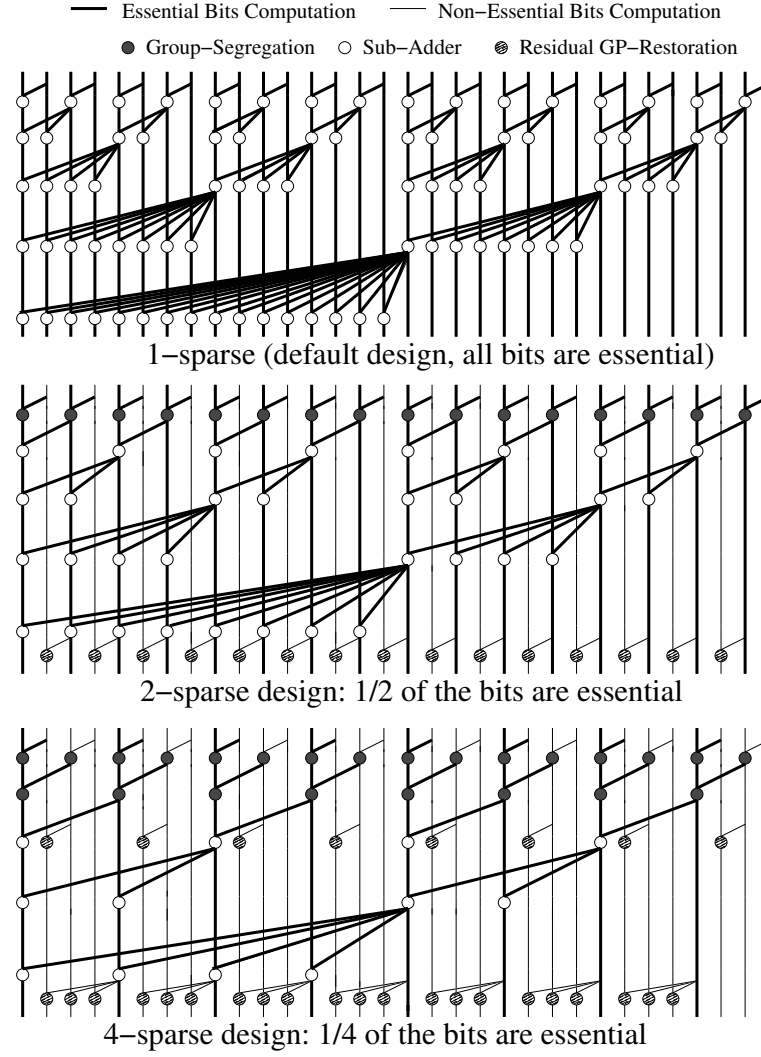


Figure 9: Various designs for 32-bit LFA's with changing sparsity ©IEEE 2017

On the other hand, longer interconnects are needed for LFA's with higher sparsities. The components of the design are discussed in the next subsections.

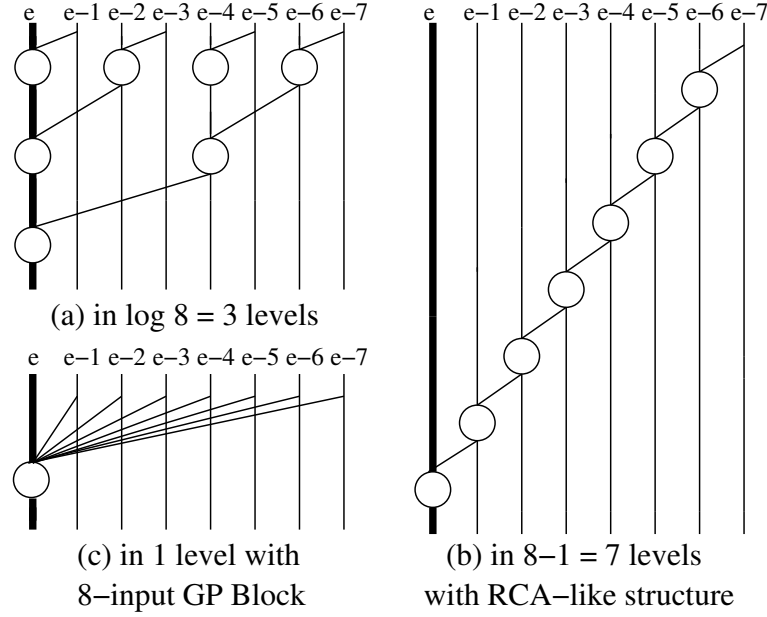


Figure 10: A variety of Group-Segregation Block structures ©IEEE 2017

#### 2.4.2.2 Group-Segregation designs for $k$ -Sparse PPA's

The Group-Segregation stage functions to “bootstrap” the Essential Bits, such that when they are sent through the Sub-Adder, no interactions are needed with the Non-Essential Bits. For 2-Sparse design, this means every Essential Bit (every  $2^{nd}$  bit)  $e$  needs to compute  $(G_{e:e-1}, P_{e:e-1})$  at the Group-Segregation stage. For a  $k$ -Sparse adder, every  $k^{th}$  bit  $e$  now needs to compute  $(G_{e:e-k+1}, P_{e:e-k+1})$ , which is identical to a  $k$ -bit Block-level GP computation.

To form a  $k$ -bit block GP computation, design choices range from 1-level (using a  $k$ -input GP computation),  $\lceil \log_2 k \rceil$  - level (using 2-input GP computation), to  $k-1$  level (using Ripple-Carry-Adder fashioned sequential computation). Such choices are shown in Figure 10(a) to (c) for an example of

8-Sparse design. Note that the 1-level design shown in Figure 10(c) essentially employs a high *radix* (Gurkayna et al., 2000) GP Block with 8 inputs, which involves a large overhead internally within each GP operator. In the experimental result section, we choose the form in Figure 10(a), where the Group-Segregation always requires  $\lceil \log_2 k \rceil$  levels.

#### 2.4.2.3 Residual GP Restoration designs for $k$ -Sparse PPA's

For each of the Non-Essential Bit  $n$ , to achieve the final block GP signal of  $(G_{n:0}, P_{n:0})$  without a Sub-Adder, two tasks are required, both related to the nearest Essential Bit  $e$  to the right of  $n$  (assuming  $n > e$ ). First, the Non-Essential Bit  $n$  needs to compute its *local* block GP signal of  $(G_{n:e+1}, P_{n:e+1})$ , and this can be done without the results from the Essential Bits. Then, once the results of the essential bit  $e$ ,  $(G_{e:0}, P_{e:0})$  from the Sub-Adder is available, bit  $n$  will integrate the result of  $(G_{e:0}, P_{e:0})$  with its local  $(G_{n:e+1}, P_{n:e+1})$  to derive its own *global* block GP result of  $(G_{n:0}, P_{n:0})$ .

Figure 11 shows various structure options for *Residual GP Restoration*, corresponding to the Group-Segregation stage given in Figure 10. The design choice of the Residual GP Restoration stage should mirror that of the Group Segregation, so as to maximize the reusability of intermediate results and minimize hardware overhead.

Overall, similar to Group-Segregation, the Residual GP Restoration has various structural options with different complexities and performance. In the experimental result of this paper, we select the structure of Figure 11(a), which matches that of Figure 10(a).

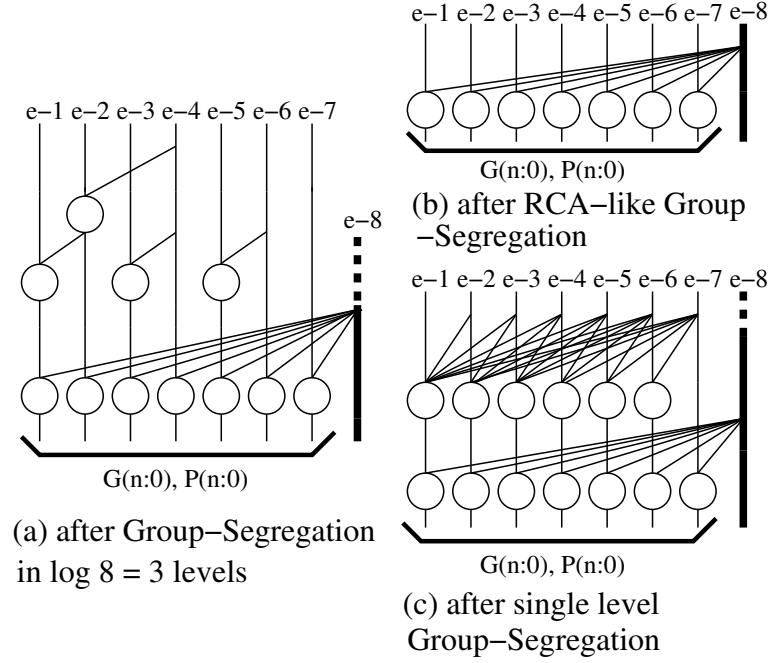


Figure 11: Various Residual GP Restoration options ©IEEE 2017

#### 2.4.2.4 Overall Design

An overall design framework of an  $m$ -bit  $k$ -Sparse PPA is shown in Figure 12. The structure starts with the **Group-Segregation** stage, which provides the initial block GP signals for all the  $\lfloor m/k \rfloor$  Essential Bits (shown in bold lines), enabling them to carry out their independent computations later on.

Next, all the Essential Bits are channeled to the  $\lfloor m/k \rfloor$  **Sub-Adder**. This building block will compute (with any choice of PPA structure) the final block-level GP signals for each Essential Bit  $e$ :  $(G_{e:0}; P_{e:0})$ .

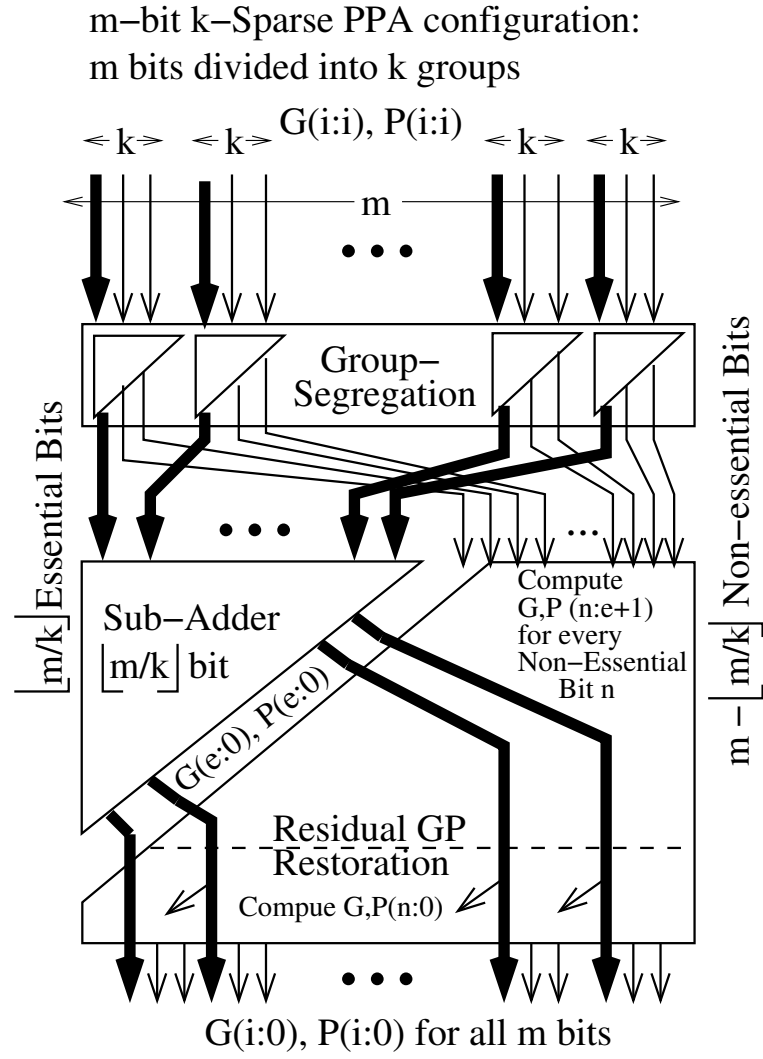


Figure 12: Overall design structure of an  $m$ -bit  $k$ -Sparse PPA ©IEEE 2017

In parallel with the Sub-Adder, the  $(m - \lfloor m/k \rfloor)$  Non-Essential Bits (shown in thin lines) are sent to the **Residual GP Restoration** building block. Here, the Non-Essential Bits first produce the local Block GP signals, in parallel with the computation being carried out at the Sub-Adder. Then, the results from the Sub-Adder are taken in to derive the final block GP  $(G_{n:0}, P_{n:0})$  for every Non-Essential Bit  $n$ .

With increased sparsity, more computations are needed for the Non-Essential Bits, while the necessitated computations for the Essential Bits gradually decrease. As a result, when sparsity increases, the Residual GP Restoration block expands, while the Sub-Adder (with fewer number of input bits) block shrinks.

### 2.4.3 Associated Metrics

We follow the metrics provided by (Liu et al., 2007) for estimation of area, interconnect, power and performance. The estimations given here can be scaled according to any given feature size.

**Area** can be represented by two factors: 1) area required by the adder given in the coordinate system shown in Figure 13; 2) number of GP Blocks in the design.

To estimate the **interconnect** requirements, we model the length of an interconnect to be equal to half of the perimeter of the smallest rectangle encompassing the source and the destinations of the interconnect, as shown in Figure 13.

**Power** ( $P$ ) has two components: static and dynamic power. **Static** ( $P_s$ ) power can be assumed to be fixed for a given GP Block. Thus, the static power consumed by the adder is proportional to the number

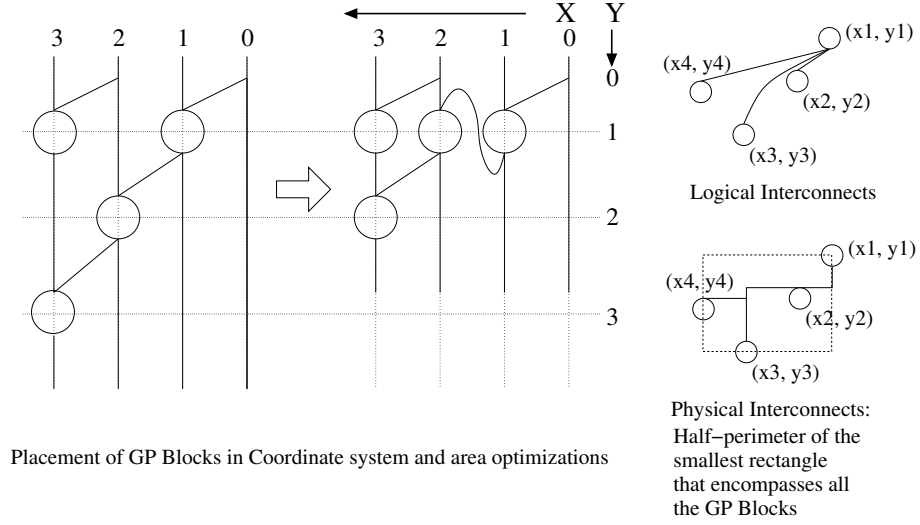


Figure 13: Representing an arbitrary PPA in a 2D space using Cartesian Coordinates, and estimation of interconnect lengths via half-perimeter method ©IEEE 2017

of GP Blocks ( $|GP|$ ). On the other hand, **dynamic** ( $P_d$ ) power is proportional to the fan-outs ( $fo$ ) of the GP Blocks. Overall, the power can be estimated as:

$$P = P_s + P_d = a \times \sum fo + b \times |GP|$$

Here  $a$  and  $b$  are scaling factors, which we assume to be 1 and 0.5 respectively.

To estimate the performance in terms of delay ( $D$ ), we use the concept of logical effort. (Liu et al., 2007) simplifies it as:

$$D_l = 1.5 \times C + 2.5, D_r = 2 \times C + 2.5$$

Delay is estimated from the critical path in the adder. If a GP Block is located along the critical path and the critical path enters it from right fan-in, then we consider the delay to be  $D_r$ . Otherwise when the fan-in comes from the left, the delay at the GP Block is considered as  $D_l$ . Computation of the overall delay is performed by considering all the GP Blocks along the critical path, and the fan-in paths of the critical path. In the given equations,  $C$  indicates the load capacitance for the corresponding GP Block, which is proportional to its fan-out. We apply a scaling factor of 0.5 to estimate  $C = 0.5 \times fo$ .

#### 2.4.4 Experimental Results

A number of PPA structures including KSA, HCA, LFA, BKA (Brent-Kung Adder) and RCA are explored to evaluate their sparse variations. We evaluated 64-bit adders with sparsity ranging from 1 to 8. The results shown in Figure 14 indicates the results (area, interconnect, GP block number, power, delay, and Power-Delay Product) for each type of PPA when sparsity changes. The simulation was carried out using the metrics described in Section IV and implemented in a C program.

For high performance adders, such as KSA, a general trend is that delay is the only negative impact of increased sparsity. All other metrics show improvement with a more sparse design. For HCA, a design of sparsity 3 provides an optimal case in power, while sparsity of 2 provides an optimal point for the metric of Power-Delay Product (PDP) of all the choices.

For LFA's, which have a high fan-out for GP Blocks, increase in sparsity does not affect significantly area or interconnect. However, delay decreases consistently with increasing sparseness due to the fan-outs decreasing. For power, we can observe an optimum point at sparsity of 3. PDP also has a downward trend, but the rate of reduction goes down as the sparsity increases.

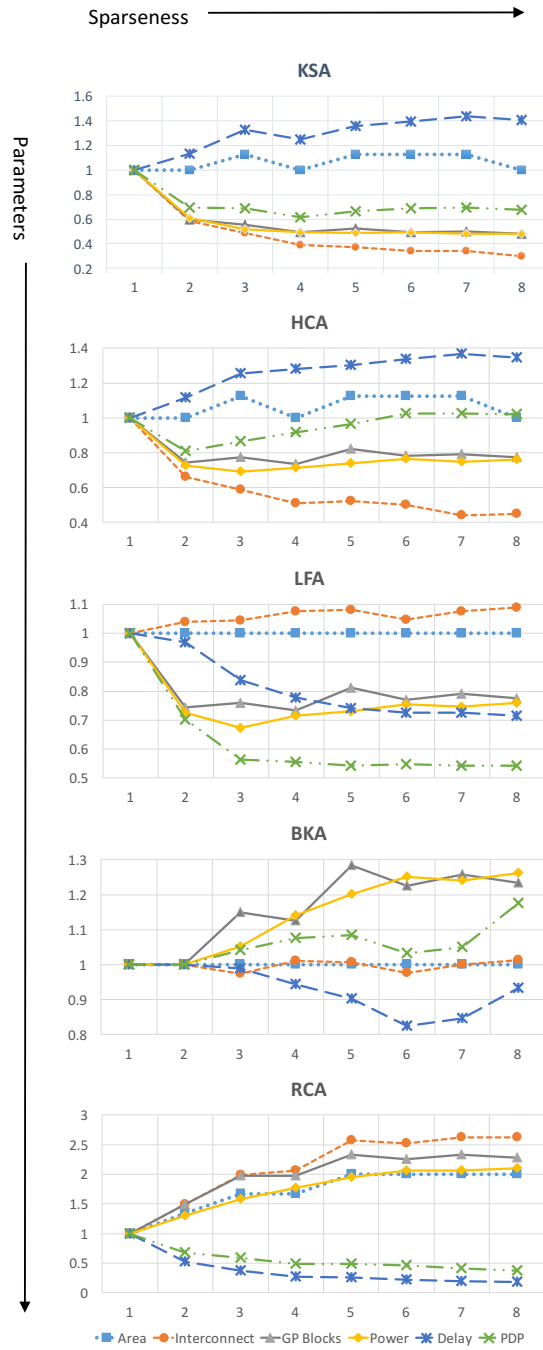


Figure 14: Various metrics evaluated on 64-bit Sparse PPA's with changing sparsity on various PPA types ©IEEE

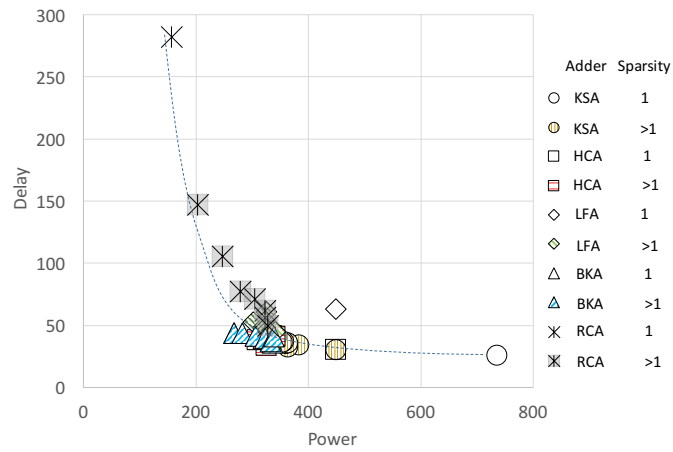


Figure 15: The 64-bit Adder Design space in power-delay domain ©IEEE 2017

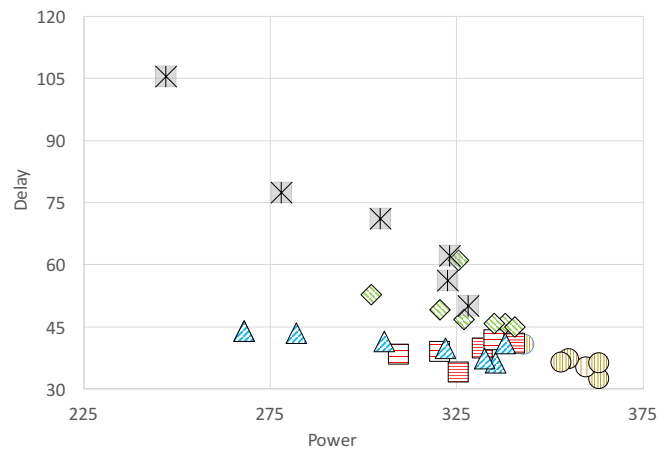


Figure 16: A zoomed-in view of Figure 15 ©IEEE 2017

For BKA's, the power and PDP is lower at the typical 1-Sparse over other higher sparsity designs. Nevertheless, we can observe the presence of the fastest possible BKA at sparsity 6.

Finally, for RCA, while area, power, number of GP Blocks and interconnect length increases, there is a big performance improvement with increased sparsity. For instance, 2-Sparse RCA is almost twice as fast as a default 1-Sparse RCA. Furthermore, PDP also has a decreasing trend with increasing sparsity, indicating the presence of an exploitable power-delay trade-off.

Figure 15 shows the design space for the plethora of sparse PPA's, including KSA, HCA, LFA, BKA and RCA in power-delay domain. While KSA and RCA 1-Sparse and 2-Sparse designs are having a large variation between them in terms of power-delay, in most of the cases, the changes can be observed to be rather small. Such small changes have been shown in the zoomed-in plot of Figure 16. Thus sparsity exploration helps in “fine-tuning” the adder parameters as is shown in this region.

## CHAPTER 3

### FORMALIZATION OF REPLACEMENT CHAIN ALGEBRA IN ARBITRARY MANY-PROCESSOR SYSTEMS

*This chapter has been partially published as (Banerjee and Rao, 2015), (Banerjee and Rao, 2017b). ©2015, 2017, IEEE.*

#### 3.1 Previous Works and Motivations

This chapter introduces a  $k$ -Fault Tolerant ( $k$ -FT) system model supporting decentralized on-the-fly repairs. The modeled system consists of  $n$  functioning Processing Elements (PE's) and at least  $k$  spare PE's. When a fault manifests, the repair is done indirectly by the spares through a “chain of replacements”: the faulty PE is replaced by a functional PE in the neighborhood, which in turn is taken over by another functional PE in its neighborhood; these local reconfigurations continue until a spare is reached.

Classically, the fault tolerance problem in Many-PE system has been termed “Structural Fault Tolerance”, since the aim is to maintain the original PE topology of the system. The most obvious way of achieving such a goal is via “global task mapping” onto a larger Many-PE network. This problem is essentially identical to a Sub-graph isomorphism problem: a classical  $NP$ -Complete problem. Thus its application is mostly concentrated in specific graph topologies, or small systems.

A computationally simpler solution is to replace the faulty PE directly using a spare, such as the one given in Figure 17(b). Although simple to implement, it suffers from a scalability issue: for larger sys-

tems, the number of spare required to maintain the fault tolerance grows significantly. This is because, to make sure every PE is repairable, spares have to be placed on every locality.

Thus, an alternative to this issue is to allow functional PE's, in addition to the spare PE's, to replace any other in the neighborhood, such as the system given in Figure 17(b). Using this approach, the spares, although small in number, can indirectly reach a faulty PE far away inside the system, using the “chain of replacements” approach. Since the reconfigurations are constrained within a neighborhood, it is the most scalable approach compared to others.

In this chapter, we propose a fault-tolerant system model following chain of replacements repair approach to maintain scalability. An additional challenge we address is: how to repair the system following every fault manifestation in a decentralized fashion on-the-fly without significantly affecting the normal operation of the system. This problem is challenging, since without knowing the future fault locations, the system has to do the best job possible to find a repair option to maximize future fault

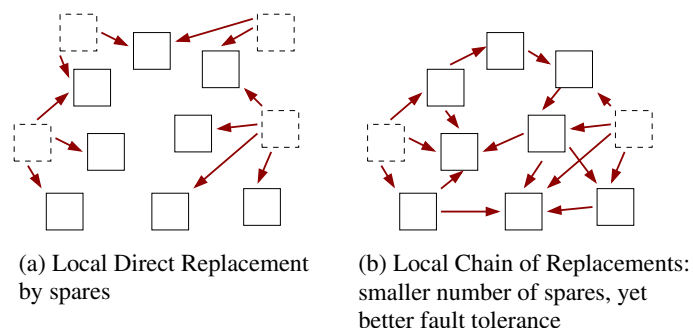


Figure 17: Typical approaches for implementing fault tolerance in systems with multiple PE's

tolerance possibilities. This is an easier problem compared to the scenario where the system needs to repair  $k$  known faults.

To achieve the given goal, we include the following in this chapter:

1. A 2-layered Task-PE Model which facilitates lightweight repairs on-the-fly after every fault manifestation for any arbitrary PE topology.
2. Necessary and sufficient conditions for guaranteeing  $k$ -FT in such systems.
3. Physical implementation of the model on popular 2D Grid structure.
4. Construction of such system following the Task-PE Model.

A general case of multi-PE fault tolerance can be defined as a problem of mapping a given heterogeneous task graph onto a heterogeneous PE graph (with spare nodes). Based on such a model, repair is equivalent to a “global remapping” of the task graph to achieve  $k$ -FT. Such a model is efficient in hardware requirements, and is general to explore the maximum number of repair solutions. However, such a global mapping is a well known *NP*-Complete problem. The presence of such inherent complexity determines that it is not scalable to large systems ( $n$ -PE’s) or a large number of faults ( $k$ ). Therefore, mainly heuristic algorithms, or approaches for specific PE topologies (grid, tree etc.) have been proposed in this domain (Singh and Youssef, 1995) (Bokhari, 1981) (Taura and Chien, 2000) (Chen et al., 1995) (Zhang et al., 2009) (Zhang, 2002). On the other hand, works such as (G. Jiang, 2015) applies this approach on a larger 3D mesh to extract the largest sized fault free 3D sub-mesh.

The alternative end of modeling  $k$ -FT for scalable systems is to retract the scope of repair solutions: repair is achieved via local reconfigurations, as is shown in Figure 17(b) and (c). The approach of local reconfigurations is computationally easier than the global reconfiguration approach. For the “Direct

Replacement” approach given in Figure 17(a), the faulty PE’s are directly replaced by spare PE’s. Usage of such local and global spares are viable for smaller systems, or in systems where the fault tolerance requirements are low. (Khaleghi and Rao, 2013) proposes such an approach where the spares have local reach. (Ren et al., 2013a) uses similar methodology for tolerating NoC router faults.

In Figure 17(b), it can be observed that using Direct replacement by spares in a large system does not scale well: they require more number of spares, at each neighborhood. Using Chain of local replacements, as shown in Figure 17(c) addresses this issue. Among the previous works on many-PE fault tolerance using chain of local reconfigurations, (Jigang et al., 2007) (Fukushi and Horiguchi, 2004) (Izadi and Ozguner, 2003) (Zhang, 2002) (Bruck et al., 1992) (Roychowdhury et al., 1990) (G. Jiang, 2015) (Dutt and Mahapatra, 1997) concentrated on grid, mesh and array topologies; (Chen and Upadhyaya, 1993) (Dutt and Hayes, 1990) focuses on tree topologies; (Ajtai et al., 1992) (Dutt and Hayes, 1992); (Hancheek and Dutt, 1998) (Hatori et al., 1993) applies similar approach to FPGA’s.

NoC’s have been a significant target of implementation of fault tolerant architectures. In NoC’s, there can be two types of faults: faults in PE’s or in Routers. Works in (Zhang et al., 2009) (Ghiribaldi et al., 2011) (Modarresi et al., 2010) (Murali and Micheli, 2004) have addressed the fault issues in PE’s. On the other hand, handling faults in Routers have two distinct approaches: replacing faulty Routers using spare Routers (Khaleghi and Rao, 2013) (Ren et al., 2013a) (Ren et al., 2013b); using virtual topologies to implement functions rather than actual physical topologies (thus ignoring the faulty Routers) (Wachter et al., 2013) (Zhu et al., 2007).

The problem of Structural Fault Tolerance is an important consideration in Systolic Wavefront Arrays as well (Abraham et al., 1987) (Sha and Steiglitz, 1993).

To summarize, most of the previous approaches works with specific PE topology (grid, tree, mesh etc). To the best of our knowledge, no paper, other than (Ajtai et al., 1992) (Dutt and Hayes, 1992), has proposed a general approach for designing fault tolerant systems for any arbitrary topology. However, (Dutt and Hayes, 1992) forces the physical placement of the PE's to be in a grid structure, irrespective of the underlying topology,. In a general case, where PE's are arbitrarily placed on a 2D space, the design does not scale well.

Moreover, the approaches proposed inherently assumes that all the  $k$  faults in the system are known before starting with repairs. Although many of the approaches can technically be applied to on-the-fly fault tolerance after every fault manifestation, the corresponding algorithms to repair 1 fault are equally complex as the algorithms for repairing  $k$  faults. Thus, to apply such complex centralized algorithms, the operation of the system has to be disrupted, which defeats the purpose of the on-the-fly repairs.

Overall, our work contrasts with the existing body of work in the following manners:

1. The proposed approach is suitable for any arbitrary topology and placements of the PE's, as provided in the specifications.
2. The repairs are simple, decentralized and lightweight: operation of the system is minimally affected.

In this chapter, we consider a *replacement chain* based mechanism for fault tolerance: a faulty PE is replaced by a chain of functional PE's, each taking over the task of another, until ultimately replaced by a spare. This model makes it possible to extend the replacement capability of local spares' towards far-away PE's globally. The number of available repair choices via replacement chains is therefore vastly greater than that is offered by a direct replacement scheme, yet the model is still supporting

decentralized, dynamic repair. Comparing to the two alternatives (graph mapping and direct spare replacement), repair through a chain of replacements strikes a trade-off point to offer both efficient hardware cost and computation complexity, and is particularly suitable for modeling large systems with high fault tolerance needs.

### **3.2 Introduction to the Proposed 2-Layered Task-PE Model**

Although replacement chain based repair has been discussed in the literature, the general conditions for guaranteed  $k$ -FT have not been proposed where the repairs take place on-the-fly after every fault manifestation, particularly for scalable systems with a high number of PE's, large  $k$ , and general (arbitrarily defined) topology among the PE's. The main challenges lie in the fact that not all existing replacement chains are compatible with each other: a particular repair via one chain will have impacts on other chains, creating a high level of complexity and dynamics. As the systems scale up, the variety and dynamics of fault sequence and replacement chain possibilities explode.

To track the changes of the system's fault tolerance capability after a repair is carried out via a replacement chain, it is important to clearly define a basis, upon which the dynamics of replacement relationships and impacts of any repair can be well defined and derived. We propose a Task-PE relationship model as the underlying baseline for the replacement relationship, which makes it possible to derive the algebra for the "repair dynamics". The key insight lies in differentiating the PE's and the tasks that they can perform: a PE is characterized to carry out one task, but is able to cover a set of tasks; whether PE  $p$  can replace another PE  $q$  depends on  $p$ 's capability to cover the task that  $q$  is currently carrying out. This model unties the replacement relationship between two PE's, and instead pins down the replacement relationship on a different level of association (PE-task coverage). This makes it

possible to track all the “true” replacement relationship changes upon repair, and ultimately supporting the development of tight conditions for a system to be guaranteed on-the-fly  $k$ -FT.

In the proposed model, each of the  $n$  tasks is executed by one PE, yet can potentially be executed by some other PE's. The capability of a PE  $p$  to cover a task  $t$  might depend on  $p$ 's functionality, interconnection, availability etc. Overall, at any time, the difference between the current *Assignment* (which task is carried out by which PE) and the potential *Coverage* (each task could be carried out by which PE's) underpins the replacement relationship among all the PE's. In other words, which PE's can replace which other PE's will change when repairs are made, shifting tasks among PE's. However the replacement relationship is always determined by the Task-PE graph, which remains static all the time.

### 3.2.1 Formal Definition of the Task-PE Model

The *Task-PE model* consists of two sets of vertices:  $n$  task nodes, and  $(n + k)$  PE nodes, with an example shown in Figure 18(a). The solid edges and the dotted edges indicate current assignment and potential coverages, respectively. A PE (for instance  $p_3$ ) can “replace” some other PE's ( $p_4$  and  $p_6$ ) when it could cover the tasks ( $t_2$  and  $t_4$ , currently being assigned to  $p_4$  and  $p_6$ ), respectively. Such replacement relationships between the PE's, fully specified by the Task-PE relationships, are shown as directed edges on the side in Figure 18(a), as well as stand-alone in Figure 18(b), denoted as a Replacement Graph. It can be seen from Figure 18(b) that two possible replacement chains:  $(p_6 \leftarrow p_3 \leftarrow p_1)$  and  $(p_6 \leftarrow p_7 \leftarrow p_5 \leftarrow p_2)$  are possible repair options, when  $p_6$  is faulty. Formally, the model consists of:

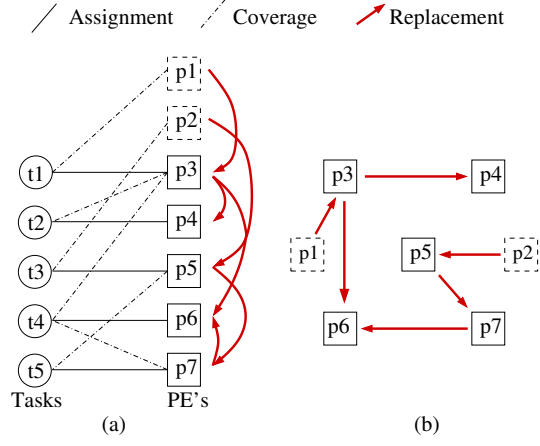


Figure 18: a) Task-PE relationships with replacement possibilities augmented for a system with 5 tasks and 7 PE's, b) its Replacement Graph

- $n$  tasks:  $T_n = \{t_1, t_2, t_3, \dots, t_n\}$
- $(n+k)$  PE's:  $P_{n+k} = \{p_1, p_2, p_3, \dots, p_n, p_{n+1}, \dots, p_{n+k}\}$
- Coverage Matrix  $C$ , where  $\forall x \in [1, n]$  and  $y \in [1, n+k]$ :
  1.  $C(x, y) = 1$  if task  $t_x$  can be covered by PE  $p_y$
  2.  $C(x, y) = 0$  otherwise
- Assignment Matrix  $A$ , where:
  1.  $A(x, y) = 1$  if task  $t_x$  is currently assigned to PE  $p_y$
  2.  $A(x, y) = 0$  otherwise

A valid assignment guarantees that: 1) every task is assigned to exactly one PE; 2) at most one task is assigned to every PE; and 3) an assignment is allowed only when a PE can cover a task. Thus,  $\forall x \in [1, n]$  and  $\forall y \in [1, n + k]$ :

$$1. \sum_{y=1}^{n+k} A(x, y) = 1$$

$$2. \sum_{x=1}^n A(x, y) \leq 1$$

$$3. A(x, y) \leq C(x, y)$$

• Replacement Matrix  $R$  (derived from  $C$  and  $A$ ):

1.  $R(i, j) = 1$ , where  $i, j \in [1, n + k]$  indicating PE  $p_i$  can replace PE  $p_j$ :  $\exists$  task  $t_x$ , where  $x \in [1, n]$ , such that  $A(x, j) = 1$  and  $C(x, i) = 1$ .
2.  $R(i, j) = 0$ , otherwise.

If a task  $t_x$  that is currently assigned to  $p_j$  can also be covered by  $p_i$ ,  $p_i$  is defined as to be able to *replace*  $p_j$ , represented as a directed edge from  $p_j \leftarrow p_i$  in the Replacement Graph, as is shown in Figure 18(b).

• A Replacement Chain (RC) of length  $l$  is defined as a sequence of PE's  $\langle p_{y0}, p_{y1}, p_{y2}, \dots, p_{yl} \rangle$ , such that  $R(y_i, y(i + 1)) = 1$  for  $i \in [0, l - 1]$ . This indicates a chain of edges in the Replacement Graph:  $p_{y0} \leftarrow p_{y1} \leftarrow p_{y2} \leftarrow \dots \leftarrow p_{yl}$ . A repair can be successfully achieved via such a chain, for a faulty  $p_{y0}$ , when  $p_{yl}$  is a spare ( $A(x, yl) = 0 \forall x \in [1, n]$ ).

### 3.3 Necessary and Sufficient Conditions for Guaranteed $k$ -FT

In Figure 18(b), Replacement Graph shows the various repair opportunities for any possible faults. As an example, should  $p_6$  be faulty, the possible chains to repair it are: 1)  $p_6 \leftarrow p_3 \leftarrow p_1$ ; 2)  $p_6 \leftarrow p_7 \leftarrow p_5 \leftarrow p_2$ . The question now is: which chain to choose for repair? Of course, we will choose the chain which will allow maximum future fault tolerance. To find such a chain, we first need to evaluate the dynamics of the repairs.

This chapter presents the rules, namely **Replacement Chain Algebra**, that define the impact of a successful repair via a replacement chain, followed by a main conclusion that a replacement chain will not disappear, but instead reverse itself after being used for a repair. This conclusion will then serve as a basis to show that it does not matter which chain is chosen for repair, the future fault tolerance depends upon the configuration of the system and the fault locations.

#### 3.3.1 Repalcement Graph Changes after a Successful Repair: Replacement Chain Algebra

Upon a successful repair via a replacement chain, the faulty PE will be taken out of the system, thus losing all the outgoing edges in the Replacement Graph.

Figure 19 shows an example repair for  $p_3$  via chain  $p_3 \leftarrow p_2 \leftarrow p_1$ . Assuming a Coverage Matrix  $C$  and an initial Assignment Matrix  $A_0$ , as is shown in Figure 19(a), and the modified assignment  $A_1$  after a repair, as shown in Figure 19(b):

- $C(1,1) = 1$  ( $t_1$  can be covered by Spare  $p_1$ )
- $C(2,2) = 1$  ( $t_2$  can be covered by PE  $p_2$ )
- $A_0(1,2) = 1$  ( $t_1$  is assigned to  $p_2$  before the repair)



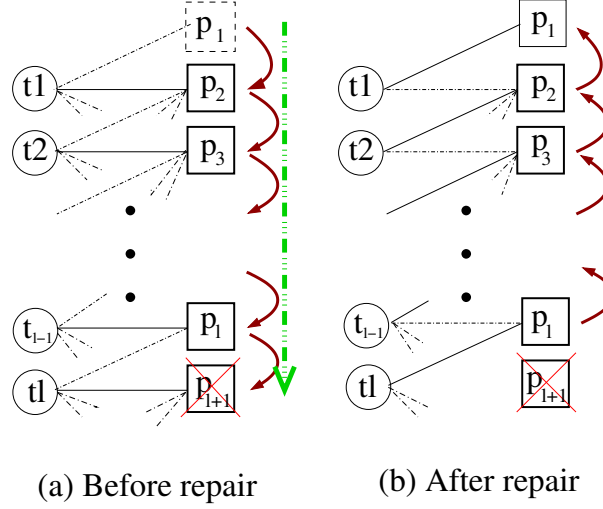


Figure 20: Illustration of Lemma 1: reversal of a replacement chain

a PE  $p_i$  replaces another PE  $p_j$ ,  $p_i$  will “inherit” all the replacements incoming to  $p_j$ . Owing to the unchanged Coverage Matrix, a PE moving on to execute a new task, will “drag” its coverages with it (thus covering the same tasks as before), but may now replace different PE’s than before, because of the reconfigured Task-PE Graph.

Based on the aforementioned dynamics, an important property of replacement chain execution is “reversal”.

**Lemma 1:** A Replacement Chain  $p_1 \rightarrow p_2 \rightarrow p_3 \rightarrow \dots \rightarrow p_l \rightarrow p_{l+1}$  of length  $l$  becomes  $p_1 \leftarrow p_2 \leftarrow p_3 \leftarrow \dots \leftarrow p_l$  of length  $(l - 1)$  after using it as a repair for faulty PE  $p_{l+1}$ .

This “reversal” of a replacement chain can be directly derived from the same argument from Figure 19. Figure 20 shows the illustration for this lemma.

### 3.3.2 Repair Chain Independence of Task-PE Model

To find out the conditions for guaranteeing  $k$  fault tolerances, we first need to define some terms:

A **Task Set**  $\mathbb{T}$  is a non-empty set of the Routers.

A **PE Set**  $\mathbb{P}$  of a given Task Set  $\mathbb{T}$ ,  $\mathbb{P}(\mathbb{T})$ , is the set of PE's which are currently Assigned to the Tasks in  $\mathbb{T}$ :  $\mathbb{P}(\mathbb{T}) = \{p_y | A(x, y) = 1 \wedge t_x \in \mathbb{T}\}$ .

**Gateway Set:** The Gateway Set of a Task Set  $\mathbb{T}$ ,  $GW(\mathbb{T})$ , is the set of all PE's (including functional and spare) which can replace the PE's in the PE Set  $\mathbb{P}(\mathbb{T})$ . Thus,  $GW(\mathbb{T}) = \{p_i | R(i, j) = 1 \wedge p_j \in \mathbb{P}(\mathbb{T})\}$ . Essentially, the size of the Gateway Set  $GW(\mathbb{T})$  represents the number of immediate PE's that can be used to replace any faulty PE's Assigned to Routers in  $\mathbb{T}$ . Thus, if the Gateway Set size of a Router Set is 0, any fault manifesting inside the corresponding PE Set is not repairable.

**Lemma 2:** After a repair of a faulty PE  $p$ , for any arbitrarily chosen Task Set  $\mathbb{T}$  in the system,  $|GW(\mathbb{T})|$  is reduced by at most 1.

To prove this lemma, we will investigate all the relationships that a faulty PE  $p$  may have with a given Router Set  $\mathbb{T}$ .

Case 1.  $p$  is Assigned to a Router in  $\mathbb{T}$ , thus  $p \in \mathbb{P}(\mathbb{T})$ , and replacement chain of  $p$  enters  $\mathbb{P}(\mathbb{T})$   $m$  times and exits  $\mathbb{P}(\mathbb{T})$   $(m - 1)$  times: This case is shown in Figure 21(a). In this scenario, size of  $GW(\mathbb{T})$  will be reduced by 1, since  $p$  uses up one of the incoming replacement chains for repair. However, if the overall replacement chain re-enters and exits  $\mathbb{P}(\mathbb{T})$  for the next  $(m - 1)$  times, as per Lemma 1, the replacement chain will be reversed. As a result, the incoming  $(m - 1)$  replacements become  $(m - 1)$  outgoing replacements and vice versa, yet  $|GW(\mathbb{T})|$  is reduced by 1 nonetheless, owing to the initial

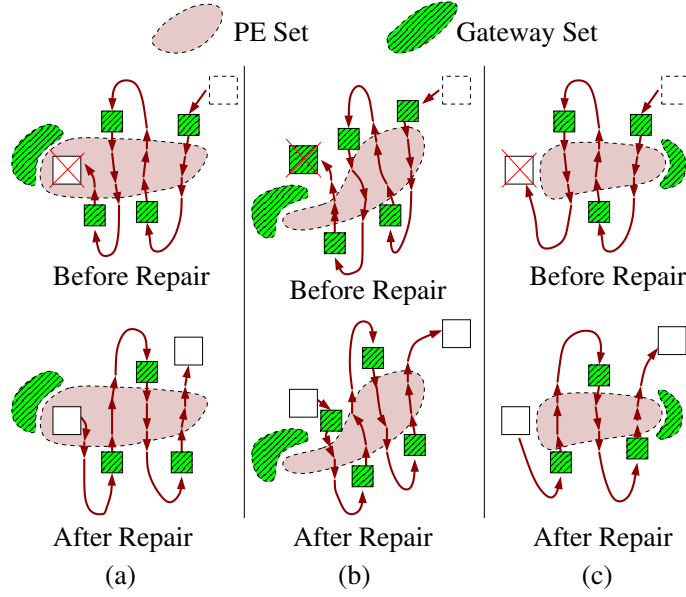


Figure 21: Illustration for Lemma 2, (a) Case 1:  $p \in \mathbb{P}(\mathbb{T})$ , (b) Case 2:  $p \in GW(\mathbb{T})$ , (c) Case 3:  $p \notin \mathbb{P}(\mathbb{T})$  and  $P \notin GW(\mathbb{T})$ .

usage of a replacement chain by  $p$ . Overall, if  $p \in \mathbb{P}(\mathbb{T})$ , after the fault is repaired (via a replacement chain),  $|GW(\mathbb{T})|$  is decreased by 1.

Case 2.  $p \in GW(\mathbb{T})$  and replacement chain of  $p$  enters and exits  $\mathbb{P}(\mathbb{T})$   $m$  times: as is shown in Figure 21(b), the size of  $GW(\mathbb{T})$  will be reduced by 1 in this scenario. Similar to the previous argument, when the replacement chain enters and exits  $\mathbb{P}(\mathbb{T})$ ,  $m$  times, due to the reversal of the chains,  $|GW(\mathbb{T})|$  will be reduced exactly by 1.

Case 3.  $p \notin \mathbb{P}(\mathbb{T})$  and  $p \notin GW(\mathbb{T})$ , and we assume the replacement chain of  $p$  enters and exits  $\mathbb{P}(\mathbb{T})$ ,  $m$  times: as is shown in Figure 21(c). In this case, all the  $m$  incoming chains to PE's executing tasks in

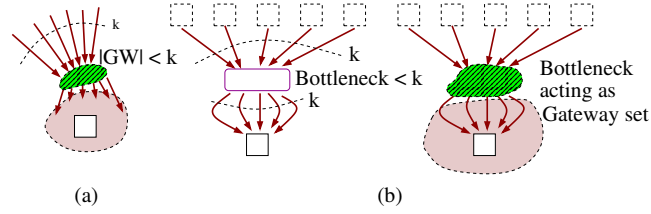


Figure 22: Illustration of proof of main claim: (a)  $1 \Rightarrow 2$ , (b)  $3 \Rightarrow 1$ .

$\mathbb{P}(\mathbb{T})$  are reversed, and they become  $m$  outgoing chains. Similarly, the  $m$  outgoing chains get converted to  $m$  incoming chains, resulting in no change in  $|GW(\mathbb{T})|$ .

Thus, in all the cases, the size of the Gateway Set reduces by at most 1 after a successful repair, no matter where the faulty PE  $p$  is with respect to the Router Set  $\mathbb{R}$ .

**Claim of Conditions for guaranteed  $k$ -FT Systems:** The following statements are equivalent:

**1. Each PE has at least  $k$  distinct (node-disjoint) incoming replacement chains from spares in the Replacement Graph.**

$\forall$  PE  $p_y$ , the system must have at least  $k$  replacement chains to  $p_y$  such that:  $\forall i \in [1, k]$ , a) chain  $RC_i$  starts at  $p_y$ ; b) chain  $RC_i$  ends at a spare; c)  $\forall RC_i$ , except for  $p_y$ , all the other nodes appear only once.

**2. All the Task Sets in the system has Gateway Set size  $\geq k$ :**  $\forall \mathbb{T} = \{t | t \in \mathbb{T}_n\}$ ,  $|GW(\mathbb{T})| \geq k$ .

**3. A system is guaranteed  $k$ -FT:**  $\forall$  fault sequence  $(p_{f1}, p_{f2}, p_{f3}, \dots, p_{fk})$ ,  $\exists k$  valid repairs via replacement chain in sequence for each fault, to deliver  $k$  successful repairs.

**$1 \Rightarrow 2$ :** If each PE in the system has at least  $k$  distinct incoming replacement chains, then each arbitrarily chosen Router Set  $\mathbb{T}$  in the system has  $|GW(\mathbb{T})| \geq k$ .

We will prove this by negation ( $\neg 2 \Rightarrow \neg 1$ ). Considering that there exists a particular Router Set  $\mathbb{T}$ , which contains a task executed by PE  $p$  ( $\in \mathbb{P}(\mathbb{T})$ ), has  $|GW(\mathbb{T})| = k_1$  and  $k_1 < k$ , then any PE  $p$  Assigned to a Router in  $\mathbb{T}$  must have the total number of incoming chains to be  $\leq k_1$ . This indicates that the number of distinct incoming chains to  $p$  is  $\leq k_1 < k$ . The illustration is given in Figure 22(a).

**2  $\Rightarrow$  3: if each Router Set in the system has Gateway Set size  $\geq k$ , then the system is guaranteed  $k$ -FT.**

From Lemma 2, we know that the Gateway Set size of any Router Set is reduced by at most 1 following a successful repair. Since a Gateway Set size of 1 will guarantee a repair for 1-FT, if all the Router Sets have Gateway Set size of  $\geq k$ , by mathematical induction, irrespective of where the fault manifests, it is guaranteed that all the  $k$  faults can be tolerated.

**3  $\Rightarrow$  1: If a system is guaranteed  $k$ -FT, then each PE in the system must have at least  $k$  distinct replacement chains from spares in the Replacement Graph.**

We will prove this claim by negation ( $\neg 1 \Rightarrow \neg 3$ ). Illustration for this proof is presented in Figure 22(b). Assume a PE  $p$  is having a maximum of  $k_1$  distinct incoming chains from spares, where  $k_1 < k$ , thus violating statement 1. This set of PE's effectively act as a bottleneck for the chains for  $p$ , and effectively form a Gateway Set, with the corresponding PE Set  $\mathbb{P}(\mathbb{T})$  of Router Set  $\mathbb{T}$  located right below it, as shown in Figure 22(b). Following Lemma 2, if all the next  $k_1$  faults occur within  $\mathbb{P}(\mathbb{T})$  or  $GW(\mathbb{T})$ ,  $|GW(\mathbb{T})|$  becomes 0. As a result, if any of the next  $(k - k_1)$  faults occur within  $\mathbb{P}(\mathbb{T})$ , it is rendered unrepairable, thus the system becomes Partial  $k$ -FT.

**Corollary:** The  $k$ -FT system designed following the Router-PE relationship model is Replacement Chain Independent.

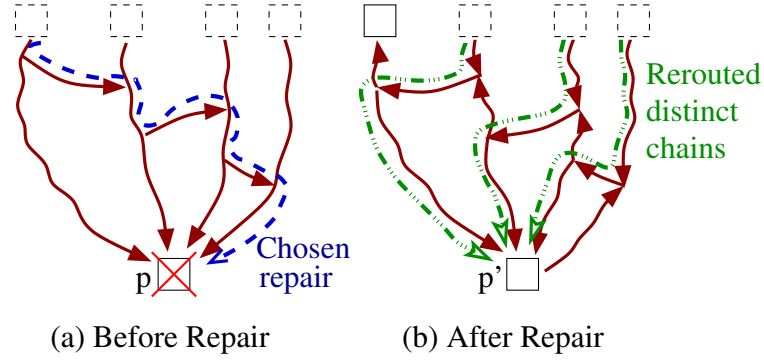


Figure 23: Illustration of the Repair Chain Independence: when a PE chooses an arbitrary chain

To prove Lemma 2, an arbitrary replacement chain is chosen for repair. Irrespective of which chain is chosen, it holds that the size of the Gateway Set is reduced by at most 1. Thus, if we make the Gateway Set size of all the Router Sets to be  $\geq k$ , any replacement chain as a repair choice for the current fault will maintain to be always able to tolerate at least  $k$  faults, making it Replacement Chain Independent.

### 3.3.3 Illustration of the Conditions

Figure 23 shows an illustration when a PE has 4 disjoint chains from 4 spares. When the PE is faulty, and an arbitrary chain is chosen, apparently it seems that the choice would render all the other disjoint chains disappear. However, due to the Reversal property of chains, it they do not disappear, rather get rerouted through another path, as shown in Figure 23(b). As a result, 4 disjoint chains becomes 3 disjoint incoming chains into the PE. If the PE chose a disjoint chain for repair, the outcome would be the same, supporting our claim of Replacement Chain Independence.

Figure 24 shows how a PE is affected when another PE chooses an arbitrary chain for repair, intersecting the disjoint chains of the former. The outcome is identical: the former PE, with 4 disjoint chains

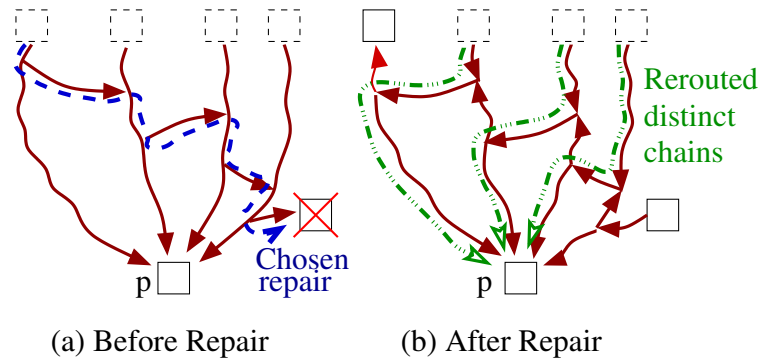


Figure 24: Illustration of the Repair Chain Independence: effect on a PE when another PE chooses an arbitrary chain

in Figure 24(a) ends up with 3 rerouted disjoint chains, as shown in Figure 24(b), this confirming the Repair Chain Independence.

Another illustration is shown in Figure 25, where, each of the shown PE's has 3 disjoint chains from 5 spares. When a PE is faulty, it can choose any of the spares for repair. However choosing one spare, shared by the other PE, may affect the fault tolerance of the other PE. This is not the case, as is shown in

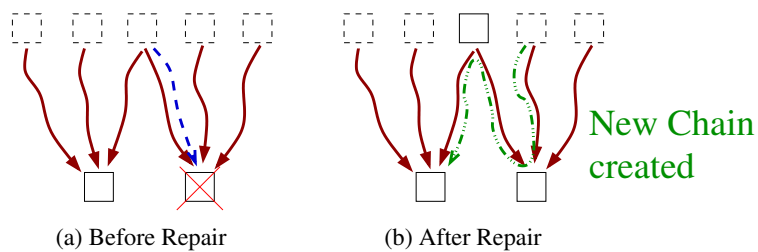


Figure 25: Effect of choosing of the shared spare by the faulty PE

Figure 25(b), where the other PE, although loses a spare, a new disjoint chain is created from the other spares, from which it did not have a path before. Thus even though the faulty PE loses a spare, the other PE remains unaffected.

### 3.3.4 Discussion

We have provided the necessary and sufficient conditions for a Task-PE based system to be guaranteed  $k$ -FT, assuming the repairs to be carried out by any arbitrary replacement chain. The proposed model and proven conditions lay out a foundation of identifying the properties of large scale systems to tolerate a high number of faults. Since the conditions guarantee a fault-independent and replacement-chain independent repair, they provide an analytic and construction basis to make a system capable of on-the-fly recovery upon every fault occurrence, with repairs carried out in a decentralized manner.

## 3.4 Physical Implementation of Task-PE Model

In the context of Structural Fault Tolerance, when a PE  $p_1$  replaces another PE  $p_2$ , it indicates that  $p_1$  is connected to all of the neighbors of  $p_2$ , as shown in Figure 26(a). However, this fundamental concept is problematic, since the interconnect overhead is dependent upon the underlying the topology itself. Dense networks will require large interconnect overhead in this case.

To make the interconnect overhead independent of the underlying topology, we propose a system consisting of PE's, Routers and Switches, as shown in Figure 26(b). Effectively, we replace the Tasks in Task-PE model with Routers. In such a system, the underlying topology is realized by the interconnects between the Routers. On the other hand, each Router is assigned exactly one PE, through the configuration of Switches. These constraints are identical to the ones in the Task-PE Model. If  $p_1$  replaces  $p_2$ , the corresponding connection between  $p_1$  and Router  $r_2$  is activated by configuring the Switches.

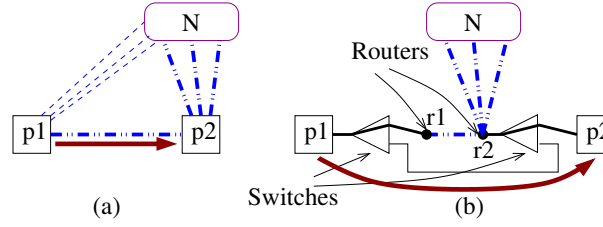


Figure 26: (a) Replacement of one PE by another: requiring the latter to be connected to former's neighbors; (b) An alternative implementation using switches and Routers

In this way, we can make sure that the original topology is maintained while the interconnect overhead (connections between Routers and PE's through Switches) remains independent of the topology.

It can be noted that, we essentially uncoupled the task of Routers from the PE's, thus dividing the components into more reliable (Router) and less reliable (PE) components. Since Routers are less complex in construction, they are less likely to manifest faults. As a result, previous approaches supporting simple Direct Replacement by spares can be applied. On the other hand, PE's are more likely to develop faults. Thus, in this paper, we concentrate on fault tolerance in PE's only.

The proposed system is shown in Figure 27. Assume, the required topology is a tree given in Figure 27(a). The corresponding system is shown in Figure 27(b). As evident, the **Backbone Network** is responsible for implementation of the tree topology. On the other hand, the two sets of vertices: **Routers** and **PE's** are interconnected via a switch network, namely the **Auxiliary Network**. The Auxiliary Network is configurable to change which Router is connected to which PE. To maintain the required topology, each Router must be connected to a PE. In case of a fault manifestation on a PE, as an example, in PE  $p_6$ , the corresponding Router  $r_4$  is now without any PE, disrupting the topology. To

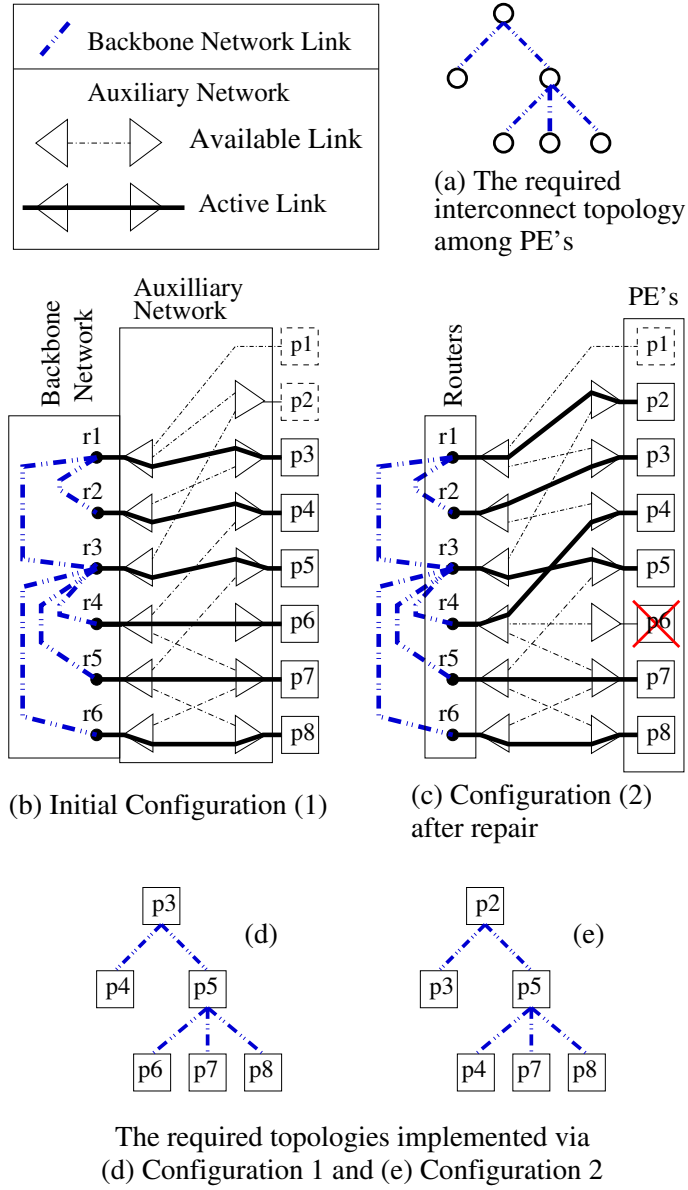


Figure 27: The proposed Router-PE System with a fault tolerance illustrated

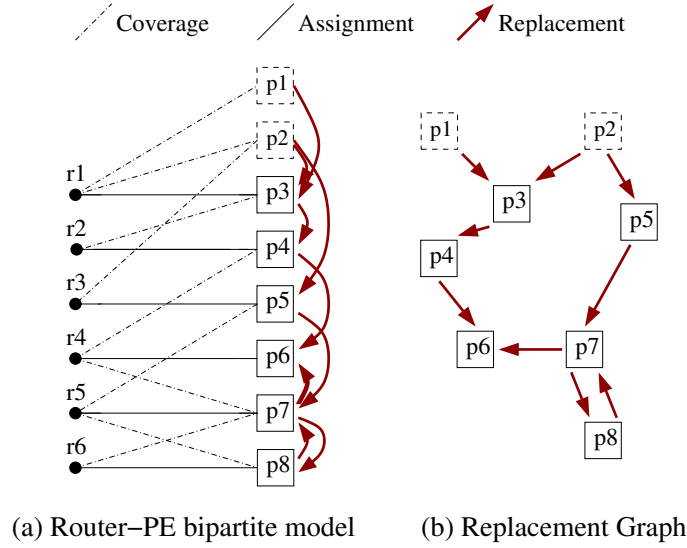


Figure 28: The Router-PE system corresponding to the system in Figure 27, isomorphic to the Task-PE model

retain the topology,  $r_4$  must convert an *available link* (Coverage in Task-PE Model) into an *active link* (Assignment in Task-PE Model). Thus, the available link from  $p_4$  is converted into an active link, with active link between  $p_4$  and  $r_2$  becomes available, but not active. Thus  $r_2$  loses its active link, and thus configures the available link from  $p_3$  into an active link. This goes on until the spare  $p_2$  is reached. The corresponding topologies before and after the repair are shown in Figures 27(d) and (e) respectively.

Thus, effectively, the Router-PE network is isomorphic to the Task-PE model, where the Routers are equivalent to the Tasks. The “task” of Routers is to maintain the communication. To maintain the communication and processing, each Router must be Assigned one PE. The fault tolerance is also done by interchanging the Assignments and Coverages along the chain. Thus, the Replacement Chain Algebra is also valid for the Router-PE network.

The system can also be represented using a Router-PE bipartite model, as shown in Figure 28(a), similar to the Task-PE Model. In this model, we ignore the Backbone Network since it is fixed for a given system. In contrast, the Auxiliary Network changes according to faults and the corresponding repairs. The Auxiliary Network is represented by two types of edges: Coverage and Assignment, corresponding to the Available and Active Links in the Router-PE system in Figure 27.

In this chapter, we perform a case study of fault tolerance in the immensely popular 2D Grid topology of the PE's, implemented using the proposed Router-PE Model. The aim of the design will be to limit the interconnect overhead by making them strictly local to make sure that there do not exist long interconnects. Furthermore, keeping the interconnect local keeps the system scalable for larger number of PE's.

Figure 29(a) shows the co-ordinance of Routers and PE's placed in a 2D Array layout. Assume a PE  $p_i$  is located at  $(x_{pi}, y_{pi})$ , and a Router  $r_j$  is located at  $(x_{rj}, y_{rj})$ . PE  $p_i$  is located in  $r_j$ 's **Immediate Neighborhood** if  $|x_{pi} - x_{rj}| + |y_{pi} - y_{rj}| \leq 1$ .

Figure 29(b) shows a localized Auxiliary Network built on the immediate neighborhoods for coverage of the system with arbitrary assignments. The spares are distributed among the system due to this property. Figure 29(c) shows the corresponding replacement relationships between the PE's.

We will analyze the system's scalability in terms of interconnect overhead. Overall, each Router has 4 coverages from the immediate neighborhood, one of which is assigned and the rest 3 are configurable. Thus, the interconnect overhead remains as  $O(3n)$ . This ensures the high scalability of the system.

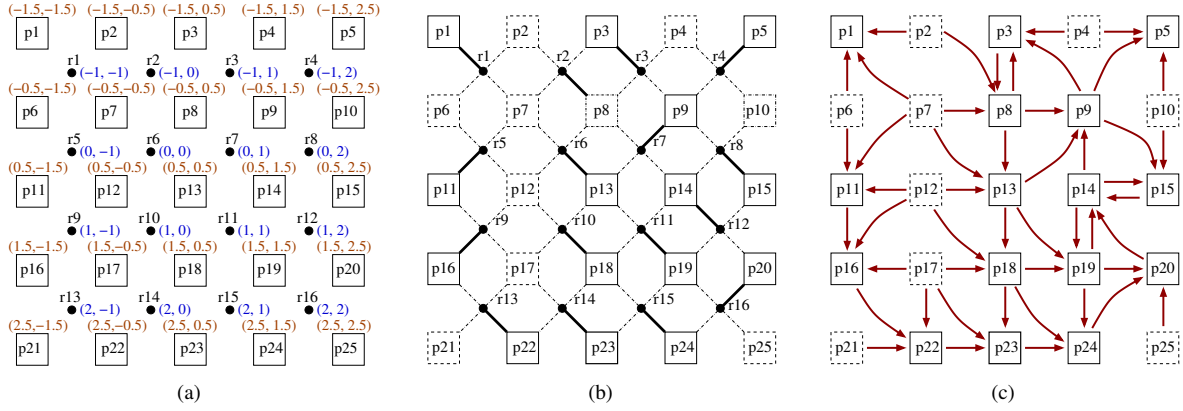


Figure 29: a) The  $4 \times 4$  2D Array layout with the corresponding coordinates assigned, b) system with spares distributed around in the system with local coverages, c) Replacement relationships among the PE's from the configuration in (b) ©IEEE 2017

### 3.4.1 Repair Strategy and Reliability Analysis

The repair in the system is done on-the-fly, i.e., after every fault manifestation, through a series of reconfigurations. An example of two such repairs is shown in Figure 30. The 4<sup>th</sup> fault at  $p_{24}$  in the system can be repaired through either the chain  $p_{24} \leftarrow p_{19} \leftarrow p_{14} \leftarrow p_{15} \leftarrow p_{10} \leftarrow p_4$ , or  $p_{24} \leftarrow p_{18} \leftarrow p_{13} \leftarrow p_7$  among other choices, as is shown in Figures 30(b) and 30(c). From the conclusions from the previous chapter, we can conclude that none of the repair option changes the future fault tolerance capabilities of the system. As a result, the “least cost” replacement chain can be chosen for repair. The cost here may refer to the total number of reconfigurations, or anything related.

For the system in Figure 29(b), based on the condition in the previous chapter, the number of disjoint replacement chains is 3 for all of the PE's. Although each PE has incoming chains from a large number of spares, the bottleneck is formed by the 3 PE's, directly replacing the PE under considera-

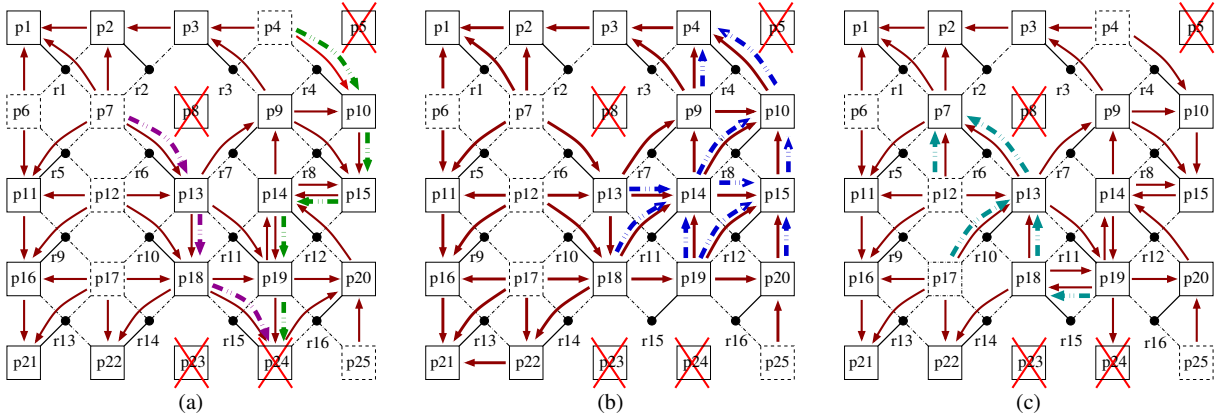


Figure 30: An example repair in a  $4 \times 4$  Array implemented on a  $5 \times 5$  PE Array a) after 3 faults are repaired, b) after the 4<sup>th</sup> fault at  $p_{24}$  is repaired through chain  $p_{24} \leftarrow p_{19} \leftarrow p_{14} \leftarrow p_{15} \leftarrow p_{10} \leftarrow p_4$ ; and c) using alternate replacement chain of  $p_{24} \leftarrow p_{18} \leftarrow p_{13} \leftarrow p_7$ . The changed replacements are also highlighted ©IEEE 2017

tion. Therefore, this system can tolerate guaranteed any 3 faults. In reality, the system might tolerate more faults, up to the number of spares, as the condition given in the previous chapter provides only the lower-bound. This claim will be verified in the simulation results section.

### 3.4.2 Reliability Enhancement

The proposed system in Figure 29(b) has a fixed reliability performance. To enhance reliability, two ways can be adopted:

**Type-1: Increased Spares:** introduce more spares to make the system likely to tolerate more faults.

**Type-2: Enlarged Coverage:** have each Router covered by more PE's.

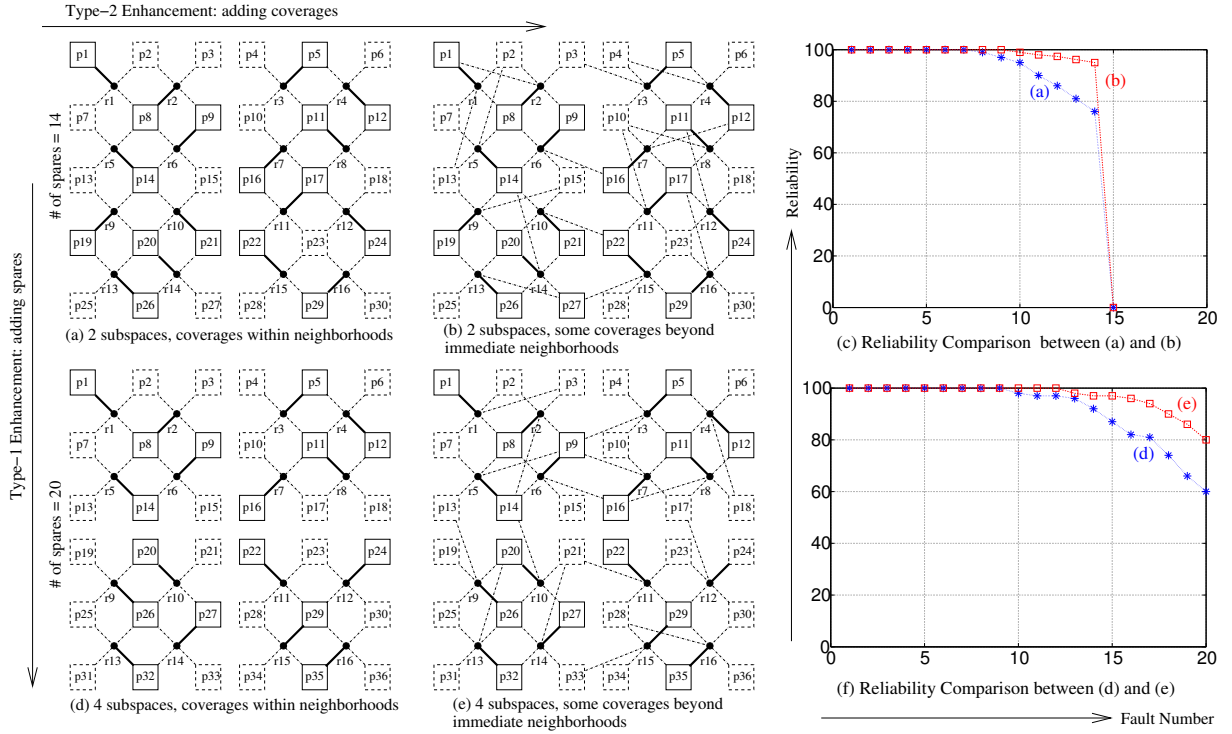


Figure 31: a) The  $4 \times 4$  Router space divided into 2  $4 \times 2$  spaces, resulting in increase in total number of spares, b) type 2 reliability enhancement by allowing some performance degradation over (a), c) Reliability comparison between (a) and (b), d) same space divided into 4  $2 \times 2$  spaces, resulting in further increase in total number of spares, and d) type 2 reliability enhancement by allowing more performance degradation over (e), f) Reliability comparison between (d) and (e) ©IEEE 2017

The first type of reliability enhancement can be achieved by dividing the space into multiple “subspaces”, as shown in Figure 31, where Figure 31(a) introduces two subspaces with 14 spares, and Figure 31(d) with 4 subspaces and 20 spares.

In general, an array containing  $l \times w$  Routers can be divided into equal sized  $ss_l \times ss_w$  number of subspaces, with each subspace containing  $(l/ss_l) \times (w/ss_w)$  Routers. Each subspace also contains  $((l/ss_l) + 1) \times ((w/ss_w) + 1)$  PE's. Thus, the total number of spares is  $((l/ss_l) + 1) * ss_l \times ((w/ss_w) + 1) * ss_w - (l \times w)$ , which can be customized by varying  $ss_l$  and  $ss_w$ .

Second type of reliability enhancement can be achieved with additional coverages added in the system. In general, **Neighborhoods** can be defined as the following: for a PE  $p_i$  located in a Router  $r_j$ 's immediate neighborhood, we consider it to be in Neighborhood(1) with respect to  $r_j$  or  $NB(i, j) = 1$ . For non-immediate neighborhoods, a PE  $p_i$  located at coordinates  $(x_{pi}, y_{pi})$  is related to a Router  $r_j$  located at  $(x_{rj}, y_{rj})$  as  $NB(i, j) = m$ , if  $(m - 1) < |x_{pi} - x_{rj}| + |y_{pi} - y_{rj}| \leq m$ . As an example,  $NB(10, 7) = 2$  while  $NB(10, 2) = 3$  in Figure 29(a).

For scalability and cost considerations, the *added coverages should be from the nearest possible neighborhoods*. Introduction of 1 additional coverage per Router from Neighborhood(2) is shown in Figures 31(b) and 31(e). Figure 31(c) shows the reliability comparison between the 2-subspace with local coverage design and the 2-subspace with 1 non-local coverage per Router design, with various initial Assignments, achieved through Monte-Carlo simulation. The system, having 14 spares, can tolerate a maximum of 14 faults. However, for the scenario where 1 non-local coverage per Router is present, the reliability (of around 95%) is higher than the case when all the coverages are local (around 75%) for the 14th fault. Figure 31(f) shows the type-1 reliability enhancement over Figure 31(c) by having 4 subspaces and 20 spares. The system then can tolerate a up to 20 faults. Like the previous scenario, the reliability of the system with 1 non-local coverage per Router, tolerating 20 faults is around 80%, higher than the system with all local coverages, delivering around 60% in reliability.

If a PE is assigned to a Router not in its immediate neighborhood, there will be a performance degradation in terms of increased delay. The delay is proportional to the square of the distance between the Router and the PE.

We define **Normalized Delay** ( $D$ ) of the system to be  $D = (\sum_{i=1}^{n+s} \sum_{j=1}^n (|x_{pi} - x_{rj}| + |y_{pi} - y_{rj}|)^2) / n$ , when PE  $p_i$  located at  $(x_{pi}, y_{pi})$  is Assigned to Router  $r_j$  located at  $(x_{rj}, y_{rj})$ , in a system with  $n$  Routers and  $(n + s)$  PE's. Thus,  $D$  indicates the delay of communication between the Routers and the assigned PE's, averaged over all the Routers.

From the given definition of  $NB(i, j)$ , it can also be shown that, for the given Processor Array,  $|x_{pi} - x_{rj}| + |y_{pi} - y_{rj}| = NB(i, j)$ . Thus,  $D = \sum_{i=1, j=1}^{i=n+s, j=n} (NB(i, j))^2 / n$ , when  $p_i$  is assigned to  $r_j$ . In the ideal case, when each PE is assigned to a Router in its immediate neighborhood,  $D = 1$ .  $D$  starts to increase when PE's are assigned to Router, not in its immediate neighborhood.

### 3.4.3 Experimental Results

Monte-Carlo Simulation was performed in C on  $10 \times 10$ ,  $20 \times 20$  and  $32 \times 32$  Router Array to compare the performances for mesh topology in approaches proposed in  $M_4$  (Bruck et al., 1993), 3-track switch (Varvarigou et al., 1993), 4-track switch (Mahapatra and Dutt, 2001) which is derived from 2D Parity Code approach (Dutt and Mahapatra, 1997) and Deterministic Design in (Zhang, 2002). 1000 trials are run with varying Assignments each time to evaluate the number of faults that can be tolerated in each trial. The *reliability* is determined as the percentage of times any fault is successfully repaired.

#### 3.4.3.1 Reliability Comparisons

Figure 37 shows the reliability comparisons between  $M_4$  (Bruck et al., 1993), 3-Track-switches (Varvarigou et al., 1993), 4-Track-switches (Mahapatra and Dutt, 2001) and the proposed approach

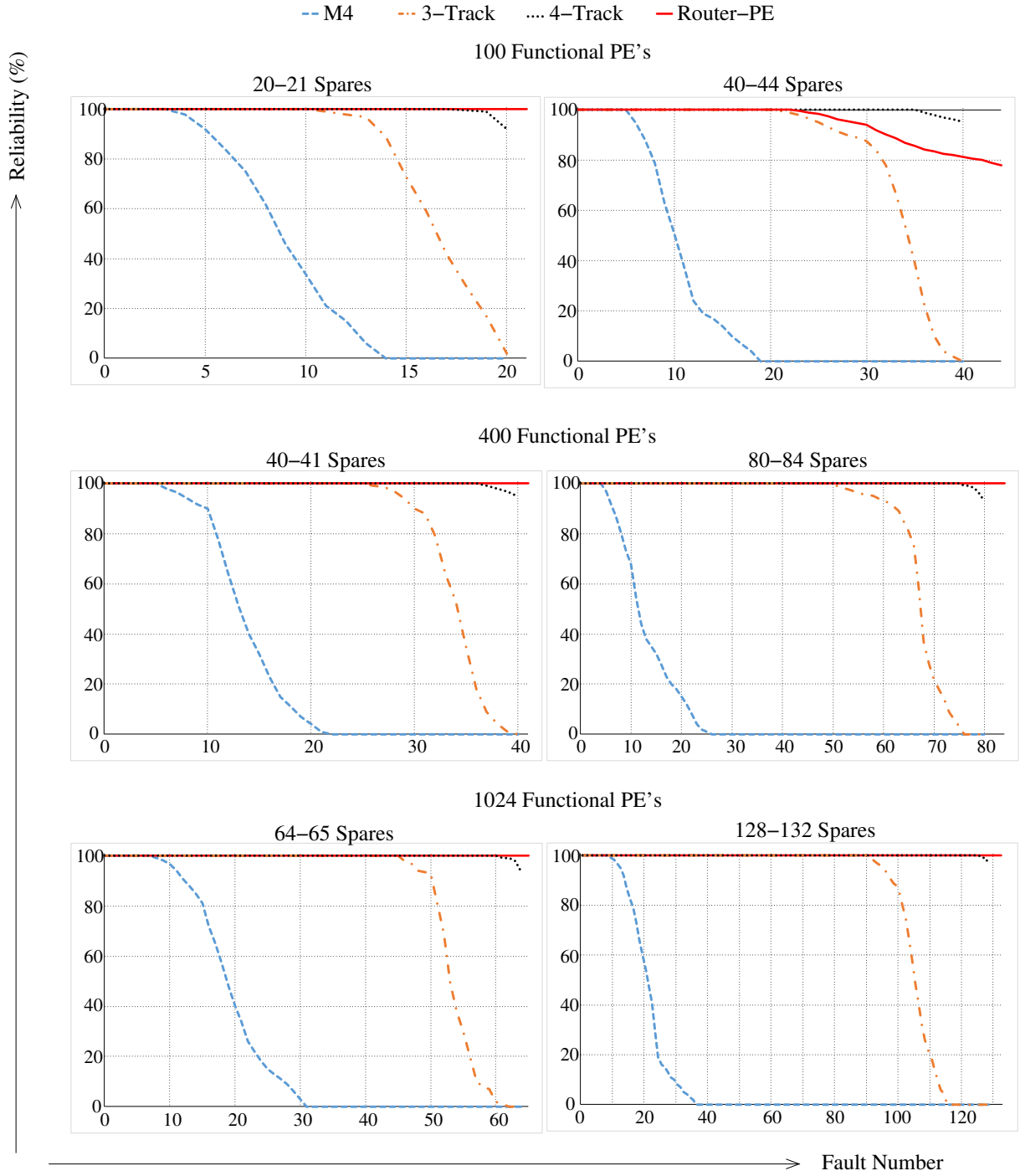


Figure 32: Reliability comparison between the previous works and the proposed approach

for  $10 \times 10$  Arrays. The results are obtained from (Dutt and Mahapatra, 1997) and (Mahapatra and Dutt, 2001). We have ignored (Zhang, 2002) in this case, since it is a deterministic design, always guaranteeing 100% reliability. Due to the difference in design methodologies, the proposed approach uses a varying number of spares. This is due to the fact that, the proposed approach considers a complete PE Array, including PE's at the corners, while the previous approaches places the spares only at the perimeters and not on the corners.

For 20/21 spares, in Figure 37, our approach provides superior reliability of around 100% when 20 faults are tolerated. The same is around 90% for (Mahapatra and Dutt, 2001), and almost 0% for both (Bruck et al., 1993) and (Varvarigou et al., 1993). For 40/44 spares, in Figure 37(b), (Mahapatra and Dutt, 2001) delivers almost 100% reliability for the fault 40, while the proposed approach delivers reliability of around 80%. Approaches (Bruck et al., 1993) and (Varvarigou et al., 1993) delivers significantly less reliability. For larger systems with  $20 \times 20$  and  $32 \times 32$  Routers, the proposed approach almost always guarantees 100% reliability, followed closely by 4-Track switches approach.

Comparing with the smaller system results obtained in Figures 31(c) and 31(f), and the larger systems of  $20 \times 20$  and  $32 \times 32$  Router grids, it can be noted that, the probability of tolerating the maximum number of faults increases as the number of PE's grows in the system. For 4 sub-spaces in  $4 \times 4$  system,  $10 \times 10$  and  $20 \times 20$  systems with strictly local coverages, the reliabilities to tolerate the maximum number of faults (20, 44, 84 respectively) are 60%, 80% and 100% respectively, showing the reliability scaling in our system with growing number of PE's.

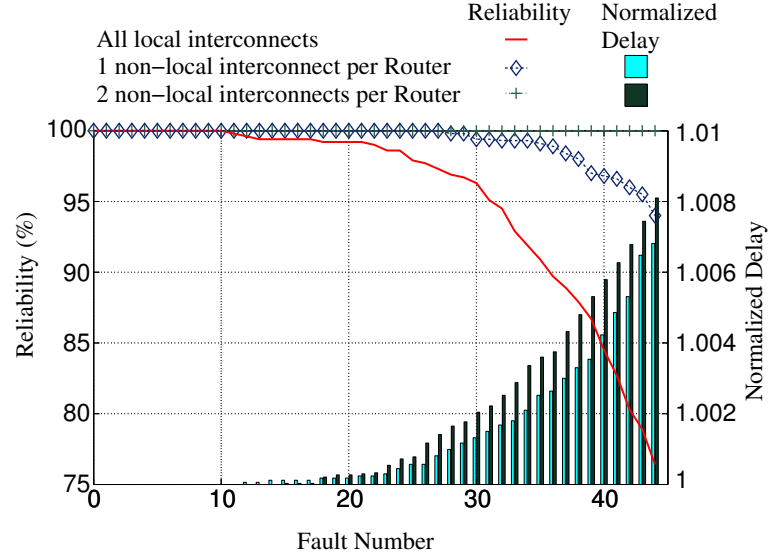


Figure 33: Reliability enhancement in the proposed approach by allowing Normalized Delay to increase ©IEEE 2017

### 3.4.3.2 Reliability Enhancement Type 2: by Enlarged Coverage

The reduced reliability delivered by the proposed approach, shown in Figure 37, is due to the fact that, there are 4 sub-spaces, the a PE in a sub-space only has access to fewer spares (a quarter of all the spares). To enable access to a larger number of spares, type 2 reliability enhancement can be considered. Figure 33 shows how the reliability can be enhanced by adding 1 or 2 additional coverages per Router<sup>1</sup>. For fault 44, without any non-local coverage, the reliability is around 75%, which is enhanced to 95% and 100% when 1 and 2 additional coverages are added respectively. The corresponding Normalized

<sup>1</sup>The repair gives higher priority to chains resulting in lowest delay over the shortest ones, unlike the case where all the coverages were strictly local.

$M_4$ (Bruck et al., 1993)	$O(10n)$
3-Track Switches (Varvarigou et al., 1993)	$O(6n)$
4-Track Switches (Mahapatra and Dutt, 2001)	$O(8n)$
Deterministic in (Zhang, 2002)	$O(n \log^3 k)$
Proposed	$O(3n)$

TABLE II: REDUNDANT INTERCONNECT COMPLEXITY COMPARISONS AMONG VARIOUS APPROACHES ©IEEE 2017

Delays can also be noted. With 2 additional coverages, it delivers higher reliability but while allowing higher delay. As an example, when there is 1 additional coverage each Router, for fault 44, the Normalized Delay is around 1.0068. This indicates that, out of 5 instances of tolerating 44 faults, there will be approximately a single scenario, when a single Router has a PE assigned not from its immediate neighborhood<sup>1</sup>. When there are 2 additional coverages, the same value is 1.008, indicating that, out of 4 cases of tolerating 44 faults, in approximately one case there will be a Router which has a PE assigned not from its immediate neighborhood.

---

<sup>1</sup>The overall delay is approximately  $1.006 \times 100 = 100.6$ . For 5 instances, the overall delay is 503, which indicates that, 4 instances have overall delay of 100 each and one with 103. The first 4 instances have all the Assignments constrained within immediate neighborhoods, while in the final case, 99 Routers have Assigned PE's in their immediate neighborhood, and one has Assigned PE in Neighborhood(2), thus the delay is 4 for this Router. The overall delay for this case is 103.

### 3.4.3.3 Hardware and Interconnect Overhead

The hardware overhead is linearly proportional to the number of spares in the system. In this sense, all the approaches in (Bruck et al., 1993), (Varvarigou et al., 1993), (Mahapatra and Dutt, 2001) and the proposed approach have comparable amount of hardware overhead.

Table 1 shows the Redundant Interconnect Overheads for the different approaches. The proposed approach, in the basic situation, requires 3 non-assigned coverages per Router. Thus, there are additional  $O(3n)$  interconnects overall. When enhancing Type-2 reliability, the Interconnect Complexity increases. When there is 1 additional coverage per Router, each interconnect will have a length of 2, which, in the overall system becomes  $O(2n)$  of non-local interconnects for all the Routers. The Redundant Interconnect Overhead becomes  $O(5n)$ . When there are 2 additional coverages, the same becomes  $O(7n)$ . Still, introducing even a small number of additional interconnects have the potential to enhance the reliability significantly, as shown from the experimental results. Furthermore, for larger systems, the proposed design methodology will require less than or equal to  $O(7n)$  interconnect overhead, to deliver around 100% reliability for maximum number of faults tolerable, as shown from the reliability scaling noted by comparing  $4 \times 4$  and  $10 \times 10$  systems.

### 3.4.3.4 Average Lengths of Replacement Chains

Figure 34 shows how the lengths of the replacement chains change depending on the fault number in the sequence, assuming the shortest replacement chains are always selected for repair. Figure 34 shows the results for  $10 \times 10$  Router Array implemented on a  $11 \times 11$  PE Array and  $20 \times 20$  Router Array implemented in a  $21 \times 21$  PE Array. Overall, the system performs repairs with rather short replacement chains of lengths around 2 for the  $10 \times 10$  system, while there is a marginal increase in the lengths with

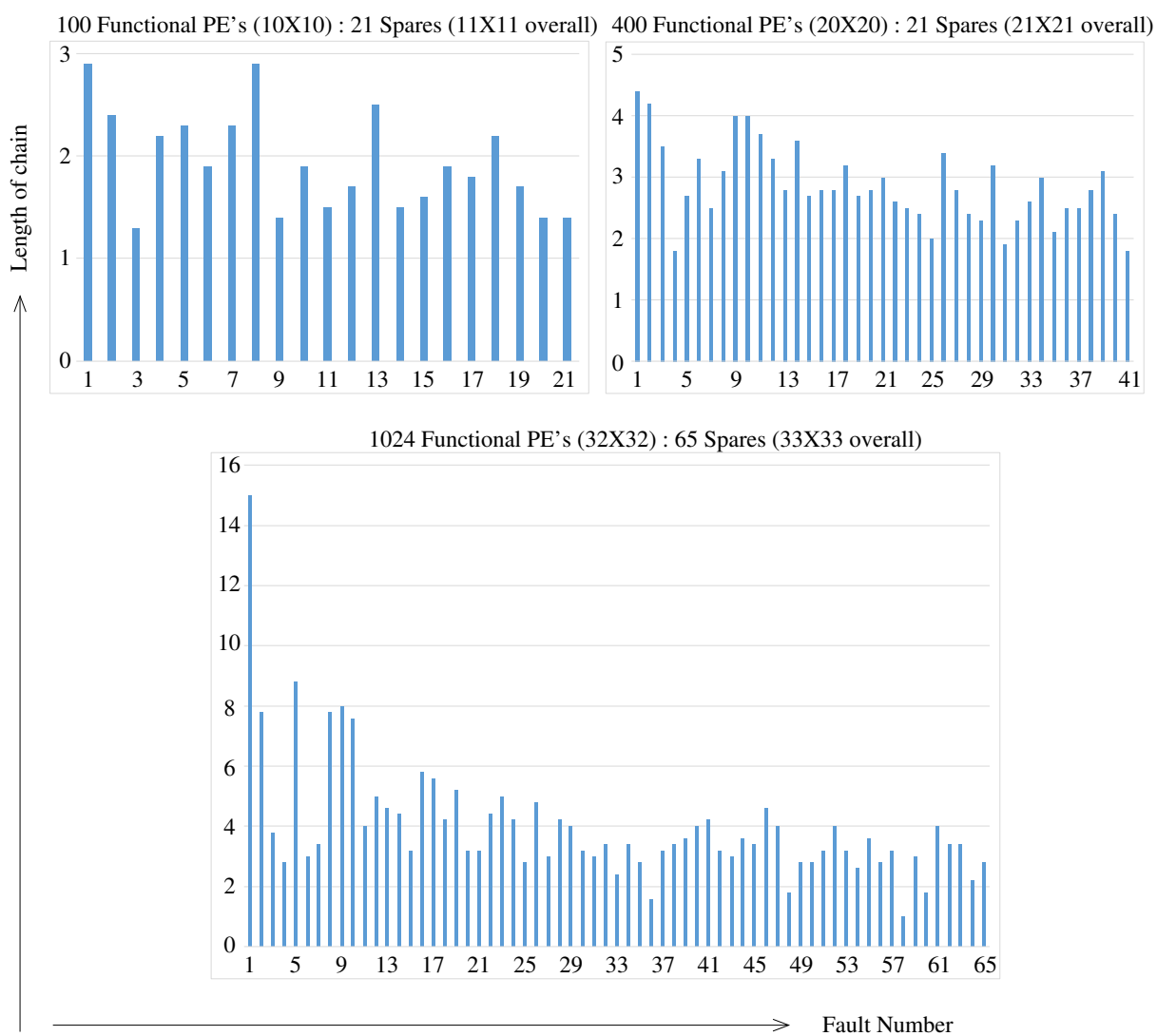


Figure 34: Average lengths of Replacement Chains for various systems

the  $20 \times 20$  system at 3. For significantly larger system of  $32 \times 32$  Router Array, while the length of the first repair chain is around 15, for the rest of the faults, the average length is around 5, again showing marginal increase in chain lengths.

#### 3.4.4 Discussion

This chapter introduces a novel approach for designing the immensely popular 2D Array of Processors with on-the-fly fault tolerance capabilities. The redundant interconnects, as shown, are strictly local, allowing the system to be scalable with more number of components. Following the 2-layered Router-PE model, it can be concluded that the repairs are lightweight, and can take place through any arbitrary replacement chain. We have shown the guaranteed deliverable fault tolerance ability of such systems, and how the reliability can be enhanced by introducing more spares, or extending coverages to larger neighborhoods. Simulation results confirm the high level of reliability of the proposed approach, and how the enhanced reliability can be achieved with scalable costs in hardware and delay.

### 3.5 Future Work: Construction of $k$ Fault Tolerant Systems

Assume a 2D space consisting of placed  $s$  Routers and  $n$  PE is given along with their coordinates. This section deals with construction of  $k$ -FT ( $k \leq s$ ) system while minimizing the interconnect lengths. Of course, when the 2D space is given, we can assume that a PE can cover any Router system-wide. Consequently, a PE can replace any other PE. However, from the global Coverages, the aim is to find out the least cost coverages which can guarantee  $k$ -FT.

In a 2D space, the cost of a Coverage is given as  $cost_{i,j} = |x_i - x_j| + |y_i - y_j|$  when PE  $p_j$  Covers Router  $r_i$ . The overall cost of the  $k$ -FT system is given as  $\sum cost_{i,j} \forall$  PE  $p_j$  Covering Router  $r_i$ . However, to achieve such a system, we must convert this understanding into Replacement paradigm.

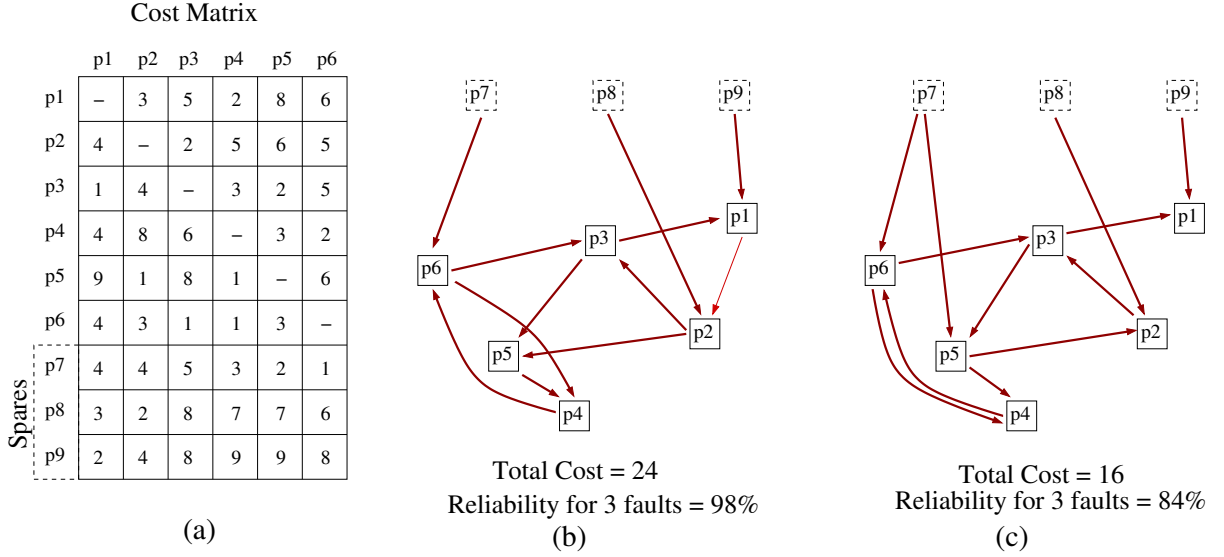


Figure 35: Systems with two different cost-reliability trade-offs (reliability obtained through simulation)

Recall that  $R_{i,j} = 1$  when PE  $p_j$  Covers the Router  $r$ , to which PE  $p_i$  is currently assigned to. Thus, we must first create the least cost Assignments before we can select the appropriate Coverages to achieve our goal.

The Least Cost Assignment problem can be solved optimally in Polynomial time using Hungarian Method (Kuhn, 1955). Following the Assignments, we obtain the Replacement possibilities. Their costs are same as the costs of the corresponding required Coverages. Consequently, we can obtain a Directed Graph. For such a graph, the Sub-graph consisting of the functional PE's is a complete graph. On the other hand, the spares can replace any other PE in the system. The problem is finding Least Cost  $k$ -disjoint paths from the spares to each of the functional PE's is a strongly *NP*-Complete problem (Li et al., 1992). Thus, we propose heuristic polynomial time algorithms for this problem.

Formally, the problem has two considerations: given placements of  $n$  Routers and  $(n + s)$  PE's, with the value of  $k$  ( $k \leq s$ ), such that the system is guaranteed  $k$ -FT:

- i) Minimize the overall cost or length of Coverages/Interconnects.
- ii) Maximize reliability of the system from fault numbers  $(k + 1)$  to  $s$ .

Two different approaches are proposed where one prioritizes one consideration over other, as shown in Figure 35. Figure 35(a) shows the cost matrix for a PE to replace another PE for a given Assignment. As an example, if  $p_5$  is to replace  $p_3$ , the cost of the corresponding Coverage is 8. Figures 35(b) and 35(c) show two different systems with guaranteed 2-FT. System in Figure 35(b) prioritizes reliability over cost, as it requires a cost of 24 while delivering a 98% chance of tolerating 3 faults. In contrast, system in Figure 35(c) prioritizes cost of reliability, with a cost of 16 and a 84% chance of tolerating 3 faults.

It can be noted that both of the systems can guaranteed tolerate 2 faults. However, there is still a difference when it comes to the 3<sup>rd</sup> fault. This is due to the fact that, for the first system, the spares  $p_7$ ,  $p_8$  and  $p_9$  all have paths to every single functional PE in the system while none such paths exist for the spare  $p_9$  in the second system, which only has a path to  $p_1$ . As a result, the PE's in the system has a smaller access to the spares, resulting in reduced reliability.

Formally, the aforementioned systems can be classified as following:

**Balanced Replacement Graph:** In such a graph, any two or more loosely connected components of the graph have equal functional PE to spare PE ratio. An example of the system is given in Figure 35(b).

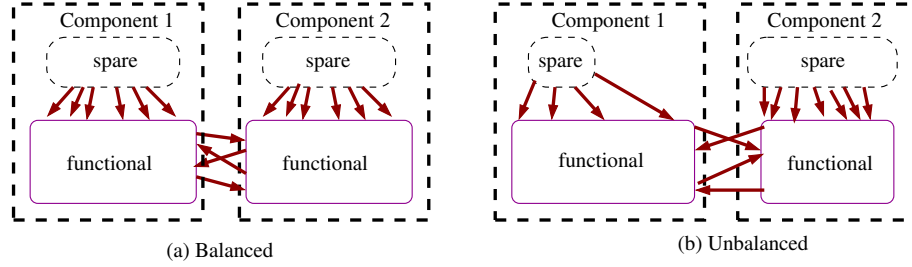


Figure 36: Balanced and Unbalanced Replacement Graphs

**Unbalanced Replacement Graph:** In such a graph, the ratio of functional PE to spare PE in different loosely connected components are unequal. An example of such a system is given in Figure 35(c). Two such loosely connected components are  $(p_1, p_9)$  and  $(p_2, p_3, p_4, p_5, p_6, p_7, p_8)$ , between which, there is only one replacement edge from the later to the former, resulting in an Unbalanced functional to spare PE ratio.

The comparisons between two such systems are given in Figure 36.

Another consideration we have for the system design is the maximum number of Coverages a PE can have. This is included to make sure that no PE has a very high degree compared to other PE's in the system. Assume a system has  $n$ -PE's and  $s$ -spares. Our aim is to make the system guaranteed  $k$ -FT ( $k \leq s$ ). Thus, each of the Routers require  $(k + 1)$  Coverages, which are provided by  $(n + s)$  PE's. Thus, the minimum value of the maximum degree of a PE is given as  $(n \times (k + 1)) / (n + s)$ .

### 3.5.1 Design of Balanced System

The motivation behind designing the system is that, each PE must have the access to maximum possible number of spares. Thus, we make  $s$  subgraphs of spanning trees in the system, with each of

**Data:** Coordinates of  $(n + s)$  PE's,  $n$  Routers, value of  $k$  such that system is  $k$ -FT, Maximum degree  $max\_deg$

**Result:** Balanced  $k$ -FT Replacement Graph with Minimal Interconnect Length

```

1 Apply Hungarian Algorithm to create the initial Least Cost Assignments;
2 for spares from 1 to s do
3    $incl \leftarrow NULL$ ;
4    $incl \leftarrow incl \cup spare$ ;
5   for Edge from 1 to  $(n \times k)/s$  do
6     for all PE  $p_i \in incl$  and  $max\_deg > deg[i]$  do
7       for all PE  $p_j \notin incl$  do
8         Find the minimum cost  $|x_i - x_j| + |y_i - y_j|$  for replacement from  $i$  to  $j$  such that
9         i) Number of disjoint chains to  $p_j$  increases by 1;
10        ii)  $p_j$  has minimum number of disjoint chains in the system;
11        iii)  $deg[incl \cup p_j] < max\_deg$  if Edge  $\neq (n \times k)/s$ ;
12      end
13    end
14     $R[j][i] \leftarrow 1$ ;
15     $C[r][j] \leftarrow 1$  such that  $A[r][i] = 1$ ;
16     $incl \leftarrow incl \cup p_j$ ;
17  end
18 end

```

**Algorithm 1:** Algorithm to design Balanced Replacement Graphs

**Data:** Coordinates of  $(n + s)$  PE's,  $n$  Routers, value of  $k$  such that system is  $k$ -FT, Maximum degree  $max\_deg$

**Result:** Unbalanced  $k$ -FT Replacement Graph with Minimal Interconnect Length

```

1 Apply Hungarian Algorithm to create the initial Least Cost Assignments, obtain set of spares;
2 for Step from 1 to  $k$  do
3    $incl \leftarrow NULL$ ;
4    $incl \leftarrow incl \cup spares$ ;
5   for all PE  $p_i \in incl$  and  $max\_deg > deg[i]$  do
6     for all PE  $p_j \notin incl$  do
7       Find the minimum cost  $|x_i - x_j| + |y_i + y_j|$  for replacement from  $i$  to  $j$  such that
8       i) Number of disjoint chains to  $p_j$  increases by 1;
9       ii)  $deg[incl \cup p_j] < max\_deg$ ;
10    end
11  end
12   $R[j][i] \leftarrow 1$ ;
13   $C[r][i] \leftarrow 1$  such that  $A[r][j] = 1$ ;
14   $incl \leftarrow incl \cup p_j$ ;
15 end

```

**Algorithm 2:** Algorithm to design Unbalanced Replacement Graphs

the subgraphs rooted at each of the spares, and each subgraph containing path from a spare to exactly  $(n \times k)/s$  functional PE's.

Thus, for Balanced Systems, the spares are chosen one at a time, and Replacement Edges (and subsequently Chains) are created from a spare to exactly  $(n \times k)/s$  functional PE's. As a result, every PE in the system is accessible from an equal number of spares. Algorithm 1 shows the technique of creation of such a system.

The Algorithm requires the coordinates of the PE's and the Routers. It also requires the value of  $k$  as well as the maximum degree  $max\_deg$  allowable for each PE. The first step is to create the initial Least Cost Assignments following the optimal Hungarian Algorithm. After the Assignment, we go for creation of the Replacement Edges from line 2 to line 15. First, we choose one spare at a time (line 2). Subsequently, we introduce a set named *incl* which indicates which PE's are currently accessible through the Replacement Edges from the current spare. Initialize *incl* to include the current spare. Then we go ahead create  $(n \times k)/s$  Coverages/Replacement Edges to same number of PE's. We evaluate all the possibilities of Replacement Edge between  $p_i \in incl$  and  $p_j \notin incl$ , such that degree of  $p_i$ ,  $deg[i] < max\_deg$  ( $p_i$  can Cover a Router). From these possibilities, we evaluate the smallest cost edge, which satisfies conditions (i) to (iii) in lines 9-11. Condition (ii) is included to make sure no single functional PE in the system utilizes majority of the least cost edges, leaving less choices for the other PE's. Condition (iii) guarantees that the new *incl* will cumulatively have at least 1 Coverage in the future to make sure it can grow further. The calculation of the number of disjoint chains is done by converting the given problem into max-flow problem from spares to each of the functional PE's.

### 3.5.2 Design of Unbalanced System

Motivation for design of such systems is that the overall cost must be least possible. This is achieved by finding least cost Replacement edges globally, so that the selected edge increases the maximum disjoint chains into the terminal of the edge by 1. Because the least cost edges are chosen globally, its overall cost is less compared to the Balanced system.

For Unbalanced Replacement Graphs, we go through a total of  $k$  steps. At each step, the number of disjoint chains for the PE's is increased by 1 through any arbitrary spare. Thus, there is a possibility that a spare may access a large number of functional PE's through low cost chains while some others can access less. Algorithm 2 shows such a design methodology.

The Algorithm begins similarly with the application of Hungarian Algorithm. Subsequently, through  $k$  steps, we attempt to increase the number of disjoint chains to each PE in the system by 1 at each step. We similarly introduce set  $incl$ , which initially contains all the spares. Then, we form a Replacement relationship between  $p_i \in incl$  and  $p_j \notin incl$  when  $p_i$  degree is  $deg[i] < max\_deg$ . We find the minimum cost Replacement which increases the disjoint chains into  $p_j$  by 1 and the resulting  $deg[incl \cup p_j] < max\_deg$ . Note that we do not need to consider the Condition (ii) from Algorithm 1, since every PE's disjoint incoming chains increases uniformly by 1.

### 3.5.3 Experimental Results

We evaluate 4 systems with arbitrary but uniform placement of PE's and Routers in a 2D space. The systems consist of 100 functional PE's along with 10 and 20 spare PE's, and 200 functional PE's along with 20 and 40 spare PE's. We design the Balanced and Unbalanced systems following the Algorithm proposed in the previous section. For the systems, we evaluate the reliability-cost trade-offs. We also

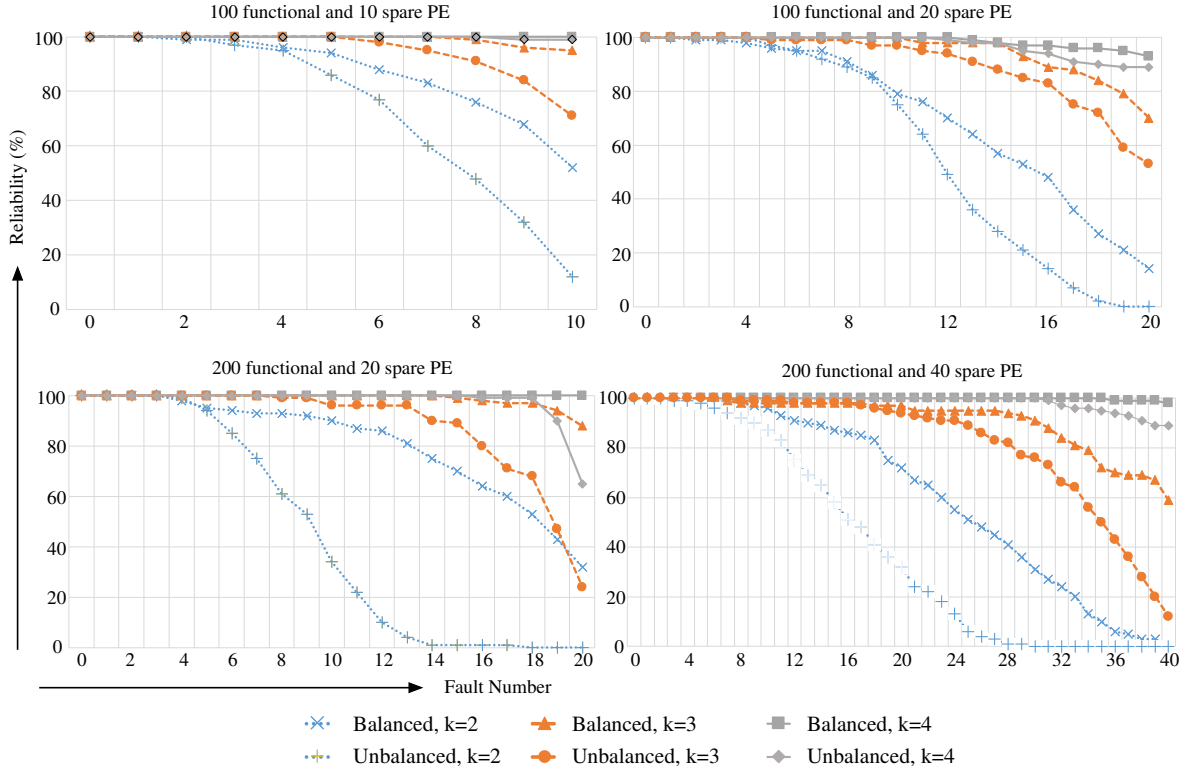


Figure 37: Reliability curves for different systems

show how the Algorithms allow the Coverages to be equally distributed among the PE's in a typical uniform system. Moreover, we also show the flexibility afforded by our proposed model in terms of repairs. All the experiments were done in C++.

### 3.5.3.1 Reliability-Cost Trade-offs

The Reliability curves, obtained via Monte-Carlo simulation for 1000 trials, for the systems are shown in Figure 37. The two important takeaways from the results are i) Balanced system has greater

reliability than Unbalanced systems as predicted, and ii) to design a system which can deliver 100% reliability for  $s$  faults in a guaranteed  $k$ -FT system ( $k \leq s$ ), we do not need to make  $s = k$ .

Firstly, Balanced systems have superior reliability compared to Unbalanced system. As an example, for the 200 PE 20 spare system with 3 faults guaranteed tolerable, Balanced system can tolerate 20 faults with  $> 80\%$  reliability, which the Unbalanced system with the same configuration can only do so around 20% of the time.

Secondly, for the systems with  $s$  spares, they have a great chance of tolerating  $s$  faults with almost 100% reliability even when the system is  $k$ -FT, when  $s \gg k$ . As an example, for the 200 PE 20 spare system, the Balanced system can guarantee around 100% reliability even when the system is made guaranteed 4-FT.

However, the discrepancies in reliabilities between the Balanced and Unbalanced systems are set straight by the reliability-cost trade-offs, shown in Table 1. The total cost, given by the total interconnect lengths in the system, is always less in Unbalanced systems, as predicted. For the data in Table 1, all the cost values have been normalized with respect to the Unbalanced system with  $k = 2$ -FT. As an example, for 200 PE 20 spare system, Balanced system with  $k = 2$ -FT is 17% costlier compared to its Unbalanced system counterpart. When  $k = 4$ , the Balanced system costs 2.15, while its Unbalanced system counterpart costs 2.05.

We have also evaluated how well the Coverages are distributed among the PE's through Mean Average Distance (MAD). MAD is given as

$$MAD = 1/(n+s) \sum_{i=1}^{n+s} |deg_{avg} - deg_i|$$

TABLE III: COST AND MEAN AVERAGE DISTANCE (MAD) FOR THE COVERAGE DISTRIBUTION CORRESPONDING TO THE SYSTEMS IN FIGURE 37

System composition		Cost						MAD					
		Balanced			Unbalanced			Balanced			Unbalanced		
		k=2	k=3	k=4	k=2	k=3	k=4	k=2	k=3	k=4	k=2	k=3	k=4
100 PE	10 spares	1.09	1.53	2.04	1	1.47	2.02	0.24	0.25	0.24	0.25	0.35	0.35
	20 spares	1.21	1.66	2.19	1	1.49	2.05	0.16	0.32	0.32	0.26	0.27	0.2
200 PE	20 spares	1.17	1.65	2.15	1	1.51	2.05	0.21	0.25	0.25	0.32	0.27	0.23
	40 spares	1.19	1.69	2.27	1	1.51	2.06	0.14	0.21	0.24	0.21	0.17	0.15

As an example, for the system with 200 PE's and 20 spares (assumed maximum allowable degree  $max\_deg = 6$ ), when  $k = 3$ , the 200 Routers require a total of  $200 \times 4 = 800$  Coverages, including Assignments. Thus, every PE must have  $800/220 = 3.64$  Coverages each. From Table 1, MAD is shown to be 0.27, which means the degrees are constrained within  $3.64 + 0.27 = 3.91$  and  $3.64 - 0.27 = 3.37$ , indicating how well distributed the Coverages are among the PE's. In general, all the MAD values are shown as  $< 1$ , indicating well distributed Coverages among PE's.

### 3.5.3.2 Replacement Chain Selection

We proved that to repair a Router-PE Based system, any arbitrary chain be be used, without affecting the future fault tolerance possibilities in the system. Figure 38 shows how we can use that to our advantage. In this plot, we evaluate the communication overhead between a Router and its Assigned

PE, depending upon how far they are located from each other. Since the communication overhead, or *delay* is proportional to the square of the interconnect length, we consider the total delay of the system to be

$$d = \sum_{i=1}^n (|x_i - x_j| + |y_i - y_j|)^2$$

where  $A_{i,j} = 1$  and  $(x_i, y_i)$  is the coordinates of the Router  $r_i$ , and  $(x_j, y_j)$  is the coordinates of the Assigned PE  $p_j$ .

The initial delay in the system is the smallest, we have employed the Least Cost Assignment (Hungarian) Algorithm, the subsequent delays following every fault must increase. However, how much they increase can be controlled by considering which chain to choose for repair. In Figure 38, it can be observed that, when we use any arbitrary Replacement Chain for 200 PE 40 spare system, the delay after the 40<sup>th</sup> fault is as much as 2.2 times that of the initial delay. However, if the least cost chain is used, such delay increase can be only 1.4 times that of the initial delay. Thus, the freedom in choosing the Replacement Chain can have a huge impact in the system performance. The system can choose the chain which is the most beneficial for the it.

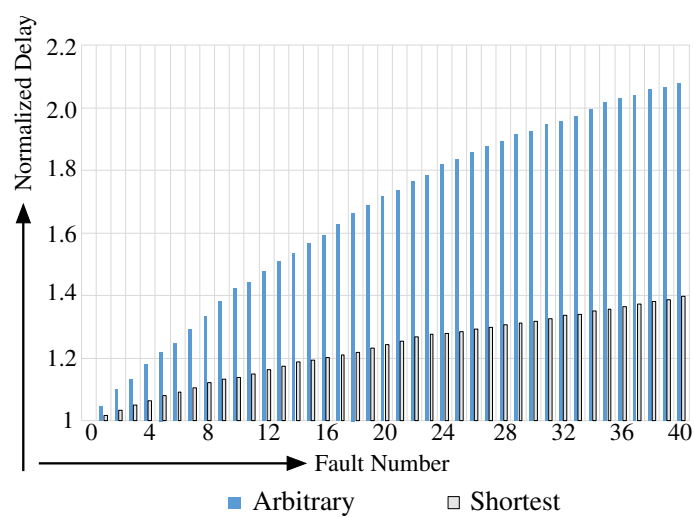


Figure 38: System delay increase for shortest vs random chains

## CHAPTER 4

### CONCLUSIONS AND FUTURE WORKS

The dissertation has proposed two novel methods of generalizing circuits and systems design with an emphasis of reliable designs. The first method deals with generalization of desired properties of an existing circuit and extension of the same to other similar circuits, while the second method formalizes certain basic properties and implements it on any arbitrary systems.

The first method generalizes the Group Segregation property of Kogge-Stone Adders (KSA) and exploits it for defect tolerant and sparse Parallel Prefix Adder (PPA) designs. For the defect tolerant PPA designs, the approach is suitable for post-manufacturing reconfiguration for defect tolerance. The sparse PPA's offer greater number of adder choices in the design space to optimize the overall design in terms of power-performance-area.

The second method applies on Many-Processor systems consisting of Processing Elements (PE). The Replacement Chain Algebra is formalized on the proposed Task-PE model. As shown, the Task-PE model is suitable for implementing any arbitrary Many-Processor system through the Router-PE network. Combined with the Replacement Chain Algebra, it is able to deliver simple lightweight repairs on-the-fly after every fault manifestation without having to stop the system. Furthermore, with additional constraints, it is able to deliver superior reliability, while maintaining scalability in terms of growing number of components as well as reliability requirements in the form of strictly local interconnects.

#### 4.1 Generalization of Group Segregation Property in Kogge-Stone Adder

For the PPA part, we have shown how the bits in a KSA get divided into two disjoint groups: even and odd, through the Group Segregation property. The two groups never interact with each other after the first level. In the following levels, the computations are done independently for each group. In addition, a group can compute its final results by using the results of the other group with a small hardware and timing overhead. This property can be used in two ways: 1. compute a redundant copy of every bit for defect tolerance purposes; and 2. remove the hardware for a group and make it compute its result using the results of the other group, to convert it into its sparse version.

We have shown that the redundant copy of the results can be used for defect tolerance purposes. The Group-Segregation property can be extended onto other adders as well. On the other hand, additional redundant copies of the results can be obtained by increasing the number of groups. With increased number of groups, the adder is able to tolerate more number of faults.

On the other hand, the hardware for one group can be entirely removed and the other group can compute the results of the former by using an additional level of computation. From a typical KSA, this results in creation of a 2-sparse KSA, which incidentally is a Han-Carlson Adder (HCA). We have shown that this idea can again be extended to other PPA's. Moreover, when the number of groups is increased similarly, we are able to obtain adders that are more sparse.

Overall, the Group-Segregation property, pointed out by the dissertation helps adder-designs in two ways: defect tolerance and sparsity in PPA. The contribution of this part is the extension of the highly desirable Group-Segregation property from KSA to any arbitrary PPA, so that any arbitrary PPA can utilize the advantage of defect tolerance and sparsity. As shown before, previous approaches for ex-

exploiting defect tolerance and sparsity in PPA's were ad-hoc: to select the most suitable adder design, the designers needed to consider all the ad-hoc approaches and select the suitable one. In comparison, our proposed approach allows the designers to use one single approach to come up with various design, thus making it easier to compare different designs.

A **Cookie-cutter** cuts the cookies in identical shapes, even though the ingredients of the dough can be different. Similarly, in this case, all the defect tolerant and sparse adders look identically shaped, when the “ingredients”, or components, can be different (type of adder, number of groups).

#### 4.2 Fault Tolerant Many-Processor Systems

We have proposed the 2-layered Task-PE model, utilizing the Replacement Chain Algebra, which we have shown to facilitate the design of fault tolerant Many-Processor systems. The proposed model is able to handle any arbitrary Many-Processor system. On the other hand, Replacement Chain Algebra enables the repairs to be simple, lightweight on-the-fly. Given certain conditions, the system can predict how many fault are guaranteed tolerable. As we have mentioned in the Introduction, the Processing Elements (PE's) are more likely to get faulty one at a time, requiring one fault tolerance after every fault manifestation. This assumption is significantly different from the other works, which assume the knowledge about  $k$  faults, and go about finding repair solutions for all of them simultaneously. To achieve the fault tolerance after every fault manifestation, we have shown that the Router-PE model supports lightweight on-the-fly self-repairs. As a result, the normal operation of the system is not significantly affected, and the repairs are kept hidden from the user.

Using the proposed model, we showed how a scalable Processor Array can be designed with strictly local interconnects through a Router-PE network. Such arrays are able to deliver nearly 100% reliability

with bigger systems delivering even better reliability. The results indicate that to tolerate  $s$  faults with significant certainty, it is not necessary for the system to be guaranteed  $s$ -FT. In fact, a  $k$ -FT system will deliver a performance very close to the target level, even when  $k \ll s$ .

The main essence of our proposed model is its flexibility in terms of design as well as the repair methodology. While most other works were ad-hoc, i.e., only applicable for specific Processor topology or physical placement, the proposed model is able to handle any Processor topology and placement. Thus, it provides an unified approach for designing a large range of systems. On the other hand, the repairs are possible via any available replacement chain on-the-fly after every fault occurrence, without having to stop the regular functionality of the system.

Two different approaches are proposed to design two different types of systems: one emphasizing on reliability when the number of faults is beyond the guaranteed tolerable level, and another one emphasizing the total and individual lengths of the required interconnects.

Overall, we have proposed a model, which can do the following:

- i) Repair faults in a lightweight fashion after every fault occurrence, on-the-fly.
- ii) System can guarantee  $k$  fault tolerances, provided it meets certain requirements.
- iii) Is applicable to any arbitrary Processor topology or placement: a case study for 2D Array is shown.

When compared with a **Cookie-cutter**, while Cookie-cutter cuts the dough in the forms of identical looking cookies of different compositions, our approach relies on identical Task-PE model, designs the systems of different configurations.

While we have proposed to combine the PE fault tolerant approach with fault tolerance in Routers and Interconnects, an overall design evaluation can be considered as a **future work**. Such a system will be able to tolerate faults at PE, Routers and even at the interconnect level, providing an overall fault tolerant design approach with the repair strategies.

The proposed approach is at a theoretical level. Another **future work** will be to implement such a system on an existing or new platform (Network-on-Chip, FPGA etc). Implementation of the proposed approach on these platforms will give a clearer picture about the usefulness of the design. In the real life implementation, certain other issues will come to light. One such issue will be data migration from one cache to another in case of a fault. If the system is designed by assuming independent L-1 cache for every PE, when one PE replaces another the contents of the L-1 cache of the later must be transferred to the former. Appropriate hardware overhead must be considered to support the data migration. On the other hand, if there is shared L-2 caches, and if a PE replaces another PE, yet they access different L-2 caches, data migration must also take place between L-2 caches as well.

## **APPENDIX**

### **COPYRIGHT PERMISSIONS**

In this section, the reuse permissions for the papers used in this dissertation are presented. The list of the articles include Design Automation and Test in Europe (DATE) 2016 (Banerjee and Rao, 2016), IEEE Computer Society Annual Symposium on VLSI (ISVLSI) 2017 (Banerjee and Rao, 2017a), IEEE International Conference on Computer Design (ICCD) 2015 (Banerjee and Rao, 2015) and Asia-South Pacific Design Automation Conference (ASP-DAC) 2017 (Banerjee and Rao, 2017b). The permissions follow in the same order. Permission for (Banerjee and Rao, 2016) was obtained by the following letter written to the copyright holder European Design and Automation Association (EDAA). The permissions for the rest were obtained from the copyright holder IEEE.

Regarding Reuse Permission For Paper “A General Approach for Highly Defect Tolerant Parallel Prefix Adder Design” Published in DATE 2016 Proceedings

To Whom It May Concern,

I, Soumya Banerjee, require a Reuse Permission, for use in my PhD dissertation, for the following paper:

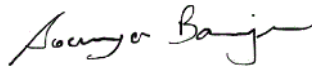
Title: A general approach for highly defect tolerant parallel prefix adder design

Co-authored by: Soumya Banerjee (ECE Department, University of Illinois at Chicago, IL, USA) and by Wenjing Rao (ECE Department, University of Illinois at Chicago, IL, USA)

Published in: Design, Automation & Test in Europe Conference & Exhibition (DATE), 2016 Proceedings, 14-18 March 2016, Dresden, Germany, ISBN: 978-3-9815-3707-9 and ISSN: 1558-1101.

I hereby declare that I will properly cite the contribution of the aforementioned paper in my PhD dissertation at the University of Illinois at Chicago, IL, USA, titled: Cookie-Cutter: Achieving Defect/Fault Tolerance For Large-Scale Systems with Highly Unreliable Components.

Sincerely,



7/26/2017

Soumya Banerjee

Co-author:



Wenjing Rao

## APPENDIX (Continued)



RightsLink®

Home

Create Account

Help



**Title:** A General Design Framework for Sparse Parallel Prefix Adders

**Conference Proceedings:** 2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)

**Author:** Soumya Banerjee; Wenjing Rao

**Publisher:** IEEE

**Date:** 3-5 July 2017

Copyright © 2017, IEEE



**LOGIN**

If you're a [copyright.com user](#), you can login to RightsLink using your copyright.com credentials. Already a [RightsLink user](#) or want to [learn more?](#)


## Thesis / Dissertation Reuse

**The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:**

*Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:*

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line  2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line  [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

*Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:*

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references:  [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to [http://www.ieee.org/publications\\_standards/publications/rights/rights\\_link.html](http://www.ieee.org/publications_standards/publications/rights/rights_link.html) to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK

CLOSE WINDOW

## APPENDIX (Continued)



RightsLink®

Home

Create Account

Help



**Title:** On the conditions of guaranteed k-fault tolerant systems supporting on-the-fly repairs

**Conference Proceedings:** 2015 33rd IEEE International Conference on Computer Design (ICCD)

**Author:** Soumya Banerjee; Wenjing Rao

**Publisher:** IEEE

**Date:** 18-21 Oct. 2015

Copyright © 2015, IEEE

**LOGIN**

If you're a [copyright.com user](#), you can login to RightsLink using your copyright.com credentials. Already a [RightsLink user](#) or want to [learn more?](#)

## Thesis / Dissertation Reuse

**The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:**

*Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:*

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line ♦ 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line ♦ [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

*Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:*

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: ♦ [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to [http://www.ieee.org/publications\\_standards/publications/rights/rights\\_link.html](http://www.ieee.org/publications_standards/publications/rights/rights_link.html) to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK

CLOSE WINDOW

## APPENDIX (Continued)



RightsLink®

Home

Create Account

Help



**Title:** A local reconfiguration based scalable fault tolerant many-processor array

**Conference Proceedings:** 2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)

**Author:** Soumya Banerjee; Wenjing Rao

**Publisher:** IEEE

**Date:** 16-19 Jan. 2017

Copyright © 2017, IEEE

**LOGIN**

If you're a **copyright.com** user, you can login to RightsLink using your copyright.com credentials. Already a **RightsLink** user or want to [learn more?](#)

## Thesis / Dissertation Reuse

**The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:**

*Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:*

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line ♦ 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line ♦ [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

*Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:*

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: ♦ [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to [http://www.ieee.org/publications\\_standards/publications/rights/rights\\_link.html](http://www.ieee.org/publications_standards/publications/rights/rights_link.html) to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK

CLOSE WINDOW

## CITED LITERATURE

- [Abraham et al. , 1987] Abraham, J., Banerjee, P., Chen, C.-Y., Fuchs, W., Kuo, S.-Y., and Reddy, A. L. N.: Fault tolerance techniques for systolic arrays. IEEE Computer, 20 Issue 7:65–75, 1987.
- [Ajtai et al. , 1992] Ajtai, M., Alon, N., Bruck, J., Cypher, R., Ho, C. T., Naor, M., and Szemerédi, E.: Fault tolerant graphs, perfect hash functions and disjoint paths. In In Proceedings of the 33rd Annual Symposium on Foundations of Computer Science, pages 693–702, 1992.
- [Aktan et al. , 2015] Aktan, M., Baran, D., and Oklobdzija, V.: Minimizing Energy by Achieving Optimal Sparseness in Parallel Adders. In Computer Arithmetic (ARITH), IEEE 22nd Symposium on, pages 10–17, 2015.
- [Banerjee and Rao, 2015] Banerjee, S. and Rao, W.: On the conditions of guaranteed k-fault tolerant systems supporting on-the-fly repairs. In IEEE International Conference on Computer Design, pages 387–390, 2015.
- [Banerjee and Rao, 2016] Banerjee, S. and Rao, W.: A general approach for highly defect tolerant parallel prefix adder design. In Design, Automation and Test in Europe Conference and Exhibition, pages 666–671, 2016.
- [Banerjee and Rao, 2017a] Banerjee, S. and Rao, W.: A general design framework for sparse parallel prefix adders. In IEEE Computer Society Annual Symposium on VLSI, pages 231–236, 2017.
- [Banerjee and Rao, 2017b] Banerjee, S. and Rao, W.: A local reconfiguration based scalable many-processor system. In Asia and South Pacific Design Automation Conference, pages 432–437, 2017.
- [Bedrij, 1962] Bedrij, O.: Carry-Select Adder. IRE Transactions on Electronic Computers, EC-11(0):340–346, June 1962.
- [Bokhari, 1981] Bokhari, S.: On the mapping problem. In IEEE Transactions on Computers, volume C-30, pages 207–214, March 1981.
- [Borkar, 2007] Borkar, S.: Thousand core chips: A technology perspective. In DAC '07, Proceedings of the 44th annual Design Automation Conference, pages 746–749, 2007.

- [Bruck et al. , 1993] Bruck, J., Cypher, R., and Ho, C.-T.: Fault tolerant meshes with small degree. In Proc. of ACM Symposium on Parallel Architectures and Algorithms, pages 1–10, 1993.
- [Bruck et al. , 1992] Bruck, J., Cypher, R., and Ho, C.: Efficient fault-tolerant mesh and hypercube architectures. In Fault-Tolerant Computing, 1992. FTCS-22. Digest of Papers., Twenty-Second International Symposium on, pages 162–169, 1992.
- [Chen et al. , 1995] Chen, S., Eshaghian, M. M., and Wu, Y.-C.: Mapping arbitrary non-uniform task graphs onto arbitrary non-uniform system graphs. In ICCP (2), 1995.
- [Chen and Upadhyaya, 1993] Chen, Y. and Upadhyaya, S.: Reliability, reconfiguration, and spare allocation issues in binary-tree architectures based on multiple-level redundancy. Computers, IEEE Transactions on, 42, Issue 6:713–723, June 1993.
- [Constantinescu, 2003] Constantinescu, C.: Trends and challenges in vlsi circuit reliability. IEEE Micro, 23, Issue 4:14–19, July 2003.
- [Dutt and Hayes, 1990] Dutt, S. and Hayes, J.: On designing and reconfiguring k-fault tolerant tree architectures. Computers, IEEE Transactions on, 39, Issue 4:490–503, 1990.
- [Dutt and Hayes, 1992] Dutt, S. and Hayes, J.: Some practical issues in the design of fault-tolerant multiprocessors. In IEEE Trans. Comput., Special Issue on Fault-Tolerant Computing, volume 41, pages 588–598, 1992.
- [Dutt and Mahapatra, 1997] Dutt, S. and Mahapatra, N.: Node-covering, error-correcting codes and multiprocessors with very high average defect tolerance. In Computers, IEEE Transactions on, volume 46, Issue 9, pages 997–1015, 1997.
- [Fukushi and Horiguchi, 2004] Fukushi, M. and Horiguchi, S.: A self-reconfigurable hardware architecture for mesh arrays using single/double vertical track switches. Instrumentation and Measurement, IEEE Transactions on, 53, Issue 2:357–367, April 2004.
- [G. Jiang, 2015] G. Jiang, J. Wu, Y. H. Y. W. J. S.: Reconfiguring three-dimensional processor arrays for fault tolerance: Hardness and heuristic algorithms. In IEEE Transactions on Computers, volume 64, pages 2926–2939, Oct. 2015.
- [Geist, 2016] Geist, A.: Supercomputing’s monster in the closet. IEEE Spectrum, 53, Issue 3:30–35, March 2016.

- [Ghiribaldi et al. , 2011] Ghiribaldi, A., Ludovici, D., and Bertozzi, D.: System-level infrastructure for boot-time testing and configuration of networks-on-chip with programmable routing logic. In VLSI and System-on-Chip (VLSI-SoC), 2011 IEEE/IFIP 19th International Conference on, pages 308–313, 2011.
- [Ghosh et al. , 2007] Ghosh, S., Ndai, P., Bhunia, S., and Roy, K.: Tolerance to Small Delay Defects by Adaptive Clock Stretching. On-Line Testing Symposium, 2007. IOLTS 07, 13th IEEE International, pages 244–252, July 2007.
- [Ghosh et al. , 2008] Ghosh, S., Ndai, P., and Roy, K.: A Novel Low Overhead Fault Tolerant Kogge-Stone Adder Using Adaptive Clocking. Design, Automation and Test in Europe, 2008. DATE '08, pages 366–371, March 2008.
- [Gurkayna et al. , 2000] Gurkayna, F., Leblebici, Y., Chaouati, L., and McGuinness, P.: Higher radix kogge-stone parallel prefix adder architectures. IEEE International Symposium on Circuits and Systems, 5:609–612, 2000.
- [Hanchek and Dutt, 1998] Hanchek, F. and Dutt, S.: Methodologies for tolerating cell and interconnect faults in fpgas. In IEEE Transactions on Computers, volume 47, Issue 1, pages 15–33, 1998.
- [Hatori et al. , 1993] Hatori, F., Sakurai, T., Nogami, K., Sawada, K., Takahashi, M., Ichida, M., Uchida, M., Yoshii, I., Kawahara, Y., Hibi, T., Saeki, Y., Muroga, H., Tanaka, A., and Kanzaki, K.: Introducing redundancy in field programmable gate arrays. In Custom Integrated Circuits Conference, 1993 Proceedings of the IEEE, pages 7.1.1–7.1.4, 1993.
- [ITRS, 2015] ITRS: International technology roadmap for semiconductors, emerging research devices. 2015.
- [Izadi and Ozguner, 2003] Izadi, B. and Ozguner, F.: Enhanced cluster k-ary n-cube, a fault-tolerant multi-processor. Computers, IEEE Transactions on, 52, Issue 11:1443–1453, Nov 2003.
- [Jigang et al. , 2007] Jigang, W., Srikanthan, T., and Wang, X.: Integrated row and column rerouting for reconfiguration of vlsi arrays with four-port switches. Computers, IEEE Transactions on, 56, Issue 10:1387–1400, Oct 2007.
- [Johnson, 1989] Johnson, B.: Design and Analysis of Fault Tolerant Digital Systems. Addison Wesley Publishing Company, 1989.
- [Kahng and Kang, 2012] Kahng, A. and Kang, S.: Accuracy-Configurable Adder for Approximate Arithmetic Designs. Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE, pages 820–825, June 2012.

- [Khaleghi and Rao, 2013] Khaleghi, S. and Rao, W.: Spare sharing network enhancement for scalable systems. In Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2013 IEEE International Symposium on, pages 249–254, 2013.
- [Kuhn, 1955] Kuhn, H.: The hungarian method for the assignment problem. Naval Research Logistics Quarterly, 2:83–97, 1955.
- [Li et al. , 1992] Li, C., McCormick, S., and Simchi-Levi, D.: Finding disjoint paths with different pathcosts: Complexity and algorithms. Networks, 22.7:653–667, 1992.
- [Lienig, 2013] Lienig, J.: Electromigration and its impact on physical design in future technologies. In Proceedings of the ACM International Symposium on Physical Design, pages 33–40, 2013.
- [Liu et al. , 2007] Liu, J., Zhu, Y., Zhu, H., Cheng, C., and Lillis, J.: Optimum prefix adders in a comprehensive area, timing and power design space. In Asia and South Pacific Design Automation Conference, pages 609–615, 2007.
- [Mahapatra and Dutt, 2001] Mahapatra, N. and Dutt, S.: Hardware-efficient and highly reconfigurable 4- and 2-track fault-tolerant designs for mesh connected arrays. Journal of Parallel and Distributed Computing, 61, Issue 10:1391–1411, October 2001.
- [Matthew et al. , 2003] Matthew, S., Anders, M., Krishnamurthy, R., and Borkar, S.: A 4-GHz 130-nm Address Generation Unit with 32-bit Sparse-Tree Adder Core. IEEE Journal of Solid-State Circuits, 38(5):689–695, May 2003.
- [Modarresi et al. , 2010] Modarresi, M., Sarbazi-Azad, H., and Tavakkol, A.: An efficient dynamically reconfigurable on-chip network architecture. In Proc. of 47th Design Automation Conference (DAC), pages 310–313, 2010.
- [Murali and Micheli, 2004] Murali, S. and Micheli, G. D.: Sunmap: A tool for automatic topology selection and generation for noc's. In Proc. of 41st Design Automation Conference (DAC), pages 914–919, 2004.
- [Ndai et al. , 2007] Ndai, P., Lu, S.-L., Somesekhar, D., and Roy, K.: Fine Grained Redundancy in Adders. Quality Electronic Design, 2007. ISQED '07. 8th International Symposium on, pages 317–321, March 2007.
- [Parhami, 2010] Parhami, B.: Computer Arithmetic Algorithms and Hardware Designs. Oxford University Press, New York, 2nd edition, 2010.

- [Ren et al. , 2013a] Ren, Y., Liu, L., Yin, S., Han, J., Wu, Q., and Wei, S.: A fault tolerant noc architecture using quad-spare mesh topology and dynamic reconfiguration. EUROMICRO Journal of Systems Architecture, 59 Issue 7:482–491, 2013.
- [Ren et al. , 2013b] Ren, Y., Liu, L., Yin, S., Wu, Q., Wei, S., and Han, J.: A vlsi architecture for enhancing the fault tolerance of noc using quad-spare mesh topology and dynamic reconfiguration. In IEEE International Symposium on Circuits and Systems, pages 1793–1796, 2013.
- [Roy et al. , 2014] Roy, S., Choudhury, M., Puri, R., and Pan, D.: Towards optimal performance-area trade-off in adders by synthesis of parallel prefix structures. IEEE Transactions on Computer-Aided Design of Intergrated Circuits and Systems, 33(10):1517–1530, 2014.
- [Roychowdhury et al. , 1990] Roychowdhury, V., Bruck, J., and Kailath, T.: Efficient algorithms for reconfiguration in vlsi/wsi arrays. Computers, IEEE Transactions on, 39, Issue 4:480–489, 1990.
- [Sha and Steiglitz, 1993] Sha, E. H.-M. and Steiglitz, K.: Reconfigurability and reliability of systolic/wavefront arrays. In IEEE Transactions on Computers, volume 42, pages 854–862, 1993.
- [Shi and Fuchs, 1992] Shi, W. and Fuchs, W.: Probabilistic analysis and algorithms for reconfiguration of memory arrays. In IEEE Trans. on Computer Aided Design, volume 11, no. 9, pages 1153–1160, Sept 1992.
- [Singh and Youssef, 1995] Singh, H. and Youssef, A.: Mapping and scheduling heterogeneous task graphs using genetic algorithms. In MS Thesis, George Washington University, 1995.
- [Sklansky, 1960] Sklansky, J.: Conditional-Sum Addition Logic. IRE Transactions on Electronic Computers, EC-9(2):226–231, June 1960.
- [Taura and Chien, 2000] Taura, K. and Chien, A.: A heuristic algorithm for mapping communicating tasks on heterogeneous resources. In Heterogeneous Computing Workshop, 2000. (HCW 2000) Proceedings, 9th IEEE, pages 120–115, 2000.
- [Townsend et al. , 2003] Townsend, W., Abraham, J., and Swartzlander, E.: Quadruple Time Redundancy Adders. Defect and Fault Tolerance in VLSI Systems, 2003. Proceedings. 18th IEEE International Symposium on, pages 250–256, November 2003.
- [Varvarigou et al. , 1993] Varvarigou, T., Roychowdhury, V., and Kailath, T.: Reconfiguring processor arrays using multiple-track models: The 3-track-1-spare-approach. In IEEE Transactions on Computers, volume 42, Issue 11, pages 1281–1293, 1993.

- [Wachter et al. , 2013] Wachter, E., Erichsen, A., Amory, A., and Moraes, F.: Topology-agnostic fault-tolerant noc routing method. In Design, Automation and Test in Europe Conference and Exhibition, pages 1595–1600, 2013.
- [Ye et al. , 2013] Ye, R., Wang, T., Yuan, F., Kumar, R., and Xu, Q.: On reconfiguration-oriented approximate adder design and its application. In Computer-Aided Design (ICCAD), 2013 IEEE/ACM International Conference on, pages 48–54, 2013.
- [Zeydel et al. , 2010] Zeydel, B., Baran, D., and Oklobdzija, V.: Energy-Efficient Design Methodologies: High-Performance VLSI Adders. IEEE Journal of Solid-State Circuits, 45(6):1220–1233, June 2010.
- [Zhang, 2002] Zhang, L.: Fault-tolerant meshes with small degree. In IEEE Transactions on Computers, volume 51, pages 553–560, May 2002.
- [Zhang et al. , 2009] Zhang, L., Han, Y., Xu, Q., and Li, X.: On topology reconfiguration for defect-tolerant noc-based homogeneous manycore systems. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, 17, Issue 9:1173–1186, 2009.
- [Zhu et al. , 2007] Zhu, H., Pande, P., and Grecu, C.: Performance evaluation of adaptive routing algorithms for achieving fault tolerance in noc fabrics. In IEEE International Conference on Application-specific Systems, Architectures and Processors, pages 42–47, 2007.
- [Ziegler, 1996] Ziegler, J.: Terrestrial cosmic rays. IBM Journal of Research and Development, 40, Issue 1:19–39, Jan 1996.
- [Zlatanovici et al. , 2009] Zlatanovici, R., Kao, S., and Nikolic, B.: Energy–Delay Optimization of 64-Bit Carry-Lookahead Adders With a 240 ps 90 nm CMOS Design Example. IEEE Journal of Solid-State Circuits, 44, Issue 2:569–583, Feb 2009.

## VITA

# Soumya Banerjee

sbaner8@uic.edu

<http://www.ece.uic.edu/~sbanerje/>

### Education

- **PhD** in Computer Engineering, University of Illinois at Chicago (UIC), 2017
- **MS** in Computer Engineering, UIC, 2016
- **BTech** in Electronics and Communication Engineering, West Bengal University of Technology (currently Maulana Abul Kalam Azad University of Technology), Kolkata, India, 2011

### Research Experience

As a Research Assistant in the Electrical and Computer Engineering (ECE) Department at UIC from August 2011 - May 2017:

- **Defect and Fault Tolerance in Nano-scale Systems:** Evaluation of novel defect and fault tolerance methodologies in error prone size-scaled circuits and systems via hardware and/or timing overhead. Systems under considerations include Parallel Prefix Adders and Many-Processor systems.
- **Fair Resource Distribution among Agents in Constrained Access Systems:** Evaluation of solution space and algorithms for fairly distributing Resources among Agents where the access of the Resources from the Agents are constrained.

### Teaching Related Experience

As a Teaching Assistant in ECE Department at UIC from May 2012 - May 2017:

- **Computer Organization:** Assembly Language Programming of MIPS Processors.
- **CAD Based Digital Design:** VHDL implementation of Digital Circuits.
- **Introduction to Logic Design:** Logic design using TTL gates on breadboards.
- **Analog and Mixed Signal VLSI Design:** Analog circuit design and simulation in HSPICE.
- **Introduction to VLSI Design:** CMOS circuit and layout design for digital VLSI circuits.

### Industry Experience

Bluetooth IC Design Intern at Broadcom Corp., San Diego, CA, May 2013 - August 2013.

**Publications**

- S. Banerjee and W. Rao, “A General Design Framework for Sparse Parallel Prefix Adders”, *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2017, Pages 231-236.
- J. Xu, S. Banerjee and W. Rao, “The Existence of Universally Agreed Fairest Semi-matchings in Any Given Bipartite Graph”, *Computing and Combinatorics. COCOON 2017, Springer Lecture Notes in Computer Science*, vol 10392, Pages 529-541.
- S. Banerjee and W. Rao, A Local Reconfiguration Based Scalable Fault Tolerant Many-Processor Array, *Asia and South Pacific Design Automation Conference (ASPDAC)* 2017, Pages 432-437.
- S. Khaleghi, P. Vinella, S. Banerjee and W. Rao, An STT-MRAM Based Strong PUF, *IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)* 2016, Pages 129-134.
- S. Banerjee and W. Rao, A General Approach for Highly Defect Tolerant Parallel Prefix Adder Design, *Design, Automation and Test in Europe (DATE)* 2016, Pages 666-671.
- S. Banerjee and W. Rao, On the Conditions of Guaranteed k-Fault Tolerant Systems Supporting On-The-Fly Repairs, *IEEE International Conference on Computer Design (ICCD)* 2015, Pages 387-390.
- S. Banerjee, K. Zhao, W. Rao and M. Zefran, Decentralized Self-Balancing Systems, *IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)* 2015, Pages 340-343.