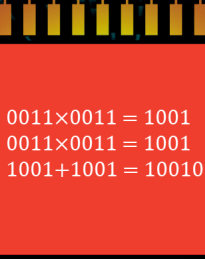
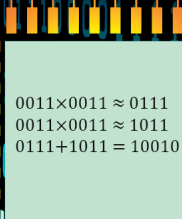


EXPLOITING ERROR RESILIENCE FOR HARDWARE EFFICIENCY

TARGETING ITERATIVE AND ACCUMULATION BASED ALGORITHMS



$0011 \times 0011 = 1001$
 $0011 \times 0011 = 1001$
 $1001 + 1001 = 10010$



$0011 \times 0011 \approx 0111$
 $0011 \times 0011 \approx 1011$
 $0111 + 1011 = 10010$

G.A. Gillani

Exploiting Error Resilience For Hardware Efficiency

*Targeting Iterative and
Accumulation Based Algorithms*

G.A. Gillani

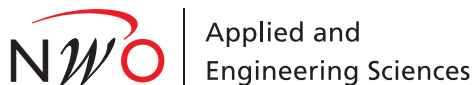
Members of the graduation committee:

Dr. ir. A. B. J. Kokkeler	University of Twente (promotor)
Dr. ing. D. M. Ziener	University of Twente
Prof. dr. ir. G. J. M. Smit	University of Twente
Prof. dr. ir. D. Stroobandt	Ghent University
Prof. dr. H. Corporaal	Eindhoven University of Technology
Prof. dr. J. Nurmi	Tampere University
Dr. ir. A. J. Boonstra	Astron (special expert)
Prof. dr. J. N. Kok	University of Twente (chairman and secretary)

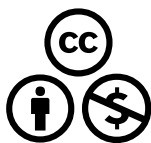
UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering, Mathematics and Computer Science, Computer Architecture for Embedded Systems (CAES) group.

DSI Ph.D. Thesis Series No. 20-004
Digital Society Institute
PO Box 217, 7500 AE Enschede, The Netherlands.



This work was supported in part by the Netherlands Institute of Radio Astronomy (ASTRON) and IBM Joint Project, DOME, funded by the Netherlands Organization for Scientific Research (NWO), in part by the Dutch Ministry of Economic Affairs, Agriculture and Innovation (EL&I), and in part by the Province of Drenthe.



Copyright © 2020 G.A. Gillani, Enschede, The Netherlands. This work is licensed under the Creative Commons Attribution-NonCommercial 4.0 International License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc/4.0/deed.en_US.



This thesis was typeset using \LaTeX and *TikZ*. This thesis was printed by Gildeprint Drukkerijen, The Netherlands.

ISBN 978-90-365-5011-6
ISSN 2589-7721; DSI Ph.D. Thesis Series No. 20-004
DOI 10.3990/1.9789036550116

EXPLOITING ERROR RESILIENCE FOR HARDWARE
EFFICIENCY

TARGETING ITERATIVE AND ACCUMULATION BASED ALGORITHMS

DISSERTATION

to obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
prof. dr. T. T. M. Palstra,
on account of the decision of the Doctorate Board,
to be publicly defended
on Friday the 3rd of July 2020 at 14 : 45 hours

by

Syed Ghayoor Abbas Gillani

This dissertation has been approved by:

Dr. ir. A. B. J. Kokkeler (promotor)

Copyright © 2020 G.A. Gillani
ISBN 978-90-365-5011-6

To Zahra, Sarah, Sakeena, ...

*One of the best objectives
of life is to seek knowledge,
absorb it, and disseminate it.*

ABSTRACT

Computing devices have been constantly challenged by resource-hungry applications such as scientific computing. These applications demand high hardware efficiency and thus pose a challenge to reduce energy/power consumption, latency, and chip-area to process a required task. Therefore, an increase in hardware efficiency is one of the major goals to innovate computing devices. On the other hand, improvements in process technology have played an important role to tackle such challenges by increasing the performance and transistor density of integrated circuits while keeping their power density constant. In the last couple of decades, however, the efficiency gains due to process technology improvements are reaching the fundamental limits of computing. For instance, the power density is not scaling as well as compared to the transistor density. Hence, posing a further challenge to control the power-/thermal-budget of the integrated circuits.

Keeping in view that many applications/algorithms are error-resilient, emerging paradigms like approximate computing come to rescue by offering promising efficiency gains especially in terms of power-efficiency. An application/algorithm can be regarded as *error-resilient* or *error-tolerant* when it provides an outcome with a required accuracy while utilizing processing-components that do not always compute accurately. There can be multiple reasons that an algorithm is tolerant of errors, for instance, an algorithm may have noisy or redundant inputs and/or a range of acceptable outcomes. Examples of such applications are machine learning, scientific computing, and search engines.

Approximate computing techniques exploit the intrinsic error tolerance of such applications to optimize the computing systems at software-, architecture- and circuit-level to achieve efficiency gains. However, the state-of-the-art approximate computing methodologies do not sufficiently address the accelerator designs for iterative and accumulation based algorithms. Taking into account a wide range of such algorithms in digital signal processing, this thesis investigates approximation methodologies to achieve high-efficiency accelerator architectures for iterative and accumulation based algorithms.

Error resilience analysis tools assess an algorithm to determine if it is a promising candidate for approximate computing. Statistical approximation (error) models are applied to an algorithm to quantify the intrinsic error resilience and to identify promising approximate computing techniques. In the context of iterative algorithms, we demonstrate that the state-of-the-art statistical model is not

effective in revealing opportunities for approximate computing. We propose an *adaptive* statistical approximation model, which provides a way to quantify the number of iterations that can be processed using an approximate core while complying with the quality constraints.

Moreover, Iterative algorithms generally apply a convergence criterion to indicate an acceptable solution. The convergence criterion is a precision-based quality function that provides a guarantee that the solution is precise enough to terminate the iterative computations. We demonstrate, however, that the precision-based quality function (the convergence criterion) is not necessarily sufficient in the error resilience analysis of iterative algorithms. Therefore, an additional accuracy-based quality function has to be defined to assess the viability of the approximate computing techniques.

Targeting energy efficiency, we further propose an accelerator design for iterative algorithms. Our design is based on a heterogeneous architecture, where heterogeneity is introduced by employing a combination of accurate and approximate cores. Our proposed methodology exploits the intrinsic error resilience of an iterative algorithm, wherein a number of initial iterations are run on the approximate core and the rest on the accurate core to achieve a reduction in energy consumption. Our proposed accelerator design does not increase the number of iterations (that are necessary in the conventional accurate counterpart) and provides sufficient precision to converge to an acceptable solution.

The conventional approximate designs follow error-restricted techniques. These techniques restrict the approximations based on the error magnitudes and the error rates they introduce to avoid an unbearable quality loss during processing. On the other hand, however, the error-restricted techniques limit the hardware efficiency benefits that can be exploited within error-resilient applications. In the context of accumulation based algorithms, we propose a Self-Healing (SH) methodology for designing approximate accelerators like square-accumulate (SAC), wherein the approximations are not restricted by error metrics but are provided with an opportunity to cancel out the errors within the processing units. SAC refers to a hardware accelerator that computes an inner product of a vector with itself, wherein the squares of the elements of a vector are accumulated.

We employ the SH methodology, in which the squarer is regarded as an approximation stage and the accumulator as a healing stage. We propose to deploy an approximate squarer *mirror pair*, such that the error introduced by one approximate squarer *mirrors* the error introduced by the other, i.e., the errors generated by the approximate squarers are approximately additive inverse of each other. This helps the healing stage (accumulator) to automatically cancel out the error originated in the approximation stage, and thereby to minimize the quality loss. Our quality-efficiency analysis of an approximate SAC shows that the proposed SH methodology provides a more effective trade-off as compared to the conventional error-restricted techniques.

Nonetheless, the proposed SH methodology is limited to parallel implementations with similar modules (or parts of a datapath) in *multiples of two* to achieve error cancellation. In an effort to overcome the aforesaid shortcoming, we propose a methodology for Internal-Self-Healing (ISH) that allows exploiting self-healing within a computing element internally without requiring a paired, parallel module. We employ the ISH methodology to design an approximate multiply-accumulate (MAC) accelerator, wherein the multiplier is regarded as an approximation stage and the accumulator as a healing stage. We propose to approximate a recursive multiplier in such a way that a near-to-zero average error is achieved for a given input distribution to cancel out the errors at an accurate accumulation stage. Our experiments show that the proposed ISH methodology relieves the *multiples of two* restriction for computing elements and enables error cancellation within a single computing element.

ix

As a case study of iterative and accumulation based algorithms, we apply our proposed approximate computing methodologies to radio astronomy calibration processing which results in a more effective quality-efficiency trade-off as compared to the state-of-the-art approximate computing methodologies.

SAMENVATTING

Computers worden voortdurend uitgedaagd door toepassingen, zoals bijvoorbeeld wetenschappelijke rekentoeepassingen, die veeleisend zijn. Dergelijke toepassingen vereisen een hoge hardware-efficiëntie en vormen dus een uitdaging om het energie-/stroom-verbruik, de benodigde rekestijd en het chip-oppervlak op een geïntegreerd circuit te verminderen om een vereiste taak te verwerken. Daarom is een verbetering van de hardware-efficiëntie één van de belangrijkste doelen bij de innovatie van computerapparatuur. Aan de andere kant hebben verbeteringen in het productieproces van geïntegreerde circuits een belangrijke rol gespeeld bij het aanpakken van dergelijke uitdagingen door de prestaties en de dichtheid van transistoren in geïntegreerde schakelingen te verhogen terwijl de vermogensdichtheid constant blijft. In de afgelopen decennia bereikten de efficiëntiewinsten als gevolg van verbeteringen in het productieproces echter fundamentele grenzen. De vermogensdichtheid is bijvoorbeeld niet zo goed schaalbaar als de dichtheid van transistoren. Het blijft daarom een uitdaging om het vermogen/thermische budget van de geïntegreerde schakelingen te beheersen.

Omdat veel toepassingen/algorithmes foutbestendig zijn, komen nieuwe paradigma's zoals *approximate computing* te hulp door veelbelovende efficiëntiewinsten te bieden, vooral op het gebied van energie-efficiëntie. Een toepassing/ algoritme kan worden beschouwd als foutbestendig of fouttolerant wanneer het een resultaat oplevert met de vereiste nauwkeurigheid, terwijl rekeneenheden worden gebruikt die niet altijd nauwkeurig rekenen. Er kunnen meerdere redenen zijn waarom een algoritme tolerant is voor fouten, een algoritme kan bijvoorbeeld ingangssignalen met veel ruis of redundante ingangssignalen en/of een reeks acceptabele resultaten hebben. Voorbeelden van dergelijke toepassingen zijn machine learning, sommige wetenschappelijke toepassingen en zoekmachines.

Approximate computing maakt gebruik van de intrinsieke fouttolerantie van dergelijke toepassingen om computersystemen op software-, architectuur- en circuit-niveau te optimaliseren zodat efficiëntiewinsten behaald worden. Echter, de huidige approximate computing technieken zijn niet voldoende gericht op het ontwerpen van specifieke circuits (acceleratoren) voor iteratieve en op accumulatie gebaseerde algoritmen. Rekening houdend met een breed scala van dergelijke algoritmen bij digitale signaalverwerking, worden in dit proefschrift benaderingsmethodologieën onderzocht om zeer efficiënte accelerator architecturen voor iteratieve en op accumulatie gebaseerde algoritmen te realiseren.

Hulpmiddelen voor de analyse van foutbestendigheid beoordelen of een (deel van een) algoritme een veelbelovende kandidaat is voor approximate computing. Statistische benaderingsmodellen (middels het introduceren van fouten) worden toegepast op een algoritme om de intrinsieke foutbestendigheid te kwantificeren en om veelbelovende approximate computing technieken te identificeren. In de context van iteratieve algoritmen laten we zien dat het hedendaagse statistische model niet effectief is in het blootleggen van mogelijkheden voor approximate computing. We stellen een adaptief statistisch benaderingsmodel voor dat een manier biedt om het aantal iteraties dat kan worden verwerkt met behulp van een *approximate core* te kwantificeren terwijl wordt voldaan aan de kwaliteitseisen.

Bovendien passen iteratieve algoritmen in het algemeen een convergentie criterium toe om een aanvaardbare oplossing aan te geven. Het convergentie criterium is een op precisie (het verschil tussen opeenvolgende iteraties) gebaseerde kwaliteitsfunctie die een garantie biedt dat de oplossing nauwkeurig genoeg is om de iteratieve berekeningen te beëindigen. We laten echter zien dat de op precisie gebaseerde kwaliteitsfunctie (het convergentie criterium) niet noodzakelijkerwijs voldoende is in de foutbestendighedsanalyse van iteratieve algoritmen. Daarom moet een aanvullende, op nauwkeurigheid (het verschil met de ideale oplossing) gebaseerde kwaliteitsfunctie worden gedefinieerd om de levensvatbaarheid van de approximate computing technieken te beoordelen.

Met het oog op energie-efficiëntie, stellen we een accelerator ontwerp voor iteratieve algoritmen voor. Ons ontwerp is gebaseerd op een heterogene architectuur, waar heterogeniteit wordt geïntroduceerd door een combinatie van *accurate* en *approximate cores* te gebruiken. Onze voorgestelde methodologie maakt gebruik van de intrinsieke foutbestendigheid van een iteratief algoritme, waarbij een aantal initiële iteraties wordt uitgevoerd op de approximate core en de rest op de accurate core om een vermindering van het energieverbruik te bereiken. Het door ons voorgestelde ontwerp van de accelerator verhoogt het aantal iteraties (die nodig zijn in de conventionele accurate tegenhanger) niet en biedt voldoende precisie om te convergeren naar een acceptabele oplossing.

De conventionele approximate ontwerpen volgen 'fout-beperkte' technieken. Deze technieken beperken de benaderingen op basis van de foutgroottes en de kans op het optreden van de fouten die ze introduceren om een onacceptabel kwaliteitsverlies tijdens verwerking te voorkomen. Aan de andere kant begrenzen de fout-beperkte technieken de hardware-efficiëntievoordelen die kunnen worden benut binnen foutbestendige applicaties. In het kader van op accumulatie gebaseerde algoritmen introduceren we een *Self-Healing (SH)* methodologie voor het ontwerpen van approximate acceleratoren zoals *square-accumulate (SAC)*, waarbij de benaderingen niet worden beperkt door grootte en frequentie van individuele fouten maar de mogelijkheid krijgen om fouten op te heffen binnen de verwerkingseenheden. SAC verwijst naar een hardware accelerator die met zichzelf een inwendig product van een vector berekent, waarbij de kwadraten van de elementen van een vector worden geaccumuleerd.

We gebruiken de SH-methodologie, waarbij de *squarer* (kwadrateereenheid) wordt beschouwd als een benaderingsfase en de *accumulator* als een *healing* fase. We stellen voor om een *approximate squarer mirror pair* in te zetten, zodat de fout die door een *approximate squarer* wordt geïntroduceerd, de fout, geïntroduceerd door de andere, weerspiegelt, d.w.z. de fouten die door de *approximate squarers* worden gegenereerd, heffen elkaar ongeveer op. Dit helpt de *healing*-fase (*accumulator*) om automatisch de fout op te heffen die in de benaderingsfase is ontstaan, en daardoor het kwaliteitsverlies te minimaliseren. Onze kwaliteit-efficiëntieanalyse van een *approximate SAC* laat zien dat de voorgestelde SH methodologie een effectievere afweging biedt in vergelijking met de conventionele fout-beperkte technieken.

Desalniettemin is de voorgestelde SH methodologie beperkt tot parallele implementaties met vergelijkbare modules (of delen van een datapad) in veelvoud van twee om foutopheffing te bereiken. In een poging om de bovengenoemde tekortkoming te verhelpen, stellen we een *Internal-Self-Healing (ISH)* methodologie voor die het mogelijk maakt om self-healing binnen een accelerator te exploiteren zonder een gepaarde, parallele module te vereisen. We gebruiken de ISH methodologie om een *approximate multiply-accumulate (MAC)* accelerator te ontwerpen, waarbij de vermenigvuldiger wordt beschouwd als een benaderingsfase en de *accumulator* als een *healing*-fase. We stellen voor om een recursieve vermenigvuldiger zo te ontwerpen dat een gemiddelde fout van bijna nul wordt bereikt voor een gegeven amplitude-verdeling van een ingangssignaal om de fouten op te heffen in een accurate accumulatiefase. Onze experimenten tonen aan dat de voorgestelde ISH methodologie de genoemde beperking tot veelvoud van twee wegneemt en foutopheffing binnen een enkel rekenelement mogelijk maakt.

Als case study van iteratieve en op accumulatie gebaseerde algoritmen, passen we onze voorgestelde *approximate computing* methodologieën toe op de kalibratie van een radiotelescoop, wat resulteert in een effectievere afweging van kwaliteit en efficiëntie in vergelijking met de hedendaagse *approximate computing* methodologieën.

ACKNOWLEDGEMENTS

It was a long journey, not only in terms of duration but also in terms of learning. It was full of ups and downs, where you need a mentor to help you get through the process. I would like to thank Dr. ir. André Kokkeler, my Ph.D. supervisor, for his invaluable mentorship. He encouraged me when I was underestimating my work and criticized me when I was over-estimating it. He knows very well how to maintain a balance between providing guidance and allowing freedom-of-decision to raise a student to the level of an independent researcher. I would also like to thank Prof. dr. ir. Gerard Smit for his general guidance and for his time to review this manuscript. Moreover, I would like to thank the graduation committee members for their review and suggestions to improve this manuscript.

I would also like to thank Dr. ir. Sabih Gerez for his support and critical discussions about the research ideas and experimentation. I also want to thank Prof. dr. ing. Muhammad Shafique (CARE-Tech, ECS group, TU Wien) for his collaboration and guidance. I would like to thank the Astron team, especially Dr. ir. Albert-Jan Boonstra, for providing me all the support required for the experimentation with radio astronomy calibration processing. I would also like to thank my co-authors and the students I have supervised during my Ph.D. tenure for helping me understand the subject better and to investigate it in various research directions. Special thanks to Muhammad Abdullah (CARE-Tech, ECS group, TU Wien) for helping me with the design space exploration of approximate multipliers.

I would like to thank the secretaries, supporting/scientific staff, and researchers/-students of the CAES group¹ for their support from finding a house in Enschede to finding a publisher for this manuscript. Thank you for the coffee breaks and fruitful social and technical discussions, for providing a nice thesis-template, and for helping me with the software tools, Dutch version of the Abstract (Samenvatting) and the \LaTeX related problems. I am sure, it would not be possible to reach this level at this point in time without your support. Moreover, I would like to thank the teaching and management team (TI/ELT-LED Saxion) not only for

¹Special thanks to Marlous Weghorst, Nicole Baveld, Thelma Nordholt, Jan Kuper, Bert Helthuis, Bert Molenkamp, Daniel Ziener, Marco Gerards, Ghazanfar Ali, Rinse Wester, Jochem Rutgers, Christiaan Baaij, Ahmed Ibrahim, Jerrin Pathrose, Hendrik Folmer, Ali Asghar, Anuradha Ranasinghe, Viktorio El Hakim, Luuk Oudshoorn, Arvid van den Brink, Vincent Smit, Alexander Karpukhin, Emil Rijnbeek, Bart Verstoep, Mark Krone, Shing Long Lin, Koen Raben, Johan Oedzes, Emil Kerimov, Masoud Abbasi, Mina Mikhael, Siavash Safapourhajari, Oguz Meteer, Guus Kuiper, Gijs Goeijen, Gerwin Hoogsteen, and Robert de Grootte.

their encouragement but also for providing me enough time to complete this manuscript.

xvi

I would like to thank the Pakistani community in Enschede, especially PSA (University of Twente), to help me & my family not to feel alone. Thank you for organizing social and sports events during my stay in Enschede. I would also like to thank the University of Twente management for providing on-campus facilities like the sports centre and prayers room that increased my efficiency of work during my stay here.

Life of a Ph.D. candidate is neither in-efficient nor complex, it's simply tough, especially when in a foreign country. One has to embrace an ambitious routine to get through. However, that's not possible without the support of life-partner. I would like to thank my wife, Zainab, for being resilient and for her consistent support from finding a suitable Ph.D. position to defending it. Furthermore, I would like to thank my parents for their utmost efforts and encouragement towards pursuing education since my childhood. I also want to thank my brother, sisters, other family members and friends for their support and well-wishes throughout my Ph.D. tenure.

Ghayoor Gillani
Enschede, July 2020.

CONTENTS

xvii

1	INTRODUCTION	1
1.1	Approximate Computing and Hardware Efficiency	2
1.1.1	<i>Approximate Computing</i>	2
1.1.2	<i>Error Resilience</i>	2
1.1.3	<i>Hardware Efficiency</i>	3
1.2	Problem Statement	3
1.3	Research Objective	4
1.4	Radio Astronomy Processing	4
1.5	Contributions	5
1.6	Thesis Outline and Organization	7
2	BACKGROUND	9
2.1	Inexact Computing	9
2.1.1	<i>Stochastic Computing</i>	10
2.1.2	<i>Probabilistic Computing</i>	10
2.1.3	<i>Approximate Computing</i>	11
2.2	Terminology	12
2.2.1	<i>Efficiency</i>	12
2.2.2	<i>Performance</i>	12
2.2.3	<i>Quality</i>	12
2.2.4	<i>Accuracy and Precision</i>	13
2.2.5	<i>Quality-Efficiency Trade-off</i>	13
2.2.6	<i>Pareto Optimal Designs and Pareto Front</i>	13
2.3	Error Resilience Analysis	14
2.3.1	<i>Quality of Service Profiler</i>	14
2.3.2	<i>Intel's Approximate Computing Toolkit</i>	15
2.3.3	<i>Automatic Sensitivity Analysis for Data</i>	15
2.3.4	<i>Statistical Error Resilience Analysis</i>	16
2.4	Approximate Computing Techniques	18
2.4.1	<i>Software Level Techniques</i>	18
2.4.2	<i>Architecture Level Techniques</i>	18

2.4.3	<i>Hardware-/Circuit-Level Techniques</i>	19
2.5	Approximate Recursive Multipliers	26
2.6	Evaluation	29
3	EXPLOITING ERROR RESILIENCE OF ITERATIVE ALGORITHMS	31
3.1	Related Work	33
3.1.1	<i>Adaptive Accuracy Techniques</i>	33
3.1.2	<i>Error Resilience Analysis Techniques</i>	34
3.2	Error Resilience Analysis of Iterative Algorithms	35
3.2.1	<i>Adaptive Statistical Approximation Model (Adaptive-SAM)</i>	35
3.2.2	<i>High-level Error Resilience Analysis</i>	37
3.2.3	<i>Significance of Quality Function Reconsideration</i>	41
3.3	Energy Efficient Accelerator Design for Iterative Algorithms	42
3.3.1	<i>Design of a Heterogeneous Least Squares Accelerator</i>	43
3.3.2	<i>Experimental Results</i>	46
3.4	Conclusions	52
4	ERROR CANCELLATION IN ACCUMULATION BASED APPROXIMATE ACCELERATORS	55
4.1	Related Work	58
4.2	Self-Healing Methodology for Approximate Square-accumulate (SAC)	59
4.2.1	<i>Terminology</i>	59
4.2.2	<i>Employing Self-Healing for Approximate SAC Architecture</i>	60
4.3	Analysis of Approximate SAC Composed of Truncated Squarer	61
4.3.1	<i>Mathematical Analysis of Truncated Squaring</i>	61
4.3.2	<i>Quality Analysis of Various Truncation Alternatives</i>	63
4.4	Absolute Approximate Squarer Mirror Pair (AASMP)	66
4.4.1	<i>Design of 2×2 Absolute Approximate Mirror Pairs</i>	68
4.4.2	<i>8×8 AASMP Design</i>	69
4.4.3	<i>$n \times n$ AASMP Design</i>	70
4.5	Designing an Optimal Approximate SAC Accelerator	70
4.6	Experimental Setup and Results	77
4.6.1	<i>Experimental Setup for Quality-efficiency Trade-off Study</i>	77
4.6.2	<i>Quality-efficiency Trade-off of 8×8 Squarer Pairs in a SAC Accelerator</i>	77
4.6.3	<i>Radio Astronomy Calibration Processing – A Case Study</i>	80
4.6.4	<i>Discussion and Future Work</i>	80
4.7	Conclusions	81

5	INTERNAL-SELF-HEALING METHODOLOGY FOR ACCUMULATION BASED APPROXIMATE ACCELERATORS	85
5.1	Related Work	87
5.2	Designing an Approximate MAC with the Internal-Self-Healing (ISH) Methodology	90
5.2.1	<i>Approximate Multiplier for MAC</i>	91
5.2.2	<i>Overflow Handling</i>	91
5.2.3	<i>Comparison of the proposed ISH with the conventional approxi- mate computing methodology</i>	95
5.3	Experimental Results	97
5.3.1	<i>Experimental Setup</i>	97
5.3.2	<i>Design Space Exploration of the Proposed ISH methodology</i>	98
5.3.3	<i>Scalability and Comparison of the ISH with the Conventional Methodology</i>	99
5.3.4	<i>Case Study: Radio Astronomy Calibration Processing</i>	104
5.3.5	<i>Synthesis based comparison</i>	105
5.3.6	<i>Discussion and Future Work</i>	108
5.4	Conclusions	109
6	CONCLUSIONS AND RECOMMENDATIONS	113
6.1	Contributions	113
6.1.1	<i>Error Resilience Analysis Of Iterative Algorithms</i>	113
6.1.2	<i>Exploiting Error Resilience Of Iterative Algorithms</i>	114
6.1.3	<i>Designing Approximate Accelerators For Accumulation Based Al- gorithms</i>	114
6.1.4	<i>Radio Astronomy Calibration Processing – A Case Study</i>	116
6.2	Recommendations for future work	116
A	8 × 8 SQUARER CONSTRUCTION	121
B	QUALITY EVALUATION FOR APPROXIMATE SQUARERS	125
C	DESIGN SPACE EXPLORATION OF APPROXIMATE MULTIPLIERS FOR MAC	131
C.1	Huge Design Space - A Challenge	131
C.2	Design Space Exploration	132
C.3	Viability of our approach	137

ACRONYMS 141

xx **BIBLIOGRAPHY** 143

LIST OF PUBLICATIONS 155

CONTENTS

1

INTRODUCTION

ABSTRACT – While the efficiency gains due to process technology improvements are reaching the fundamental limits of computing, emerging paradigms like approximate computing provide promising efficiency gains for error resilient applications. However, the state-of-the-art approximate computing methodologies do not sufficiently address the accelerator designs for iterative and accumulation based algorithms. Keeping in view a wide range of such algorithms in digital signal processing, this thesis investigates systematic approximation methodologies to design high-efficiency accelerator architectures for iterative and accumulation based algorithms. As a case study of such algorithms, we have applied our proposed approximate computing methodologies to a radio astronomy calibration application.

Increasing hardware efficiency is one of the major targets to innovate computing devices. This includes the following, (1) reducing the size/chip-area of a transistor, i.e., increasing the number of transistors per unit area (*transistor density*), (2) reducing the power consumption of a transistor to keep the *power density* constant while the transistor density is increased, (3) increasing the speed, i.e., increasing the *performance*. The increase in hardware efficiency is generally achieved by the advancements in Very Large Scale Integration (VLSI) technology. The improvements in transistor density are more or less following Moore's law [84]. The law states that the transistor density doubles every 1.5 years. For that matter, we have been witnessing smaller sizes of devices that have gradually brought gadgets in our hands. In the last century, the advancements in VLSI technology were also following Dennard's scaling of keeping the power density constant [28].

However, there are physical limitations to the increase in efficiency of computing devices [10, 73]. One of the biggest challenges faced by designers today is power/energy-consumption (Dennard's scaling) [34]. The power density is not scaling as well as compared to the transistor density [13, 73]. The consequence is that

a part of an integrated circuit (IC) has to be turned-off to control the power budget, bringing us to the era of *dark silicon* [33, 73, 113]. While architectural power management techniques like Dynamic Voltage and Frequency Scaling (DVFS) and clock-/power-gating are not enough to meet the power challenges [115], new computing paradigms have to be explored. One of the paradigm shifts is to move from conventional 'always correct' processing to processing where controlled errors are allowed. In this thesis, computing techniques that are based on the latter paradigm are called *approximate computing techniques* or in short *approximate computing*.

1.1 APPROXIMATE COMPUTING AND HARDWARE EFFICIENCY

1.1.1 APPROXIMATE COMPUTING

Approximate computing can be regarded as an aggressive optimization because it allows controlled inexactness and provides results with the bare minimum accuracy to increase computing efficiency. An increase in computing efficiency or simply *efficiency*¹ means reduction in computing costs like run-time, chip-area, and power/energy consumption. The introduction of inexactness brings errors in the intermediate and/or the final outcomes of the processing compromising output quality, or simply the *quality* of processing. Approximate computing has shown high-efficiency gains for error-resilient applications like multimedia processing, machine learning and search engines [128, 132]. Such applications tolerate a quantified error within the computation while producing an acceptable output.

1.1.2 ERROR RESILIENCE

An application/algorithm can be regarded as *error-resilient* or *error-tolerant* when it provides an outcome with required accuracy while utilizing processing components that do not always compute accurately. There are several reasons why an application is tolerant of errors as discussed in [26]. These include noisy or redundant inputs of the algorithm, approximate or probabilistic computations within the algorithm, and a range of acceptable outcomes.

Image processing and search engines are among the prominent examples of error-resilient applications. The outcome of image processing is generally observed by humans who have perceptual limits, therefore, the outcome is acceptable as far as the observer cannot differentiate between the quality of an accurately computed image and an approximately computed image. In the case of search engines, a similarity rank is computed between a vector in the search space with that of the objective vector. A high similarity rank means a better matching of the search objective. While computing the similarity ranks, accurate computations

¹See Section 2.2 for definitions.

are not required as far as the similarity ranking of the search vectors remains the same.

The quantification of error tolerance is achieved by utilizing error resilience analysis tools [26, 41, 78, 80]. Approximate computing techniques exploit this error tolerance to optimize the computing systems at software-, architecture- and circuit-level to achieve efficiency gains [50, 81, 115].

1.1.3 HARDWARE EFFICIENCY

An increase in hardware efficiency means a reduction in computing costs at the circuit-/hardware-level, e.g., latency within circuits, chip-area, and power/energy requirements of the circuit to compute an algorithm.

At the hardware level, the prominent approximation techniques are transistor-level pruning and logic-level pruning. Pruning refers to the elimination of the parts of a circuit that have a low contribution towards the final output. In this regard, approximate adders and multipliers have been researched for their indispensable role in digital signal processing [39, 63, 101, 114]. For instance, Kulkarni et al present an approximate multiplier that provides 32% to 45% power reduction with an average error of 1.4% to 3.3%. For an image filtering application, they demonstrate an average power reduction of 41% with a signal-to-noise ratio (SNR) of 20.4 dB [63].

1.2 PROBLEM STATEMENT

While the state-of-the-art approximate computing techniques have shown highly-efficient adders and multipliers, they do not sufficiently address accelerator designs for iterative and accumulation based algorithms. Iterative algorithms are mathematical methods that utilize an initial guess to compute a sequence of approximate solutions until the outcome converges to an acceptable solution. An accumulation based algorithm accumulates the outcome of its component-process, e.g., a multiplication, to compute an overall outcome. For example, to compute an inner product of two vectors, products of corresponding elements of the vectors are accumulated. Such an algorithm can be implemented as a multiply-accumulate (MAC) unit/accelerator. Similarly, to compute an inner product of a vector, squares of the elements of the vector are accumulated. Such an algorithm can be implemented as a square-accumulate (SAC) unit/accelerator.

The state-of-the-art approximate computing methodologies apply approximations by restricting the error rates and/or error magnitudes. This ensures an acceptable outcome when applied to general error-resilient algorithms. However, when applied to iterative and accumulation based algorithms, these techniques limit the achievable efficiency gains due to limited approximations.

1.3 RESEARCH OBJECTIVE

Keeping in view a wide range of iterative and accumulation based algorithms [26, 76, 104, 107], the research objective for this thesis is the following,

Investigating high-efficiency approximate accelerator designs for iterative and accumulation based algorithms.

We further decompose our research objective into the following research questions,

- » How to analyze iterative algorithms for error resilience? and how trustworthy is a precision-based quality metric (convergence) in the error resilience analysis process?
- » How to exploit intrinsic error resilience of iterative algorithms effectively, i.e., how to design approximate accelerators for such algorithms?
- » How to design high-efficiency approximate accelerators for accumulation based algorithms? In accumulation based algorithms like MAC (or SAC), there is an accumulation stage after multiplication (or squaring). If the multiplier (or squarer) is approximated, the accumulator accumulates the error. Is it possible to design such approximate multipliers (or squarers) that bring an opportunity to cancel out errors within accumulation without the overhead of error correction circuitry?
- » Considering a case study of radio astronomy processing, how do the proposed approximate computing methodologies affect the quality and efficiency of the processing? Moreover, what are the opportunities and challenges to embrace approximate computing principles for radio astronomy processing?

In view of the above, we have investigated error resilience analysis techniques and approximate computing elements targeted for the iterative and accumulation based algorithms. We have performed a case study of a radio astronomy processing application, namely the calibration processing, which is an iterative algorithm with underlying accumulation based computations.

1.4 RADIO ASTRONOMY PROCESSING

Radio astronomy studies celestial objects by utilizing radio telescopes. Modern radio telescopes like the Square Kilometer Array (SKA) aim to increase our understanding of the universe like creation and evolution of galaxies, cosmic magnetism, and the possibility of life beyond earth [1]. To investigate such phenomena, a radio telescope has to offer very high sensitivity, resolution, and survey speed [19]. This brings terabytes of raw data per second to be processed. Consequently, radio astronomy processing is an energy-/power-hungry application.

Imaging in radio astronomy is mainly composed of the following steps [14, 126]: correlation of digitized input signals acquired from pairs of distinct stations to obtain *visibilities*, calibrating the instrument gains for environmental effects, and converting the corrected visibilities to sky images. The Science Data Processing (SDP) pipeline of radio astronomy processing acquires the visibilities as input and generates a radio image of the sky as output. It consists of an instrument calibrator, gridded, and FFTs [51, 124, 126] and is dominated by iterative and accumulation based algorithms like least squares [107]. An estimation of power consumption was made for the SKA in 2014, which predicts a power consumption of 7.2MW for the fused multiply-add operations within the SDP pipeline of the medium frequency array SKA1-Mid [51].

The input signal received at a radio telescope has a low signal-to-noise ratio (SNR) and can be regarded as Gaussian noise [21]. The signal processing pipeline in radio astronomy can be considered as an error-resilient application because it reflects the following attributes: noisy/redundant data input, and approximate/statistical computation patterns. An example is the calibration algorithm, StEFCal [107]. It computes antenna gains of a radio telescope, iteratively, by processing the model and measured visibilities. It utilizes a least squares algorithm that is approximate in nature.

In this thesis, the calibration processing (StEFCal) is utilized as a case study to analyze and develop promising approximate computing methodologies for iterative and accumulation based algorithms. In Chapter 3, we discuss the StEFCal algorithm in detail and present its error analysis. Continuing with Chapter 3, and also in Chapter 4 and Chapter 5, we present the quality-efficiency advantages for StEFCal based on proposed accelerator designs. Our approximate accelerator designs provide efficiency benefits for Application Specific Integrated Circuits (ASICs). On the other hand, Field Programmable Gate Arrays (FPGAs) based acceleration is also common in radio astronomy processing [46, 93, 131]. It is to be noted that our designs may require modifications for optimized FPGA based acceleration as the approximate computing techniques do not directly translate into efficiency benefits for FPGA based architectures, see Section 2.4.3. Finally, in Chapter 6, we discuss overall opportunities and future directions for energy-/power-efficient radio astronomy processing from the perspective of approximate computing techniques (based on [G:5]).

1.5 CONTRIBUTIONS

This thesis contributes to approximate computing methodologies for iterative and accumulation based algorithms by providing several improvements, such as;

We contribute to improving the error resilience analysis of iterative algorithms that utilize a convergence criterion to indicate an acceptable solution. The convergence criterion is a precision-based quality function that provides a guarantee that the solution is precise enough to terminate the iterative computations. We

propose an adaptive statistical approximation model for error resilience analysis, which provides an opportunity to divide an iterative algorithm into exact and approximate iterations. This improves the existing error resilience analysis methodology by quantifying the number of approximate iterations in addition to other parameters used in the state-of-the-art techniques. Moreover, we demonstrate that the precision-based quality function (the convergence criterion) is not necessarily sufficient in the error resilience analysis of iterative algorithms. Therefore, an additional accuracy-based quality function has to be defined to assess the viability of the approximate computing techniques.

We propose an energy-efficient accelerator design for iterative algorithms. Our design is based on a heterogeneous architecture, where the heterogeneity is introduced using accurate and approximate processing modules. Our proposed methodology exploits the intrinsic error resilience of an iterative algorithm by processing the initial iterations on approximate modules while the later ones are processed on accurate modules. Our accelerator design does not increase the number of iterations (as compared to the conventional accurate counterpart) and provides sufficient precision to converge to an acceptable solution.

We propose an error-cancellation based design methodology for accumulation based approximate accelerators like square-accumulate (SAC). We employ a self-healing² (SH) methodology, wherein the squarer is regarded as an approximation stage and the accumulator as a healing stage. We propose to deploy an approximate squarer *mirror pair*, such that the error introduced by one approximate squarer *mirrors* the error introduced by the other, i.e., the errors generated by the approximate squarers are approximately additive inverse of each other. This helps the healing stage (accumulator) to automatically cancel out the error originated in the approximation stage, and thereby to minimize the quality loss. Our case study shows that the proposed SH methodology provides a more effective quality-efficiency trade-off as compared to the conventional approximate computing methodology.

Nevertheless, the SH methodology is constrained to parallel implementations with similar modules (or parts of a datapath) in *multiples of two* to achieve error cancellation. Therefore, we propose a methodology for Internal-Self-Healing (ISH) that allows exploiting self-healing within a computing element internally without requiring a paired, parallel module. We employ our ISH methodology to design an approximate multiply-accumulate (xMAC), wherein the multiplier is regarded as an approximation stage and the accumulator as a healing stage. We propose to approximate a recursive multiplier in such a way that a near-to-zero average error is achieved for a given input distribution to cancel out the error at an accurate accumulation stage.

The above contributions address our research objective to design approximate accelerators for iterative and accumulation based algorithms that bring a more

²In this thesis, the term *self-healing* differs from that of [59], see Chapter 4 for details.

effective quality-efficiency trade-off as compared to the state-of-the-art approximate computing methodologies. It is to be noted that quantitative results very much depend on specific synthesis technology, tooling, and settings. Therefore, we have performed quantitative comparisons between designs that belong to the proposed and the state-of-the-art approximate computing methodologies by implementing them using the same technology, tooling, and settings.

1.6 THESIS OUTLINE AND ORGANIZATION

Following this introductory chapter, we provide a brief background of the approximate computing field in Chapter 2. The background discusses in-exact computing in general and approximate computing techniques in particular. In Chapter 3, we discuss our error resilience analysis methodology for iterative algorithms and propose an energy-efficient approximate least squares accelerator design for iterative algorithms. Chapter 4 and Chapter 5 propose error-cancellation based approximate accelerators for SAC and MAC processing. Finally, Chapter 6 discusses the overall conclusions of our research and indicates the further line of action towards high-efficiency approximate accelerators for iterative and accumulation based algorithms.

While Chapter 2 provides a basic understanding of approximate computing techniques, the *related work* of each contribution is discussed in the specific chapter. The references are provided in the *Bibliography* section. However, references to our own publications are provided in the *List of Publications* section. Own publications are cited as [G:<number>], e.g., [G:4].

2

BACKGROUND

ABSTRACT – When an algorithm is resilient to the effects of noise in its computation or tolerates a relaxation in its specifications, deviations from accurate behavior can be traded by software and hardware to achieve a higher computing efficiency. This chapter discusses such computing paradigms like stochastic, probabilistic and approximate computing. Moreover, we discuss the approximate computing concepts that help the readability of the subsequent chapters.

2.1 INEXACT COMPUTING

Inexact computing allows controlled errors in computing to increase efficiency. Efficiency gains have been demonstrated for error-resilient applications such as multimedia digital signal processing, search engines, radio communication, machine learning, and scientific computing [3, 4, 81, 132]. The design target in inexact computing is to achieve the best possible computing efficiency for a given quality-constraint of the algorithm, or to achieve the best quality output for a given cost constraint. In literature, inexact computing is also coined as *best-effort computing*, as it executes an algorithm without guaranteeing a correct output, i.e., an algorithm is executed on best-effort bases [18, 76]. Another related term is *error-efficient computing*, originating from the notion that it prevents as many errors as necessary to execute an algorithm [119].

In literature, inexact design techniques are mainly divided into three categories, namely: stochastic computing, probabilistic computing, and approximate computing.

This chapter is partly based on [G:7].

2.1.1 STOCHASTIC COMPUTING

In stochastic computing, data is represented with randomized bit-streams. In contrast to normal binary computation, there is no significance in the order of 1's and 0's [4, 5]. For instance, both (1, 1, 0, 1, 1, 1, 0, 1) and (1, 0, 1, 1, 1, 0, 1, 1) mean 0.75 in stochastic computing as the probability of having a 1 at an arbitrary position is 0.75. The advantage is that the computations become simple, e.g., a simple AND operation provides the multiplication computation. Consider an example of multiplying two numbers x and y , which are numbers between 0 and 1. Their product can be given as: $P = x * y$. By applying the stochastic computing principle,

$$P' = x' \wedge y' \quad (2.1)$$

where \wedge is the bit-wise AND operation. x' and y' are the randomized bit (stochastic) representations of x and y . Let $x = 4/8$ and $y = 6/8$ and the randomized bit representations of x and y are (0, 1, 1, 0, 1, 0, 1, 0) and (1, 0, 1, 1, 1, 0, 1, 1) respectively. Therefore,

$$P' = (0, 1, 1, 0, 1, 0, 1, 0) \wedge (1, 0, 1, 1, 1, 0, 1, 1) = (0, 0, 1, 0, 1, 0, 1, 0) \quad (2.2)$$

where (0, 0, 1, 0, 1, 0, 1, 0) represents 3/8, the expected result of the multiplication operation.

Nevertheless, stochastic computing implies computing on probabilities and there is a chance of error due to various possible randomized bit streams. For instance, it is equally possible that $x = 4/8$ and $y = 6/8$ are represented by the following randomized bit representations: $x' = (0, 1, 0, 1, 1, 1, 0, 0)$ and $y' = (1, 1, 1, 0, 1, 0, 1, 1)$ [4]. In such a case,

$$P' = (0, 1, 0, 1, 1, 1, 0, 0) \wedge (1, 1, 1, 0, 1, 0, 1, 1) = (0, 1, 0, 0, 1, 0, 0, 0) \quad (2.3)$$

where (0, 1, 0, 0, 1, 0, 0, 0) represents 2/8, which is an approximation of the expected result.

Formally introduced in 1960 [36], stochastic computing was an attractive choice of computing for its simple arithmetic operations like multiplication, especially when the transistors were expensive. However, as the transistors became cheaper, its advantages were dominated by its disadvantages like slow speed and limited precision [4]. Keeping in view the disadvantages of stochastic computing, it has a limited application range, a few examples are specific control systems [122] and neural networks [29, 58].

2.1.2 PROBABILISTIC COMPUTING

Probabilistic computing refers to a circuit (or fundamentally a CMOS switch) operating at such a low-voltage that its intrinsic noise affects its behavior. The consequence is that a trade-off is introduced between energy consumption (E) and the probability of correct output (p). In his pioneering research [90, 91],

Krishna V. Palem showed that the potential energy saving of a probabilistic switch (as compared to a deterministic switch) is $k_B T \ln(1/p)$ Joules. Here T and k_B refer to temperature and the Boltzmann constant respectively. This shows that decreasing the probability of correctness (p) results in an increase in energy savings and vice versa.

By employing probabilistic computing, an improvement in energy efficiency has been shown for probabilistic algorithms like probabilistic cellular automata in [22]. It is to be noted that a probabilistic algorithm requires a random source while being processed by deterministic hardware. On the other hand, when using a randomized (probabilistic) switch as a basic building block, an explicit random source is not required [91]. Moreover, energy efficiency improvements have also been demonstrated for digital signal processing algorithms (other than probabilistic algorithms) that can tolerate quantified noise in computations [37, 57].

Probabilistic computing is referred to as a non-deterministic inexact computing paradigm, i.e., if a specific input is provided for several times, a specific output is not guaranteed. Only the probability of correct output (p) is guaranteed for which noise-based models have been formulated in [24, 66, 91]. These models provide probability of correctness (p) as a function of noise RMS [24]. In the context of multi-stage probabilistic circuits (e.g., a ripple carry adder), these models assume error propagation based on the probability of incorrectness ($1 - p$) of each stage only [66]. We have contributed to identifying the impact of delay propagation in probabilistic multi-stage circuits. The delay introduced due to low-voltage operation also adds to the error that is propagated through a multi-stage circuit. Our results highlight a need to improve the existing probabilistic computing models to include the effects of gate-width and frequency of operation [G:6].

Featuring a low-voltage scheme, probabilistic computing gained a lot of interest in its beginning for its high energy efficiency. However, it proved to be less effective as compared to *deterministic inexact computing*, especially for the algorithms that do not have probabilistic nature. In [71], Palem and his co-authors show that removing some parts of a circuit based on their low probability of usage introduces deterministic inexact computing that provides a more effective trade-off as compared to probabilistic computing. Moreover, probabilistic computing is only valid for the voltage of operation nearly equivalent to the CMOS intrinsic noise level. As this is not true for any CMOS technology at present, probabilistic computing is not an attractive approach for inexact computing nowadays [110].

2.1.3 APPROXIMATE COMPUTING

Besides bringing a different representation of signals (stochastic computing) or bringing a circuit to operate in the probabilistic region (probabilistic computing), there is plenty of research named under *approximate computing*. This emerging paradigm introduces approximations at software-, architecture-, and hardware-

level to achieve efficiency benefits. Loop perforation, reducing refresh rate of Dynamic Random Access Memory (DRAM), and circuit pruning are among the prominent examples of approximate computing.

In this thesis, we mainly focus on the approximate computing paradigm and develop methodologies for iterative and accumulation based algorithms. A brief survey of approximate computing techniques is provided in Section 2.4 along with the explanation of the designs that serve to ease the readability of the following chapters. We first provide the terminology to introduce the terms that are used throughout the thesis.

2.2 TERMINOLOGY

2.2.1 EFFICIENCY

In computing systems, the term *computing efficiency* (or simply *efficiency*) is used in contradiction of computing resource usage or *computing costs* for executing a specific task. It is defined as the output of a computing system per unit resource input, e.g., energy efficiency of a floating-point processor is defined as floating-point operations per Joule or floating-point operations per second per Watt. An increase in *efficiency* is referred to as a reduction in computing costs like chip-area, runtime, and/or power/energy consumption. In this thesis, following the literature, an increase in computing *efficiency* or a decrease in computing *costs* means the same; and a decrease in computing *efficiency* or an increase in computing *costs* means the same.

2.2.2 PERFORMANCE

Performance is the reciprocal of *execution time* [44]. Let t_1 be the execution time of a process, its performance can be given as $1/t_1$. Performance can also be defined as the output of a computing system per unit time, e.g., performance of a floating-point processor can be given as floating-point operations per second (FLOPS). In general, and also in this thesis, *performance* is referred to as the speed of the computation. Therefore, an increase in *performance* means decreasing runtime to execute a specific task. For instance, an increase in performance is achieved by reducing the latency of a circuit or by reducing the number of iterations that utilize a specific circuit.

2.2.3 QUALITY

The term *output quality* (or simply *quality*) is defined in contradiction to *deviation from exact behavior* or *error*. In this thesis, unless explicitly mentioned, the terms *exact* and *accurate* refer to a specified precision where there is no approximation involved with reference to the specified precision. For instance, if the specified precision is 8-bit, the 8-bit design (e.g., an 8-bit multiplier) is considered as accurate or exact design. Any approximations, e.g., in terms of data (e.g.,

reducing the precision of inputs to 7-bit) or circuit (e.g., removing parts of 8-bit multiplier circuit), brings an *inexact* or *in-accurate* or *approximate* entity, where the *entity* is referred to as circuit and/or data. Moreover, an increase in *quality* is referred to as a reduction in *error*. In literature, both terms (quality and error) have been used to indicate the output quality. Also in this thesis, an increase in output *quality* or a decrease in output *error* means the same; and a decrease in output *quality* or an increase in output *error* means the same.

2.2.4 ACCURACY AND PRECISION

Accuracy defines how close the output of a system is to that of the exact behavior. In this thesis, we define *exact behavior* as the theoretical behavior of a computing system for a specified precision. For instance, an 8-bit multiplier has an exact behavior when 8-bit inputs are multiplied without any approximation. On the other hand, *precision* is referred to as the amount of detail utilized in representing the output [2]. Therefore, it provides a measure of how close the outputs of a specific system are to each other. In the context of iterative algorithms, where the iteration process is terminated based on convergence, the precision of an approximate computing system is also important. In Chapter 3, we elaborate on this difference based on our analysis of an iterative algorithm.

2.2.5 QUALITY-EFFICIENCY TRADE-OFF

While approximate computing aims to increase the efficiency of a computing system, some errors may be introduced that degrade the quality of the output. Systematically increasing the level of approximations, e.g., gradually decreasing the bit-width of operands in a multiplier, generally increases the efficiency of a computing system and decreases the output quality. This introduces a trade-off between the quality of output and the efficiency and is referred to as *quality-efficiency trade-off*. For illustration, Fig. 2.1 shows the area and mean error of several design alternatives of an 8-bit squarer with different quality (or conversely error) and efficiency (or conversely resource usage) levels. In literature, another term is also used: *quality-cost trade-off*, which is merely an alternate term.

2.2.6 PARETO OPTIMAL DESIGNS AND PARETO FRONT

In approximate computing, only those design alternatives are interesting that provide the best efficiency for a given quality constraint or provide the best quality for a given efficiency target. Such design alternatives are called *pareto optimal designs* or *pareto optimal configurations* and are represented as *pareto optimal points* in the trade-off plot as shown in Fig. 2.1. All the other design alternatives are referred to as *sub-optimal designs* or *sub-optimal configurations* and are represented as *sub-optimal points* in the trade-off plot. The line joining the pareto-optimal points is referred to as *pareto front*. While comparing two approximate computing methodologies, their pareto fronts can be compared to

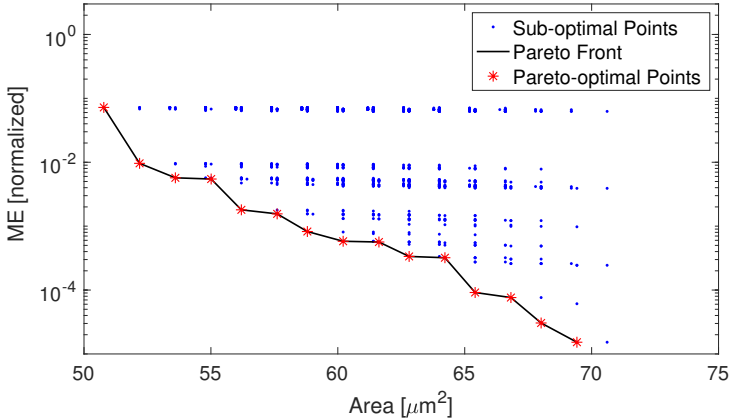


Figure 2.1: An illustration of quality-efficiency trade-off of an approximate 8-bit squarer for uniformly distributed input. ME stands for mean error. The chip-area (Area) values are estimated for TSMC 40nm Low Power (TCBN40LP) technology synthesized at 1.43GHz. Pareto-optimal points are chosen that provide the best efficiency designs for a given quality constraint and vice versa.

find which one provides a more effective trade-off, see Chapter 4 (Fig. 4.12) for an example.

2.3 ERROR RESILIENCE ANALYSIS

Error resilience is inherent to an application due to its possibly redundant/noisy real-time inputs, probabilistic or self-healing computational patterns, and a range of acceptable outputs [26]. However, in general, there are error-sensitive parts or kernels within every error-resilient application. Therefore, it is important to analyze applications for error resilience to separate the error-sensitive parts from that of error-tolerant parts and to get insights into promising approximation techniques before employing the implementation efforts [26, 108]. In this section, we discuss some of the important works that analyze the error resilience of applications.

2.3.1 QUALITY OF SERVICE PROFILER

The quality of service (QoS) profiler indicates the resilient parts of an application that can be replaced with approximate computations to gain performance with a low error introduction [78]. It transforms loops within an application to perform a reduced number of iterations to generate a quality-efficiency trade-off. This technique is called loop perforation in literature. The QoS profiler utilizes a user-provided quality metric to quantify the resilience within the sub-

computations. The authors applied their technique to several applications and demonstrated an increase in performance (two to three times) with a less than 10% of quality degradation.

2.3.2 INTEL'S APPROXIMATE COMPUTING TOOLKIT

The intel's approximate computing toolkit (iACT) is an open-source tool that analyzes the error resilience of an algorithm by applying approximations to user annotated pragmas [80]. Similar to QoS, the iACT toolkit offers resilience analysis based on a quality function provided by the user. However, unlike QoS, the sub-computations to be considered for the error resilience analysis are also identified by the user.

Specifically, the user identifies the parts of the code, say functions, with pragmas. The *pragma_axc* simulates the noisy hardware behavior, i.e., noisy load and store effects in memory operations, and noisy computation effects in floating point arithmetic instructions. The *pragma_axc_memoize* applies *approximate memoization* to the annotated sub-computation, where *memoization* refers to creating a table of outputs based on the ranges of inputs. Instead of executing an expensive floating-point operation, the related approximate output is selected from the table by just looking at the input range. The *pragma_axc_precision_reduce* reduces the precision range of floating-point operations to fixed-point operations. The authors applied their tool to sobel filtering, bodytracking and classification algorithms and demonstrated up to 22% of energy reductions with a maximum of 10% quality degradation.

2.3.3 AUTOMATIC SENSITIVITY ANALYSIS FOR DATA

Unlike iACT, the automatic sensitivity analysis for approximate computing (ASAC) tool, analyzes the data sensitivity only and in an automatic fashion, without the user annotation [105]. In this error resilience analysis technique, the variables of a program are systematically perturbed to assess their effect on the output quality. The variables are ranked based on their contribution to the output error. Given the overall ranking of the variables, they are classified as *approximable* and *non-approximable*. To demonstrate the viability of their approach, the authors applied ASAC to a set of benchmark applications and identified approximable and non-approximable variables. Afterwards, they applied bit-flip error behavior in their identified approximable variables for the FFT algorithm and showed that a less than 4% quality degradation is observed. However, applying the same error behavior to their identified non-approximable variables, they showed that the output becomes simply unacceptable.

Nevertheless, ASAC is a dynamic tool that requires computationally expensive runs of the target algorithm [106]. On the other hand, the program analysis for the approximation-aware compilation (PAC) tool introduces a static analysis method that has a significantly less runtime as compared to dynamic tools like

$B \backslash A$	00	01	10	11
00	0000	0000	0000	0000
01	0000	0001	0010	0011
10	0000	0010	0100	0110
11	0000	0011	0110	0111

(a) Truth table of M1 [63].

$B \backslash A$	00	01	10	11
00	0000	0000	0000	0000
01	0000	0000	0010	0010
10	0000	0010	0100	0110
11	0000	0010	0110	1001

(b) Truth table of M2 [101].

Figure 2.2: Truth tables of two approximate multiplier (2×2) designs discussed in [115]. M1 has a lower error rate and higher error magnitude as compared to that of M2.

ASAC [106]. In addition to distinguishing the variables as approximable and non-approximable, PAC quantifies the Degree of Approximation (DoA) for each variable. The DoA guides the level of approximation that can be applied to the data, e.g., the number of least significant bits of a variable that can be approximated.

2.3.4 STATISTICAL ERROR RESILIENCE ANALYSIS

A Motivational Example

Consider two approximate multiplier designs ($AxMul_1$ and $AxMul_2$) as discussed in [115]. Here we refer to them as M1 and M2. These multipliers have a size of 2×2 (input size= 2-bit for each operand) and can be used to construct higher-order multipliers, e.g., 4×4 , 8×8 , and so on. M1 has better area and power costs as compared to the accurate design with one error case (error magnitude=2) out of sixteen possible cases, see Fig. 2.2a for the truth table where the error case is marked in black. M2 is even more energy-efficient as compared to M1. However, the error rate for M2 is three out of the sixteen possible cases (error magnitude=1), see Fig. 2.2b for the truth table. Therefore, M2 has a higher error rate and a lower error magnitude as compared to M1 while offering better energy efficiency.

The selection from such design choices is based on an algorithm's error resilience characteristics depending on whether the target algorithm can tolerate a higher error rate or higher error magnitude. Moreover, this design space (number of alternatives) becomes larger for higher-order multipliers that can have a number of such multipliers (approximate or accurate) and a number of adders (approximate or accurate) for the adder tree to compute the final higher-order product [101]. For that matter, it is important to analyze an algorithm for statistical error resilience, which is referred to as *high-level error resilience*.

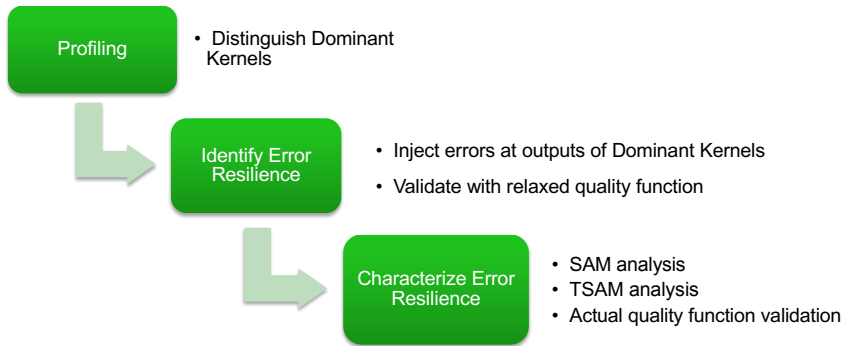


Figure 2.3: Error resilience analysis methodology based on the application resilience characterization framework [26]. The dominant parts of the application are distinguished in the profiling phase and tested for error resilience by injecting errors. The identified error resilient parts are then characterized by applying the statistical approximation model (SAM) and technique specific approximation model (TSAM).

ARC Framework

The Application Resilience Characterization (ARC) framework [26] includes the statistically distributed error injection model to generate the *statistical error resilience profile* of an algorithm. That is, it quantifies the error resilience of an application based on statistical parameters: error mean (EM), error predictability (EP) and error rate (ER). EM determines the *mean* of the normally distributed error. EP corresponds to the *standard deviation* of the normally distributed error [26]. Noteworthy, referring standard deviation to error predictability is nonintuitive because when the standard deviation is increased the predictability of error does not increase. However, to maintain the convention of the authors in [26], we also use EP to indicate the standard deviation of the error. ER defines the rate at which errors are injected in the approximation analysis. The statistical error resilience profile helps to reduce the available design space in order to choose the best possible quality-cost design alternative.

An overview of the ARC methodology is shown in Fig. 2.3. The first step is to identify the dominant kernels based on their run-time share in the profiling phase. The kernels that run for at least 1% of the total execution time are selected to perform analysis. Secondly, the error resilience is identified by injecting random errors in the outputs of the dominant kernels and the overall output of the application is compared with a relaxed quality function to distinguish potentially resilient kernels from that of the sensitive ones. Relaxed quality function means that the application behavior is only checked for relatively bigger errors (e.g., if the application crashes or hangs) rather than the actual quality required by the application.

Finally, the high-level approximation model (SAM) and the Technique Specific Approximation Model (TSAM) are applied to characterize the resilience by using the actual quality function. The high-level approximation model is also termed as Statistical Approximation Model (SAM) because it injects errors based on the statistical (Gaussian) distribution. This defines a high-level approximation space of an application by providing a quality profile based on statistical parameters and can help to narrow down technique-specific approximation choices such as arithmetic operations, data representation and algorithm level approximations [26]; for instance, choosing M1 or M2 as discussed in the motivational example earlier.

2.4 APPROXIMATE COMPUTING TECHNIQUES

Approximate computing techniques can be broadly divided into three main categories, namely: software-level, architectural-level, and circuit-level techniques. In this section, we provide a brief survey of such techniques and their underlying concepts.

2.4.1 SOFTWARE LEVEL TECHNIQUES

These techniques tend to reduce the complexity of software to gain efficiency benefits such as reducing run-time of the target application. The prominent software-level techniques include code perforation, loop perforation and relaxed synchronization.

A simple form of a software-level approximate computing technique is *code perforation*, wherein error-resilient parts are automatically identified within the target code. These parts are then skipped during execution to save resource usage [45]. A relevant technique is to skip loop iterations selectively as the loops contribute largely to the overall resource usage [117]. Such a technique is known as *loop perforation*. For some applications like recognition and mining, synchronization is also an expensive part. Research in [79, 102] shows that a relaxed synchronization criterion can lead to resource savings while having a low impact on the output quality.

For machine learning classifiers, based on the observation that some instances are easier to classify than others, the work in [25, 129] demonstrates that utilizing different complexity classifiers for different instances provides resource reduction. For the HEVC video encoder application, Palomino et. al. demonstrate that adaptively varying the approximation levels—based on the video properties—can lead to an improved thermal profile of the application [92].

2.4.2 ARCHITECTURE LEVEL TECHNIQUES

At architecture-level, the approximate computing techniques mainly focus on memory and Input/Output (I/O) communication. One way to decrease mem-

ory latency and to increase the overall performance of an architecture is to avoid reading the memory while predicting its values. For instance, *load value approximation* (LVA) predicts data values on an event of a cache miss [109, 121]. LVA may result in a wrong value provided to the processor but reduces average memory access time by hiding cache miss latency. Meanwhile, the predictor can be trained by accessing the memory location.

Ranjan et al. introduced the accuracy configurable memory load and store mechanism for increased energy efficiency. Considering recognition and vision applications, the authors demonstrated a 19.5% energy reduction while introducing less than 0.5% of quality degradation [100]. In the case of Dynamic Random Access Memories (DRAMs), energy savings can be achieved by reducing the refresh rate required to maintain the data reliably. Flicker [72] introduces this scheme by dividing the memory into sensitive and error-resilient portions. They proposed to store critical data in the sensitive portion and the non-critical data in the error-resilient portion of the DRAM. The refresh rate of the sensitive portion was proposed to be unchanged while a reduction in the same was proposed for the resilient portion of the DRAM. Utilizing this scheme, the authors demonstrated up to 25% of power savings in a mobile device while running typical smart-phone applications like mpeg2, c4, rayshade, vpr and parser [72].

In computing architectures, communication between a source and a destination causes energy dissipation and latency. Sources and destinations can be multiple processors on a board, multiple cores in a chip, processors and caches or I/O devices. Based on the observation that the energy consumption is directly dependent on switching activity at wires, i.e., '1' to '0' and '0' to '1' transitions, the research in [103, 118] proposed a value-deviation-bounded serial (VDBS) communication scheme. As a case study, the VDBS scheme demonstrated the switching activity reduction by 54% for a pedometer, while limiting the step-count error within 5%. Moreover, in the case of a network-on-chip, APPROX-noc [17] introduces a data-type aware approximation utilizing approximate matching between recently sent data and the data to be sent. Based on approximate matching, the data is compressed to reduce the overall data communication. For a case study of the graph processing algorithm, the authors demonstrated a 36.7% of latency reduction as compared to the state-of-the-art compression schemes.

2.4.3 HARDWARE-/CIRCUIT-LEVEL TECHNIQUES

Hardware-/circuit-level techniques include transistor-level and gate-level techniques. Transistors are the basic building block of contemporary computing systems. Their organization forms analog circuits and gates. The organization of gates forms digital circuits. Therefore, the organization of transistors and gates can be altered to increase computing efficiency at the expense of quality. Voltage-overscaling and circuit-pruning are the major approximate computing techniques at the hardware level.

Voltage-overscaling

The quadratic relation of supply voltage (V_{dd}) with dynamic power consumption (Eq. 2.4) substantiates the viability for low voltage design to save energy. Let f and C be the frequency of operation and effective capacitance respectively. The dynamic power consumption P is given as [98],

$$P \propto C V_{dd}^2 f \quad (2.4)$$

Different levels of low-voltage designs have been explored in literature; namely: ultra-low or sub-threshold voltage, near-threshold voltage (NTV), and super-threshold or nominal-voltage range [30, 47]. Energy efficiency decreases and performance increases as we go from ultra-low voltage to nominal-voltage. Moreover, dynamic voltage and frequency scaling (DVFS) techniques have also been explored to achieve a better energy-delay product [112]. The design methodology for low-voltage design is to reduce the V_{dd} to a minimum viable/optimum level (V_{opt}) and compensate for the performance loss by exploiting parallelism within the applications while running the parallel threads on additional cores. V_{opt} depends on the technology size, frequency of operation and intrinsic characteristics of the target application like architectural and Amdahl's overhead [95]. A recent analysis of low-voltage designs [96] suggests that the NTV operation is a promising technique to attain higher energy benefits with reasonable performance for error-free computing.

Energy consumption can be saved further by operating a circuit at a lower voltage than is typically assumed safe for a given frequency of operation. As a consequence, this introduces circuit delays, and in turn, leads to timing errors. Such a computing technique is referred to as *voltage-overscaling* in literature. It has attracted researchers over the last decade to trade-off accuracy of results for increased energy gains. As an example, adaptive voltage over-scaling without error correction saves more power than its error-free low-voltage design equivalent [61]. Other examples of voltage-overscaling techniques/applications include adaptive quality tuning [55, 82], discrete cosine transform (DCT) [7, 43], motion estimation using error correction [125], and power-efficient static random access memories (SRAMs) [16, 23].

Nevertheless, the timing errors introduced by voltage-overscaling affect the critical paths of the circuit, which may also introduce errors in the higher significant bits of the computation. In [77] and [71], it is demonstrated that removing some parts of a circuit that have a low probability of usage or they contribute to the lower significant part of the circuit, can provide a better quality-efficiency trade-off as compared to the voltage-overscaling techniques. Such a technique is referred to as circuit-/logic-/gate-level pruning or simply *circuit pruning*.

Circuit Pruning

Pruning enhances the efficiency of circuits without introducing the overheads related to voltage-overscaling [71, 110, 116]. Here we discuss some of the promi-

nent approximate adder and multiplier circuits to show how pruning is applied for achieving high-efficiency gains.

Approximate Adder Circuits

Gupta et al. [39] presented transistor-level pruning for approximate full adder circuits, which can be utilized to design multi-bit adders. Considering a case study of digital signal processing applications (DCT and FIR filter), they demonstrated up to 69% power savings as compared to accurate adders. Moreover, the authors showed that the error-mean and error-variance of their approximate adder designs are smaller than that of the truncation approach, especially when the number of approximate (least significant) bits exceed 2.

Another technique for approximate adder circuits is the *carry propagation chain simplification* or reduction of the critical path to reach the overall sum of two inputs [32, 77, 130, 134]. Reducing the number of carry propagation bits also reduces power consumption related to the glitches produced during the carry propagation [134].

Consider for example the Error Tolerant Adder (ETA) [134], wherein an addition operation is divided into two parts, a Higher Significance Part (HSP) and a Lower Significance Part (LSP). The HSP produces the most significant bits of the result and the LSP produces the least significant bits of the result. The carry propagation path in the LSP is avoided by using half adders that do not produce a carry output. However, to reduce the overall error of the adder, the following scheme is utilized (shown in Fig. 2.4). The LSP adds bits from left to right (MSB to LSB), a normal addition of the bits is performed if both the input bits are different or 0. On the other hand, if both inputs are 1, the addition process is stopped and all the remaining LSB outputs are assigned to 1. In the addition operation shown in Fig. 2.4, half of the carry propagation is removed which results in reducing latency and power consumption of the adder circuit.

Another approach to reducing the overall latency of an adder circuit is to use a variable latency adder [130]. This adder utilizes an almost correct adder (ACA), which provides an addition output with a low error rate while reducing the latency of the circuit. An error correction circuit is also introduced that provides a correct result in a rare case of error occurrence. Although the latency of the error-corrected output is higher than that of the normal addition, the overall performance (considering a large number of additions) is increased because of the low error rate introduced by the ACA.

Similar to variable latency, the idea of variable accuracy has also been explored in literature, such as: accuracy configurable adder (AcCA¹), gracefully degrading adder (GDA), and generic accuracy configurable adder (GeAr) [53, 114, 133]. The AcCA provides a run-time knob to vary the accuracy of the circuit to

¹both designs (accuracy configurable adder [53] and almost correct adder [130]) have been abbreviated with ACA in literature. To make a clear distinction, we abbreviate accuracy configurable adder as AcCA in this thesis.

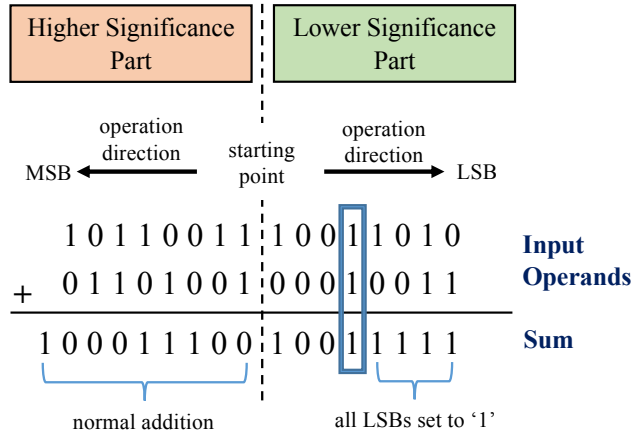


Figure 2.4: An illustration of ETA methodology using a 16-bit addition operation. The carry chain propagation is reduced by avoiding the carry outputs in the lower significance part of the adder [134].

improve latency and power consumption. Similarly, GeAr is an open-source adder model that utilizes different accuracy sub-adders to provide a wide-range of approximation levels, or in other words accuracy configurability, for an overall addition operation.

Approximate Multiplier Circuits

A multiplication operation includes generating partial products and adding them in a specific shift-order to achieve the overall product of two input numbers (operands). The approximation techniques for a multiplier include input truncation, partial product truncation, simplified addition of partial products, or simplified partial product generation.

Consider a 4×4 multiplication operation as shown in Fig. 2.5a, where two 4-bit numbers are multiplied to get an 8-bit product. One way to simplify the multiplication operation is to truncate the input operands for a specified number of bits. For example, 1-bit (LSB) truncation is employed in Fig. 2.5b. The consequence of 1-bit truncation is the removal of the corresponding partial products. On the other hand, in the case of the partial products truncation method, any partial product can be removed to simplify the multiplication operation., see Fig. 2.5c for an example of six partial products truncation.

A clever technique related to truncation is proposed by Hashemi et al [42]. They introduced a Dynamic Range Unbiased Multiplier (DRUM) that truncates the input operands to a smaller bit-width and then apply an accurate multiplier. As an example, they showed a 16-bit multiplication ($n = 16$) executed by a 6-bit multiplier ($k = 6$). However, the truncation of inputs is based on the sequence of 1s and 0s, which renders an overhead of detecting 1s in the sequence. The

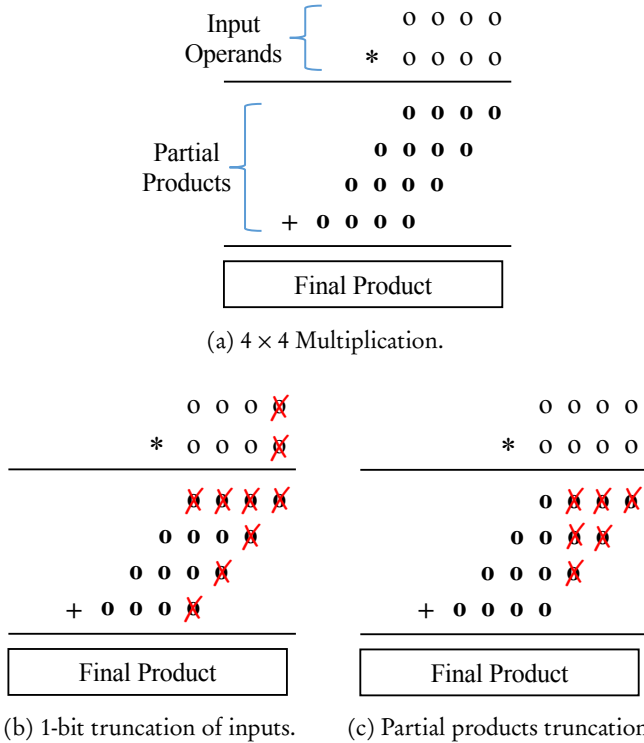


Figure 2.5: Illustration of a 4-bit multiplication using input truncation and partial products truncation methods; (a) accurate version; (b) 1-bit truncation of input operands; (c) truncating six partial products corresponding to the lower significance part of the product.

obtained $2k$ -bit product is shifted, depending upon the number of bits truncated, to obtain the final product. An advantage in their scheme is a potential un-biasing of the multiplier. A normal truncated multiplication always provides a negative bias, i.e., the approximated output is either less or equal to the accurate output. However, the proposed truncation in DRUM sets the least significant bits of the input operands to 1 (for approximate un-biasing), before applying a k bit multiply operation [42].

Another approximation approach is to use approximate Full Adders (FAs) while adding the partial products of a multiplier [39, 101]. In [101], several designs of a FA have been demonstrated that show a better efficiency as compared to the accurate FA. Two of such FA designs (FA_{x1} and FA_{x2}) along with the accurate FA are shown in Fig. 2.6. FA_{x1} provides better efficiency in terms of area, latency and power as compared to that of the accurate FA. On the other hand, it introduces two error cases out of eight possible cases, see Fig. 2.6d where

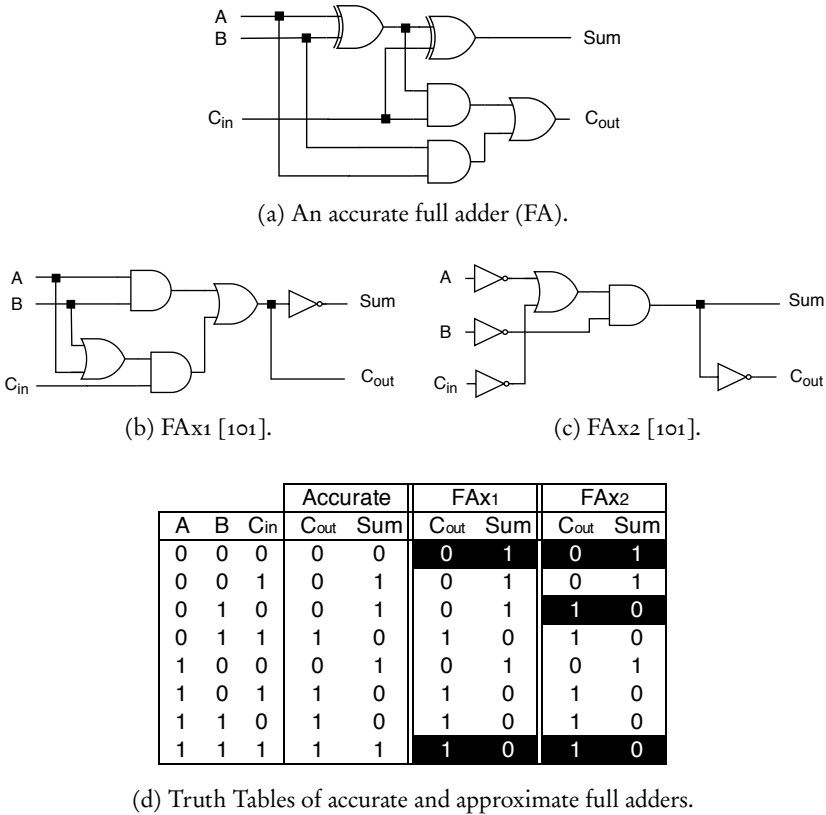


Figure 2.6: Full Adder (FA) circuits. FAx1 and FAx2 are approximate versions proposed in [101], which are more efficient (at the cost of a quality loss) as compared to that of the accurate counterpart.

the error cases are marked in black. FAx2 is even more efficient than that of FAx1 but brings an additional error case, see Fig. 2.6d. These FA designs can be utilized for adding partial products generated by the sub-multipliers (e.g., 2×2 sub-multipliers) to compute the overall result of a given $n \times n$ multiplier. Several promising FA and sub-multiplier designs are considered to explore the design space of a given $n \times n$ multiplier for achieving pareto-optimal designs [101].

Targeting power efficiency, gate-level pruning is applied to design a 2×2 multiplier in [63]. Such multipliers can be used recursively together with adder trees to form a higher-order ($n \times n$) multiplier. An advantage of such multipliers is that they are scalable and provide flexibility in the design process. For that matter, these multipliers can be optimized based on input distribution to achieve an effective quality-efficiency trade-off based on the target application [101], see Section 2.5 for details.

FPGA Based Optimization

Although circuit pruning brings remarkable area, power, and latency benefits for Application Specific Integrated Circuits (ASICs), it does not provide the same extent of benefits for Field Programmable Gate Arrays (FPGAs) because of their underlying architecture [32, 97]. For instance, if we want to implement a full adder in a Look-Up Table (LUT) based FPGA, where a LUT provides more than three inputs that are required for a full adder, the unused inputs (and/or other unused resources) do not allow achieving high-efficiency benefits. To this end, FPGA specific techniques have been presented in literature, e.g., [32, 97].

In the context of FPGAs, approximate circuits in general and adders, in particular, can utilize the unused resources to increase efficiency for a given quality constraint or to increase quality for given resource usage. In case of an implementation of a full-adder in a LUT based FPGA, the unused inputs can be utilized to share information between the lower significant part and the higher significant part of the adder to improve the quality of the output without increasing the resource usage [32]. Moreover, DeMAS presents a generic methodology for designing approximate adders based on the resources and architectural features of the target FPGA [97].

Automatic Generation of Approximate Circuits

Approximate circuits have been discussed above, which provide a means to trade accuracy of results for improved hardware efficiency. However, utilizing these approximate circuits in designing relatively larger accelerators with an optimal quality-efficiency trade-off is challenging. To this end, automatic tools have been presented in literature that generate approximate circuits based on their behavioral specifications, i.e., register-transfer level (RTL) descriptions [87, 127]. For example, systematic logic synthesis of approximate circuits (SALSA) automates the approximate design process by considering the quality-constraint and RTL descriptions of a circuit as input and providing an ASIC net-list of an optimal approximate circuit as an output [127].

When the approximate combinational circuits are utilized to design a sequential circuit, the output quality is required to be assessed after multiple cycles of execution instead of a single computational cycle [99]. Therefore, approximate sequential circuits pose a challenge to model the generation and propagation of errors introduced by the embedded approximate combinational blocks to multiple cycles. To this end, a methodology for automatic sequential logic approximation (ASLAN) [99] has been proposed by Ranjan et. al. ASLAN provides a systematic framework that takes the sequential circuit and the related quality-constraint as inputs and generates an energy-efficient approximate circuit as an output.

A recent automation approach introduces approximations at the gate-level standard cells (ASIC library) and employs such cells to automatically generate pareto-optimal approximate circuits for a given quality constraint [27]. In the context of high level synthesis, approximate high-level synthesis (AHLS) has been in-

roduced for the direct conversion of a high-level C code (accurate) to an RTL description of an optimal approximate circuit [67, 70].

2.5 APPROXIMATE RECURSIVE MULTIPLIERS

Approximate recursive multipliers prune the recursive multiplication structure to reduce the number of gates and the critical path of the circuit. Such multipliers are (especially) known for their power efficiency benefits [63]. Moreover, such multiplier structures are scalable and provide a huge number of possible approximation choices that help in their optimization process based on the input distribution [75, 101]. In the context of providing guarantees for approximate multiplication, the worst-case error of approximate recursive multipliers has been demonstrated to be better than the worst-case error of some of the other approximate approaches like truncation [85]. Keeping in view the benefits of the approximate recursive multipliers, we have utilized them to demonstrate some of our approximate computing methodologies (see Chapter 4 and Chapter 5). Therefore, in this section, we provide the basic knowledge to help the readability of the later chapters.

An $n \times n$ recursive multiplier is constructed using four $n/2 \times n/2$ sub-multipliers, where n is the bit-width of input operands, $n \in \{4, 8, 16, 32, \dots\}$. For example, a 4×4 multiplier contains four 2×2 multipliers. Fig. 2.7 illustrates the concept of recursive multipliers using an example of an 8×8 multiplier. Considering a 2×2 multiplier as an elementary module/multiplier, an $n \times n$ multiplier requires $(n/2)^2$ of them. These 2×2 multipliers generate partial products. Summation of the bit-shifted partial products produces the overall output of an $n \times n$ recursive multiplier.

Any number out of the set of 2×2 multipliers and/or adders (used for the summation of the partial products) can be approximated to achieve an approximate $n \times n$ multiplier [63, 101]. Kulkarni et al. [63] introduced an approximate 2×2 multiplier shown as M_1 in Fig. 2.8d. The truth table of M_1 shows an approximation of one out of the sixteen possible outputs, see Fig. 2.8c. Therefore, the error rate of M_1 is $1/16$, where the magnitude of the error is 2.

The motivation behind this approximation is the reduction of output-bits that are to be computed, i.e., from four to three. The overall effect is that the critical path and number of gates of the circuit have been reduced. This brings an increase in computing efficiency in terms of power, chip-area, and performance. Utilizing accurate 2×2 multipliers (M) and their proposed (M_1) multipliers, the authors constructed higher-order multipliers (4×4 , 8×8 , and 16×16) and showed power savings of 32%–45% with an average error of 1.4%–3.3%. While assessing the mean error vs power reduction trade-offs, they draw two important conclusions [63, 64]: (1) The quality-efficiency trade-off of the multiplier designs utilizing M_1 is more effective than that of the conventional truncation technique. (2) The quality-efficiency trade-off of the multiplier designs utilizing M_1 and accurate

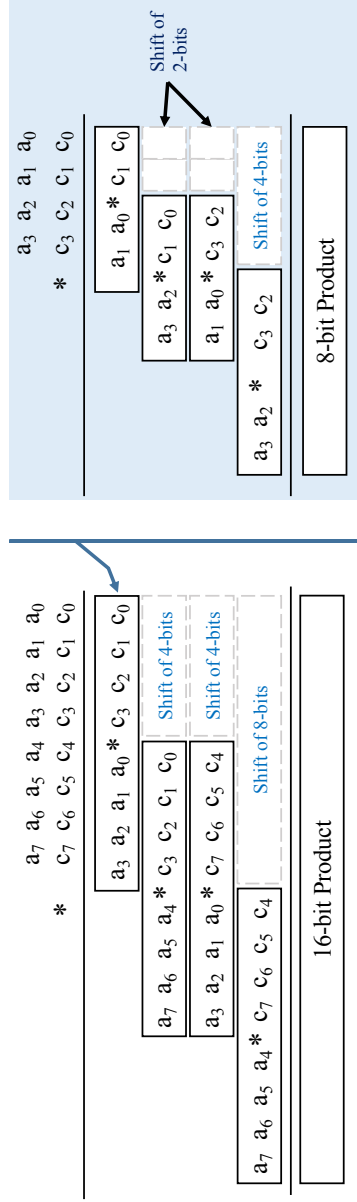
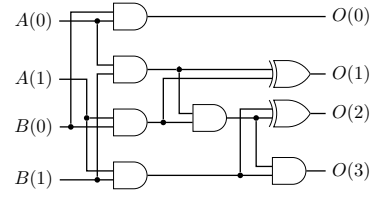


Figure 2.7: An example of a recursive multiplication (a and c are 8-bit operands). An 8×8 recursive multiplier utilizes four 4×4 sub-multipliers (left), where each 4×4 multiplier requires four 2×2 sub-multipliers (right).

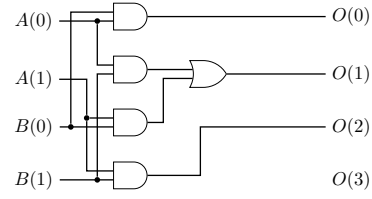
$B \backslash A$	00	01	10	11
00	0000	0000	0000	0000
01	0000	0001	0010	0011
10	0000	0010	0100	0110
11	0000	0011	0110	1001

(a) Truth table of accurate multiplier (M).

(b) M: $3 * 3 \mapsto 9$.

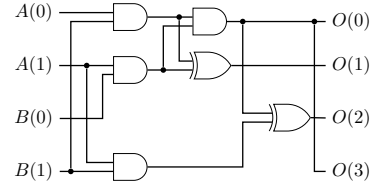
$B \backslash A$	00	01	10	11
00	0000	0000	0000	0000
01	0000	0001	0010	0011
10	0000	0010	0100	0110
11	0000	0011	0110	0111

(c) Truth table of M1 [63].

(d) M1: $3 * 3 \mapsto 7$ [63].

$B \backslash A$	00	01	10	11
00	0000	0000	0000	0000
01	0000	0000	0010	0010
10	0000	0010	0100	0110
11	0000	0010	0110	1001

(e) Truth table of M2 [101].



(f) M2 [101].

Figure 2.8: 2×2 Multiplier designs. M is an accurate version, M1 [63] and M2 [101] are approximate versions that provide a higher power efficiency as compared to that of M. However, M1 and M2 produce errors, as shown in (c) and (e).

adders (for summation of partial products) is more effective than using accurate 2×2 multipliers (M) and approximate adders.

An even more power-efficient approximate 2×2 multiplier has been proposed in [101]. Building on the observation that the LSB and the MSB of the product are equal for thirteen out of the total sixteen cases, see Fig. 2.8a, the LSB and MSB have been equated. Therefore, producing an error rate of $3/16$. Fig. 2.8e and Fig. 2.8f show the truth table and the circuit design of the aforesaid multiplier, shown as M2. Although M2 has a higher error rate as compared to that of M1, its error magnitude (based on each error case) is less than that of M1. In [101], it is demonstrated that M2 requires 28% less power as compared to M1. Secondly, when M2 is utilized in a circuit where error detection and correction is supported, the only requirement is to invert the LSB to get the correct output. On the other hand, however, its latency and area consumption are higher than

that of M1.

2.6 EVALUATION

In this chapter, we have discussed stochastic computing, probabilistic computing, and approximate computing that trade determinism and/or accuracy of computation with efficiency gains. In the context of approximate computing, we have discussed various techniques that span from software-level to hardware/circuit-level abstraction layers. In this thesis, we focus on hardware efficiency and propose methodologies for high-efficiency accelerator designs for iterative and accumulation based algorithms. Therefore, here we evaluate the related techniques to clarify their context with the rest of the thesis.

As discussed in Section 2.1, stochastic computing provides efficiency benefits due to its low-complexity arithmetic operations. However, its disadvantages include slow-speed and low-precision operations [4]. Secondly, probabilistic computing offers energy efficiency due to its low-voltage operation. On the other hand, it is noted that the (deterministic) circuit pruning technique brings a more effective quality-efficiency trade-off as compared to probabilistic computing, especially for the algorithms that are not probabilistic in nature [71, 110]. Similarly, voltage-overscaling also proposes a low-voltage operation to offer efficiency benefits. Again, circuit pruning brings a more effective quality-efficiency trade-off as it does not introduce overheads related to voltage-overscaling, and it does not (generally) affect the critical paths or most significant parts of the computation [71, 77, 110, 116].

In view of the above, we utilize circuit pruning techniques to demonstrate our proposed approximate computing methodologies for high-efficiency accelerator designs. In Chapter 3, we utilize a truncated multiplier concept based on input truncation to design a heterogeneous accelerator for iterative algorithms. In Chapter 4 and Chapter 5 we utilize approximate recursive multiplier concepts to present our approximate computing methodologies for accumulation based algorithms.

3

EXPLOITING ERROR RESILIENCE OF ITERATIVE ALGORITHMS

ABSTRACT – Iterative algorithms are widely used in digital signal processing applications. With the case study of radio astronomy calibration processing, this chapter contributes towards revealing and exploiting the intrinsic error resilience of iterative algorithms for efficiency benefits. Our contributions for improving the error resilience analysis are focused primarily on iterative methods that use a convergence criterion as a quality metric to terminate the iterative computations. We propose an adaptive statistical approximation model for high-level resilience analysis that provides an opportunity to divide an iterative algorithm into exact and approximate iterations. We realize an energy-efficient accelerator based on a heterogeneous architecture, where the heterogeneity is introduced using accurate and approximate processing cores. Our proposed methodology exploits the error-resilience of the algorithm, where initial iterations are processed on approximate modules while the later ones on accurate modules. The proposed accelerator design does not increase the number of iterations as compared to that of an accurate counterpart and provides sufficient precision to converge to an acceptable solution.

Assessing error resilience inherent to an algorithm provides application-specific insights towards approximate computing strategies. The error resilience analysis is performed by applying approximations while monitoring the quality function to verify that the approximations produce acceptable results. In this way, approximate computing techniques are substantiated for an algorithm that can be employed in the implementation phase to achieve the desired benefits.

With the advent of accuracy-configurable architectures like adaptive voltage over-scaling [61] and adaptive processors comprised of configurable adder/multiplier

This chapter is based on [G:1] and [G:2].

blocks [115], where the quality-cost trade-off can be controlled at run time, it is of remarkable importance to analyse an algorithm for Adaptive Error Resilience (AER) as well. AER has the potential to reveal approximation opportunities even in strict quality function (relatively less error-resilient) algorithms, where they can be processed adaptively for varying approximation levels to gain hardware efficiency benefits.

Moreover, heterogeneous architectures have the ability to handle various workloads/algorithms efficiently while using different power and performance trade-off computing nodes, e.g., ARM's big.LITTLE architecture [54]. In the context of approximate computing, the definition of a heterogeneous architecture extends further to include exact and inexact computing units, where control instructions and sensitive computational parts run at precise cores while the error-resilient parts run at the error-prone cores to achieve the overall efficiency in speed and energy [56, 108].

Iterative methods are common candidates of approximate computing like K-means [76], GLVQ training [26], and model predictive control [104]. Therefore, we propose an Adaptive Statistical Approximation Model (Adaptive-SAM) to perform the high-level error resilience analysis of iterative algorithms. The high-level error resilience analysis refers to applying the statistical error based on a Gaussian distribution and assessing the output-quality for a relaxed quality function, see Section 2.3.4 for details. Adaptive-SAM performs better than the Statistical Approximation Model (SAM) [26] by quantifying the number of approximate iterations (N_{ax}) in addition to the statistical parameters: EM, EP, and ER (see Section 2.3.4). Therefore, it can better exploit the heterogeneous/accuracy-configurable architectures by assigning resilient iterations to the approximate computing cores/modes and the sensitive iterations to the exact counterparts.

Being pivotal building blocks of DSP architectures, approximate multipliers and adders have been extensively researched for increased hardware efficiency [42, 50, 77, 85, 94, 97]. However, approximate accelerator designs for relatively bigger algorithms have been of less attention yet. The Least Squares (LS) algorithm is widely utilized in digital signal processing applications like image reconstruction in radio astronomy [86, 107], medical [89], and synthetic aperture radar [20] domains. Despite its importance, no approximate least squares accelerator design has been investigated to the best of our knowledge.

As discussed in Chapter 1, modern radio telescopes like Square Kilometer Array (SKA) [51] require highly energy-efficient processing architectures to process terabytes of raw data per second. For instance, double-precision fused multiply-add operations will require 7.2MW of power consumption in the medium-frequency array of SKA if contemporary¹ technology would be used [51]. In this regard, we investigate an energy-efficient LS accelerator architecture based on a case study of radio astronomy calibration processing. The aforesaid processing employs an

¹This estimation is based on the technology in 2014.

iterative LS algorithm to compute sensors' gains for a certain configuration of a radio telescope.

The primary contribution of this work is an effective way of exploiting the error resilience of iterative algorithms for achieving energy efficiency benefits. In this regard, the following is presented in this chapter,

- » An adaptive statistical approximation model (Adaptive-SAM) for error resilience analysis of iterative algorithms (Section 3.2.1).
- » Error resilience analysis of a radio astronomy calibration algorithm (case study) by performing the state-of-the-art statistical approximation model (SAM) analysis and our proposed Adaptive-SAM analysis (Section 3.2.2).
- » An assessment of utilizing convergence criterion as a quality metric for the error resilience analysis of iterative algorithms (Section 3.2.3). We demonstrate the importance of quality function reconsideration for convergence based iterative processes as the original quality function (the convergence criterion) is not necessarily sufficient in the error resilience analysis phase.
- » An energy-efficient heterogeneous architecture for iterative algorithms with a case study of an approximate least squares (LS) accelerator design for radio astronomy calibration processing (Section 3.3).

3.1 RELATED WORK

This section reviews some state-of-the-art approximate computing architectures that motivate statistical and adaptive-statistical error resilience analysis. Moreover, we discuss contemporary analysis methodologies/tools and the need of Adaptive-SAM analysis.

3.1.1 ADAPTIVE ACCURACY TECHNIQUES

Approximate computing utilizing adaptive accuracy techniques have been discussed in [115] and [61]. The adaptive voltage over-scaling (AVOS) [61] has shown improvements in power efficiency (25% to 30%) at a negligible quality loss for texture decompression application. The aforesaid scheme reduces the supply voltage until a specific number of errors are introduced and can increase the voltage again to attain error-free operation. Therefore, AVOS can adjust the quality-cost trade-off during run time.

Nevertheless, as discussed in Chapter 2, circuit pruning based schemes outperform voltage-overscaling based schemes. Therefore, the idea of accuracy-configurable architectures composed of approximate accelerators has been proposed in [115]. These architectures contain accuracy-configurable operators, such as multipliers and adders, which can change the computation mode from accurate to approximate and vice-versa during the run-time. This helps in run-time

adjustment of the quality-cost trade-off based on the algorithm's error resilience. In this context, we argue that the aforementioned adaptive accuracy architectures can be better exploited provided that the error resilience profile of an iterative algorithm quantifies the number of approximate iterations in addition to the insights of promising approximations. This allows an adaptive employment of approximations during the run-time to gain target benefits.

3.1.2 ERROR RESILIENCE ANALYSIS TECHNIQUES

Based on literature study, various error resilience analysis techniques have been discussed in Chapter 2 (Section 2.3). Here we summarize them in the context of related work to our proposed technique.

Quality of service (QoS) profiling utilized loop perforation as a compiler pass in order to identify sub-computations that can be replaced with less accurate counterparts [78]. Intel's open-source approximate computing toolkit (iACT) assesses the scope of approximations within the applications using programmer annotated pragmas to analyse the programmer guided parts of the code [80]. iACT has the ability to apply static approximate transformations such as precision scaling and run-time approximations like memoization² during the error resilience analysis.

Automatic sensitivity analysis for approximate computing (ASAC) applies perturbation of program data to study the overall effects on the output quality [105]. Program analysis for approximation-aware compilation (PAC) provides a relatively faster way to study the accuracy requirement of each component in an algorithm [106]. Approximate C compiler for energy and performance trade-offs (ACCEPT) is another open-source tool that applies a conservative approach to perform safe approximate relaxation analysis within an algorithm [108]. The aforesaid tools are limited in assessing the error-resilience of an algorithm as they do not cover all the approximation strategies and they do not provide a statistical error resilience profile to reduce the available design space (alternatives) as discussed in Section 2.3.2.

On the other hand, the application resilience characterization (ARC) framework [26] includes a statistically distributed error injection model to generate the statistical profile of an algorithm. The aforesaid model is known as Statistical Approximation Model (SAM) and is utilized to perform the so-called high-level error resilience analysis. However, in order to better utilize the adaptive accuracy architectures, we need Adaptive-SAM analysis of iterative algorithms that can quantify the adaptive resilience by identifying the number of approximate iterations in addition to the statistical parameters.

²See Section 2.3.2.

3.2 ERROR RESILIENCE ANALYSIS OF ITERATIVE ALGORITHMS

In this section, we elaborate on the Adaptive-SAM analysis methodology and present its significance with a case study of the radio astronomy calibration application. In practice, the error resilience of an application is quantified by injecting the errors defined by the approximation models (statistical or technique-specific) and monitoring the overall output of the application in compliance with the quality function. The range of error injection within an approximation model for which the quality function is satisfied can be regarded as the *approximation space* of an application. Therefore, defining a quality function is a very deliberate task in the approximate computing domain. In Section 3.2.3, we demonstrate that the original (precision-based) quality function of an iterative process is not necessarily sufficient in the error resilience analysis procedure, which requires defining an additional (accuracy-based) quality function to serve the purpose.

3.2.1 ADAPTIVE STATISTICAL APPROXIMATION MODEL (ADAPTIVE-SAM)

As discussed earlier, our aim is to improve the high-level error resilience analysis of iterative algorithms to better exploit accuracy configurable and heterogeneous architectures. In this regard, we present Adaptive-SAM that can replace SAM in the error resilience analysis methodology to provide the number of approximate iterations (N_{ax}) in addition to statistical parameters of the approximation space (EM, EP and ER).

To elaborate on the Adaptive-SAM methodology, consider a template of an iterative algorithm shown in Algorithm 1. The algorithm has inputs (x_1, x_2, x_3, \dots) and an output (K_op). It iterates to improve the output by utilizing the inputs and previously computed result (last iteration), where i is the current iteration and N is the maximum number of iterations. The algorithm also uses an intermediate variable (im_var) that is computed by calling a kernel: function_im, having input arguments as x_1, x_2, x_3, \dots and the computed output of the previous iteration K_op($i - 1$). Then the output is computed by calling another kernel: function_Kop that has input arguments x_1, x_2, x_3, \dots and im_var.

Subsequently, the convergence metric (convergence_met) is computed by calling function_conv kernel that considers the current output K_op(i) and the previously computed output K_op($i - 1$). Iterative algorithms may use different arithmetic operations within function_conv, but the aim is generally to compute the improvement in result within two consecutive iterations. Finally, Algorithm 1 checks the convergence metric for the allowed tolerance limit (tol) based on the quality function of the iterative algorithm. If the convergence is reached, the iterative process is terminated to provide the final outcome.

As discussed in Section 2.3.4, the error resilience methodology identifies the dominant kernels in the first place. These kernels are selected based on the percentage of their run-time or floating point operations (FLOPs) relative to the overall algorithm. The kernels that have the higher share are regarded as

Algorithm 1 An Iterative Algorithm Template.

Input: x_1, x_2, x_3, \dots **Output:** K_op

```

1: Initialize  $K\_op(0)$ 
2: for  $i = 1, 2, \dots, N$  do
3:    $im\_var = function\_im(x_1, x_2, x_3, \dots, K\_op(i - 1))$ ;
4:    $K\_op(i) = function\_Kop(im\_var, x_1, x_2, x_3, \dots)$ ;
5:    $convergence\_met = function\_conv(K\_op(i), K\_op(i - 1))$ ;
6:   if ( $convergence\_met \leq tol$ ) then
7:     break;    // convergence reached
8:   end if
9: end for

```

dominant kernels as it is likely to attain desired benefits (area, power or latency) while approximating them. In Algorithm 1, we assume that `function_Kop` is a dominant kernel to explain Adaptive-SAM analysis.

Fig. 3.1 shows the signal flow of Adaptive-SAM. We assume N number of iterations where i is the current iteration ($1 \leq i \leq N$). The output of iteration i of the dominant kernel ($K_op(i)$) is added to an error (ϵ_i) if the randomized error (`ER_rand(i)`) and approximate iterations flag (`ax_iter_flag`) allow error injection to this stage. This flag is controlled via a parameter: N_{ax} , which is the number of initial iterations to be instrumented with errors. It is important to note that ϵ_i is a Gaussian randomness that is generated in a relative manner, i.e., relative to the kernel output ($K_op(i)$). This ensures that unsuitably big/small magnitudes of errors are not inserted into the kernel output. Also note that ϵ_g is a function of the EP and EM as given in Eq. (3.1),

$$\epsilon_g = \frac{(EP \times randn(i) + EM)}{100} \quad (3.1)$$

where `randn(i)` generates a random number with the standard normal distribution ($\mu = 0$ and $\sigma = 1$). As an example, let $EM = 10$ and $EP = 0.2$, the probability is approximately >99.7% that ϵ_i lies between 9% to 11% of the $K_op(i)$. The approximate kernel output is assessed for quality function compliance to test the validity of the approximation space. Therefore, Adaptive-SAM quantifies the high-level error resilience of an application based on the acceptable range of a Gaussian randomness for the quantified approximate iterations. This helps to assign the number of approximate iterations to the inexact cores (fast and/or energy efficient) with the specified approximation space while running the exact iterations on the exact cores to better exploit the heterogeneous architectures. Similarly, in case of accuracy configurable architectures, the approximate iterations can run in approximate (error prone) hardware modes at higher energy efficiency and/or speed.

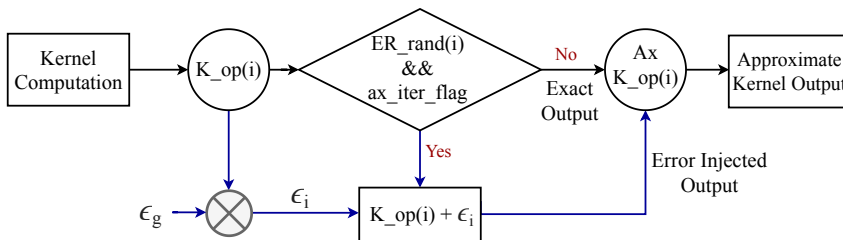


Figure 3.1: Proposed high-level error resilience analysis model – Adaptive-SAM.

3.2.2 HIGH-LEVEL ERROR RESILIENCE ANALYSIS

We have applied SAM and Adaptive-SAM on an iterative algorithm, which is the radio astronomy calibration algorithm (StEFCal) [107]. The said algorithm has been instrumented with the run-time statistical error injection in Matlab. The quality function compliance has also been checked at run-time to determine the statistical error tolerance limits. In this way, we have quantified the high-level error resilience for StEFCal by using SAM and Adaptive-SAM models.

Radio Astronomy Calibration Processing

In radio telescopes, gain calibration improves the quality of sky images and reinforces the signal processing techniques against interference and spatial effects [15]. Calibration processing computes complex antenna gains in a radio telescope. The gains are estimated by minimizing the following [107],

$$\|\mathbf{V} - \mathbf{G}\mathbf{M}\mathbf{G}^H\|_F^2 \quad (3.2)$$

where \mathbf{M} and \mathbf{V} represent the model and the measured visibilities, respectively ($\mathbf{M}, \mathbf{V} \in \mathbb{C}$). $\mathbf{G} = \text{diag}(\mathbf{g})$ represents complex antenna gains ($\mathbf{G} \in \mathbb{C}$). In this work, we assume 124 antennas. Therefore, the gains are 124 complex numbers, so the matrix \mathbf{G} has a size of 124×124 with gains on the diagonal and zeros at other entries.

The calibration algorithm, also known as StEFCal (statistically efficient and fast calibration) [107], is a strict quality-of-service method that estimates complex antenna gains (g_p) for the P sensors in a radio telescope. The algorithm computes a \mathbf{g} vector (representing P gains) based on a measured signal/array covariance matrix (\mathbf{V}) and the model covariance matrix (\mathbf{M}), where each iteration (i) computes P independent linear least squares problems,

$$g_p^{[i]} = \frac{\mathbf{V}_{:,p}^H \cdot \mathbf{Z}_{:,p}^{[i-1]}}{(\mathbf{Z}_{:,p}^{[i-1]})^H \cdot \mathbf{Z}_{:,p}^{[i-1]}} \quad (3.3)$$

where $\mathbf{V}_{:,p}^H$ is the hermitian transpose of array covariance matrix's p th column. $\mathbf{Z}_{:,p}^{[i-1]}$ is the element-wise product of $\mathbf{g}^{[i-1]}$ and the model covariance matrix's

p th column ($\mathbf{M}_{:,p}$),

$$\mathbf{Z}_{:,p}^{[i-1]} = (\mathbf{M}_{:,p} \odot \mathbf{g}^{[i-1]}) \quad (3.4)$$

It should be noted that $\mathbf{g}^{[i-1]}$ is the antenna gains vector computed in the previous iteration. In our experiments, representative input data (\mathbf{V} and \mathbf{M} matrices) of the LOFAR facility [124] has been utilized (for $P = 124$).

The convergence criterion of StEFCal is based on the relative length of the difference of consecutive iterations' solution vectors in the Euclidean space [107],

$$\text{Convergence} = \frac{\|\mathbf{g}^{[i]} - \mathbf{g}^{[i-1]}\|_F}{\|\mathbf{g}^{[i]}\|_F} \leq 1.10^{-6} \quad (3.5)$$

In our initial experiments, we defined our quality function solely based on the convergence criterion. However, this proved to be insufficient in the error resilience analysis process as will be explained in Section 3.2.3. Therefore, we have defined an additional quality metric: Diff_rel, which is the relative difference in length between the exact (ex) and approximate (ax) solution vectors,

$$\text{Diff_rel} = \frac{\|\mathbf{g}_{ex}^{[i]} - \mathbf{g}_{ax}^{[i]}\|_F}{\|\mathbf{g}_{ex}^{[i]}\|_F} \leq 1.10^{-5} \quad (3.6)$$

We assume that the *quality acceptance* is achieved, if and only if both the convergence (Eq. 3.5) and Diff_rel (Eq. 3.6) criteria are satisfied.

Simulation Results of SAM and Adaptive-SAM

To apply SAM and Adaptive-SAM, the initial steps are the same as elaborated in Section 2.3.4, i.e., distinguishing dominant kernels that run at least 1% of the total execution time and identifying error resilience by injecting the random errors in the outputs of the dominant kernels. Three dominant kernels have been identified in StEFCal by using profiling [40]. One of them is \mathbf{Z} computation (Eq. 3.4), which brings 27% of the computational load. The other two are the dot products (Eq. 3.3), which bring 72% of the computational load. We have analysed the aforesaid three kernels for high-level error resilience. As the response of the dot products is almost similar to \mathbf{Z} computation, we only present the simulation results for \mathbf{Z} computation here. They are sufficient to demonstrate the comparison between SAM and Adaptive-SAM outcomes.

Fig. 3.2 shows an illustrative example of StEFCal response for SAM analysis. The figure presents the effect of error mean (EM), error predictability (EP) and error rate (ER) on the considered quality metric: Diff_rel (Eq. 3.6). It can be seen that Diff_rel is increased as we increase the EM and ER, while it has no remarkable effect due to changes in EP. However, the convergence limit is only achieved at minimum EP and maximum ER, see Fig. 3.3. The iteration count

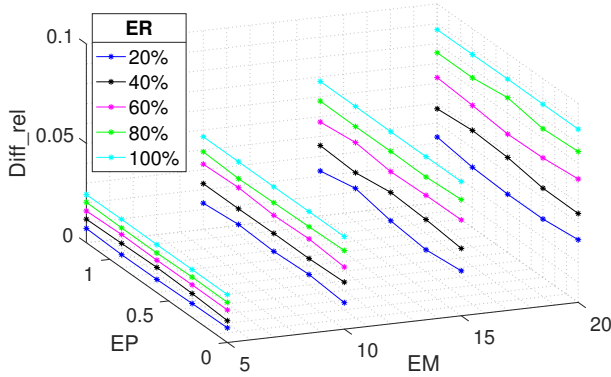


Figure 3.2: StEFCal response for SAM analysis.

in Fig. 3.3 is based on even³ iterations, which means that the total iterations are two times the number of iterations shown in the figure. It is to be noted that the convergence criterion (Eq. 3.5) does not allow a random selection of iterations for error injection because this reduces the precision of the computation among consecutive iterations. Therefore, our SAM analysis shows that the convergence is achieved either at ER=0% (no error injection at all), or at ER=100% (error injection for all iterations without random skipping) with suitable EM and EP values, see Fig. 3.3. Likewise, high values of EP also reduce the precision of computation among consecutive iterations, and therefore, do not satisfy the convergence criterion.

In practice, there can be some cases where the benefits of energy efficiency/performance are achieved even if the approximate computation runs for more iterations than the exact computation [104]. However, for the sake of simplicity, we assume that the convergence limit is satisfied when the number of iterations required to converge for error-injected computation is less or equal to that of the exact computing counterpart. To quantify the error resilience intrinsic to StEFCal based on the SAM analysis, extensive simulations have been performed by varying EM, EP, and ER values. The *quality acceptance* has been achieved for $EM \leq 0.002$, $EP \leq 2.10^{-4}$ at ER=100%. This implies that the algorithm is resilient to an error (Gaussian randomness) that has the following maximum values: $EM = 0.002$, $EP = 2.10^{-4}$, and that the error is employed in every iteration. This shows a very small approximation space that does not appreciate employing approximate computing techniques to gain efficiency benefits.

An illustrative example of Adaptive-SAM analysis of StEFCal is shown in Fig. 3.4. In this case, the simulations are performed for various number of approximate

³At every even iteration in StEFCal, the gain solution is replaced by the average of the previous (odd) iteration and the current (even) iteration to help fast convergence [107].

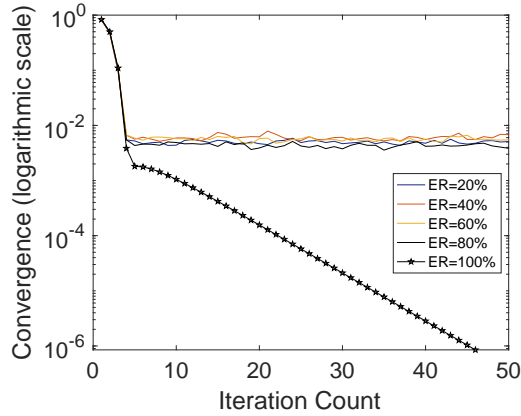
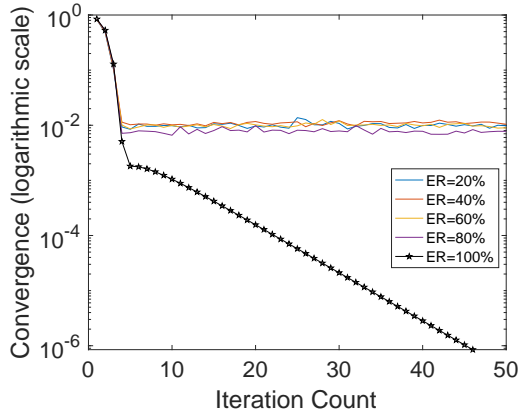
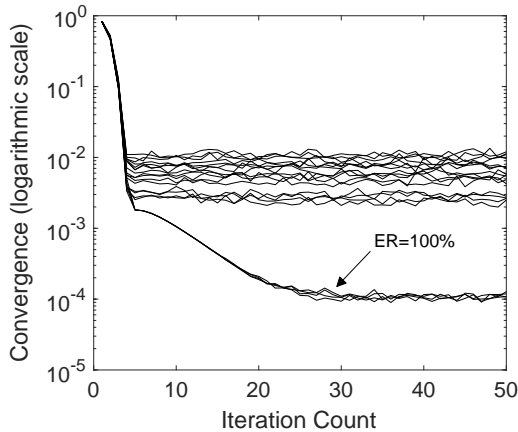
(a) $EP=0$ and $EM=10$.(b) $EP=0$ and $EM=20$.(c) $EP=0.1$, $EM=5 : 5 : 20$, $ER=20 : 20 : 100$

Figure 3.3: Convergence (logarithmic scale) w.r.t the number of iterations; the algorithm converges at $ER=100\%$ when $EP=0$ (a) and (b). The algorithm does not converge when EP is raised to 0.1 (c). For the simulations regarding $EP=0.1$ (c), EM has been varied from 5 to 20 with a step size of 5 and ER has been varied from 20% to 100% with a step size of 20% .

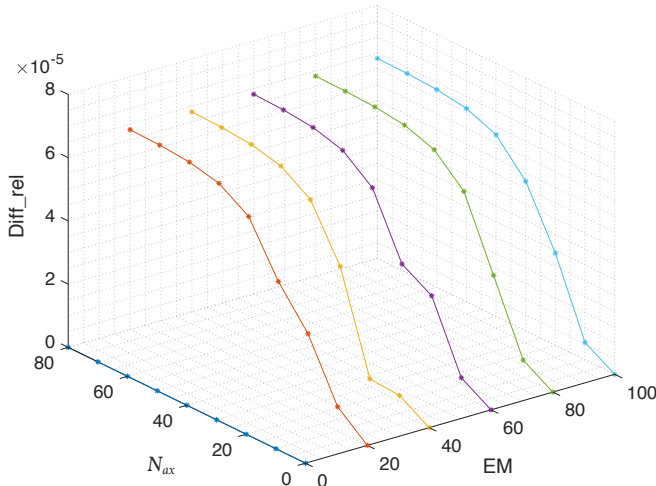


Figure 3.4: StEFCal response for Adaptive-SAM analysis.

iterations (N_{ax}) and error mean (EM) values. However, error predictability and error rate are fixed to the values that allow the solution to converge ($EP=0$ and $ER=100\%$). This means that in every approximate iteration an error of $EM\%$ of the output value is injected. As expected, the $Diff_rel$ decreases with the decrease in N_{ax} and EM, see Fig 3.4. This suggests that the higher the number of exact computing iterations, the better the quality of output and vice versa.

To quantify the error resilience intrinsic to StEFCal based on the Adaptive-SAM analysis, extensive simulations have been performed by varying N_{ax} , EM, EP, and ER values. The *quality acceptance* is achieved for $N_{ax} \leq 23\%$, $EM \leq 12$, $EP \leq 0.2$ at $ER = 100\%$. This implies that if the initial (up to) 23% of iterations are approximated, the algorithm is resilient to an error (Gaussian randomness) that has the following maximum values: $EM=12$, $EP=0.2$, and that the error is employed in all the initial (23%) iterations. This shows the availability of an approximation space for up to 23% of the initial iterations. Therefore, Adaptive-SAM reveals additional error resilience opportunities by quantifying the number of approximate iterations in addition to EM, EP and ER for iterative algorithms.

3.2.3 SIGNIFICANCE OF QUALITY FUNCTION RECONSIDERATION

During the error resilience analysis, the statistical and technique-specific approximation models are applied offline⁴ and are validated using the quality function⁵. This makes the selection of a quality function very crucial as it decides whether

⁴Error resilience analysis is performed before the deployment of the application for real use.

⁵The quality function is either given within an application/algorithm or has to be defined for error resilience analysis.

to reject or accept an approximation technique. Therefore, in order to utilize the original quality metric of an iterative application (convergence criterion) in the approximate computing domain, it has to be reconsidered rigorously to attain reliable insights about the approximation space.

In our case study of an iterative algorithm (StEFCal), the convergence criterion is utilized for the exact computing case. It computes the relative distance in Euclidean space between the current and previous solution vectors. The convergence is assumed to be satisfied if the improvement within two consecutive iterations is less or equal to 1.10^{-6} , which means that the solution has already been converged and no further precision can be achieved by computing more iterations. However, in the error resilience analysis process, it cannot be guaranteed that the acceptable solution is achieved when it satisfies convergence. Perhaps, the solution is converged in the wrong plane, which is very distant from that of acceptable solution plane.

We have observed this phenomenon during the error-resilience identification phase. As discussed in Section 3.1, random errors are injected into the resilience identification analysis while using a relaxed quality function. Two cases are shown in Fig. 3.5 to illustrate the problem. The first case is randomly skipping computations (Fig. 3.5a and Fig. 3.5b), while the second case is an arbitrary error injection in the first element of each column of the \mathbf{Z} , i.e., $\mathbf{Z}_{:,p}$ (Fig. 3.5c and Fig. 3.5d).

For the first case, we can see a comparable response of approximate and exact computing for gains outputs, albeit with no quality improvement beyond a specific number of iterations for the convergence metric. However in the second case, although the approximate solution converges quickly (Fig. 3.5c), it produces unacceptable gains (Fig. 3.5d). This shows that the original quality metric of StEFCal (the convergence criterion) is not sufficient in the resilience analysis process. For that matter, we introduced an additional quality parameter: Diff_rel (Eq. 3.6) that can ensure the convergence of an approximate solution within an acceptable distance to that of an exact solution in the Euclidean space. Therefore in Section 3.2.2, we assume that the *quality acceptance* is achieved if and only if both the convergence and Diff_rel criteria are satisfied.

3.3 ENERGY EFFICIENT ACCELERATOR DESIGN FOR ITERATIVE ALGORITHMS

So far we have discussed that the statistical approximation models can be applied to evaluate a target algorithm for error resilience. These models inject errors during the execution of the algorithm on statistical bases to quantify the bearable error profile. For iterative applications, our analysis suggests that a certain number of initial iterations can be approximated while producing an acceptable outcome. Therefore, in this section, we present our hardware realization of an accelerator that utilizes a heterogeneous architecture composed of two process-

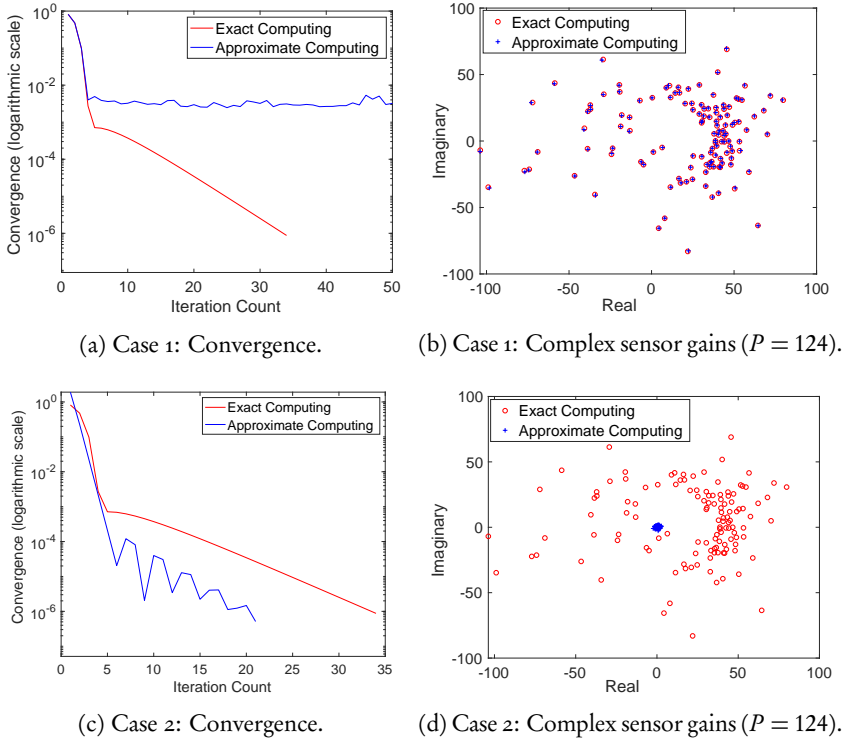


Figure 3.5: Error resilience identification; convergence (logarithmic scale) w.r.t the number of iterations (a) and (c); complex sensor gains for $P = 124$ (b) and (d). Case 1 and Case 2 represent a random skipping of a computation and an arbitrary error injection, respectively, in each iteration. Similar to Fig. 3.3 the iteration count is based on even iterations.

ing cores. The two cores differ in their precision of computation, namely an *accurate core* and an *approximate core*. We show how a set of initial iterations can be processed in an approximate core, while the rest of the iterations are processed in an accurate core to achieve an overall energy-efficiency increase.

3.3.1 DESIGN OF A HETEROGENEOUS LEAST SQUARES ACCELERATOR

Our design methodology for an approximate Least Squares (LS) accelerator is shown in Fig. 3.6. The accelerator architecture is composed of two cores that differ in computation precision, introducing heterogeneity in the architecture. The accurate core is optimized for the required precision for the LS algorithm. However, the approximate core introduces a reduced-precision computation to provide energy efficiency. In the proposed LS accelerator, the initial iterations are run on the approximate core, while the rest of the iterations on the accurate

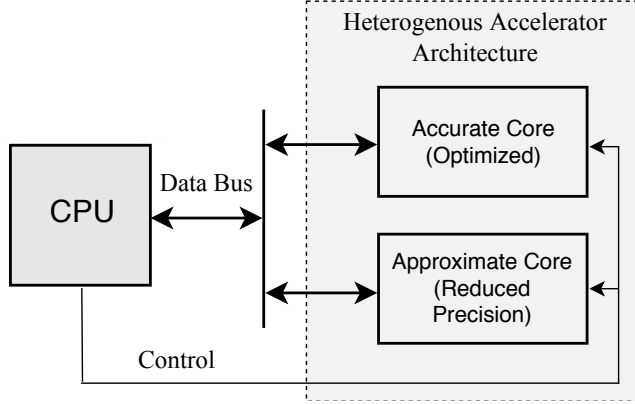


Figure 3.6: Our design methodology for an approximate Least Squares (LS) accelerator enables initial iterations to be processed on an approximate core (while the rest on an accurate core) to achieve an overall energy-efficiency.

core. This brings an overall energy efficiency when a central processing unit (CPU) switches off the unused core.

Nevertheless, using two cores instead of one brings area overhead. However, if the CPU can utilize both cores simultaneously for parallel processing of independent processes, this area overhead can be translated into increased throughput. In any case, the energy efficiency can be increased for processing the LS algorithm with or without area penalty.

Overall Energy Savings

Consider an accurate LS accelerator that utilizes only an accurate core. Let E_a be the total energy consumption of an accurate LS accelerator for processing N_{acc} iterations to solve a given LS problem. Therefore,

$$E_a = E_{acc} \times N_{acc} \quad (3.7)$$

where E_{acc} is the energy consumption of the accurate core for processing one iteration. Now consider our proposed LS accelerator design that processes N_{ax} iterations utilizing the approximate core while the rest ($N_{acc} - N_{ax}$) is utilizing the accurate core. The energy consumption (E_b) of such an accelerator is given as,

$$E_b = E_{ax} \times N_{ax} + E_{acc} \times (N_{acc} - N_{ax}) \quad (3.8)$$

where E_{ax} refers to the energy consumption of the approximate core for one iteration. For our proposed architecture, we assume that the total number of iterations (running on accurate and approximate cores) remain the same as N_{acc} .

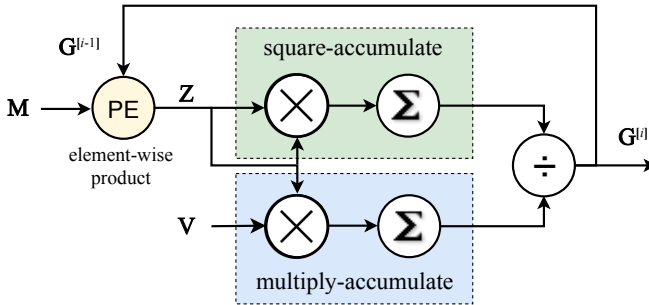


Figure 3.7: Least Squares (LS) algorithm for radio astronomy calibration processing.

The overall energy savings (S_E) while utilizing our proposed accelerator design can be given as,

$$S_E = \frac{(E_a - E_b)}{E_a} \quad (3.9)$$

Using Eq. 3.7 and Eq. 3.8, S_E is given as,

$$S_E = \frac{(E_{acc} - E_{ax}) \times N_{ax}}{E_{acc} \times N_{acc}} \quad (3.10)$$

E_{acc} and N_{acc} are constant terms as they correspond to the reference accurate architecture. On the other hand, reducing E_{ax} or increasing N_{ax} would increase energy benefits (S_E). However, lowering E_{ax} means introducing more coarse approximations in the approximate core and this results in practice in a decrease in the number of iterations (N_{ax}) that can survive this approximation level. Therefore, there is a trade-off between E_{ax} and N_{ax} and the goal is to find an optimal balance where S_E is maximized.

Radio Astronomy Calibration Algorithm (StEFCal)

We consider a case study of radio astronomy calibration processing that employs a Least Squares (LS) algorithm, StEFCal, as discussed in Section 3.2.2. We demonstrate how to design an LS accelerator using the proposed methodology, wherein the accurate LS core and the approximate LS core are optimized to achieve energy-efficient LS processing. We consider complex data as utilized by StEFCal.

Fig. 3.7 shows the four stages of the algorithm. The PE block computes the element-wise product. Square-accumulate (SAC) computes the inner product of \mathbf{Z} with itself. Multiply-accumulate (MAC) computes the inner product of \mathbf{V} and \mathbf{Z} . Finally, a division operation is performed to compute the gain values. Considering complex inputs, Fig. 3.8 illustrates the signal flow of the algorithm.

Noteworthy, four multiplications are required (along with an addition and a subtraction) to multiply two complex numbers, i.e.,

$$(a + ib)(c + id) = ac - bd + i(ad + bc) \quad (3.11)$$

and two squaring operations are required (along with an addition) to find the square of a complex number, i.e.,

$$(e + if)(e + if) = e^2 + f^2 \quad (3.12)$$

It is to be noted that m , v , z and g correspond to the respective elements of \mathbf{M} , \mathbf{V} , \mathbf{Z} , and \mathbf{G} matrices.

3.3.2 EXPERIMENTAL RESULTS

Here we compare the accurate and the proposed LS accelerator designs as discussed in Section 3.3.1. We assume an equal frequency of operation for both designs, which means an equal processing time for executing a single iteration. Therefore, Eq. (3.10) is reduced to power (P) consumption values only, as in Eq. (3.13).

$$S_E = \frac{(E_{acc} - E_{ax}) \times N_{ax}}{E_{acc} \times N_{acc}} = \frac{(P_{acc} - P_{ax}) \times N_{ax}}{P_{acc} \times N_{acc}} \quad (3.13)$$

Power consumption and chip-area estimates have been obtained by synthesizing the designs in Synopsys ASIC flow (Design Compiler and Power Compiler) for the TSMC 40nm Low Power (TCBN40LP) technology library at 50MHz. Our experimental setup is shown in Fig. 3.9. Questasim has been utilized to verify the functionality of the synthesized designs (gate-level netlists) and to generate the related SAIF (Switching Activity Interchange Format) files based on the respective standard delay file (.sdf) and test data. The aforesaid SAIF files and gate-level-netlists are utilized by Synopsys Power Compiler for power estimation.

Output-quality assessment has been performed in Matlab, where the radio astronomy calibration was performed. Here, the Test Data (see Fig. 3.9) refers to the radio astronomy calibration data of the LOFAR facility [124]. Similar to Section 3.2.2, it is assumed that the *quality acceptance* is achieved, if and only if both the convergence (Eq. 3.5) and the Diff_rel (Eq. 3.14) criteria are satisfied.

$$\text{Diff_rel} = \frac{\|\mathbf{g}_{\text{float}} - \mathbf{g}_{\text{fixed}}\|_F}{\|\mathbf{g}_{\text{float}}\|_F} \leq 10^{-5} \quad (3.14)$$

where $\mathbf{g}_{\text{float}}$ and $\mathbf{g}_{\text{fixed}}$ refer to the gains obtained using double-precision floating-point and fixed-point computations, respectively.

We assume the accurate core as the optimized fixed-point design of StEFCal [12, 60], wherein the word length of each signal (shown in Fig. 3.8) remains

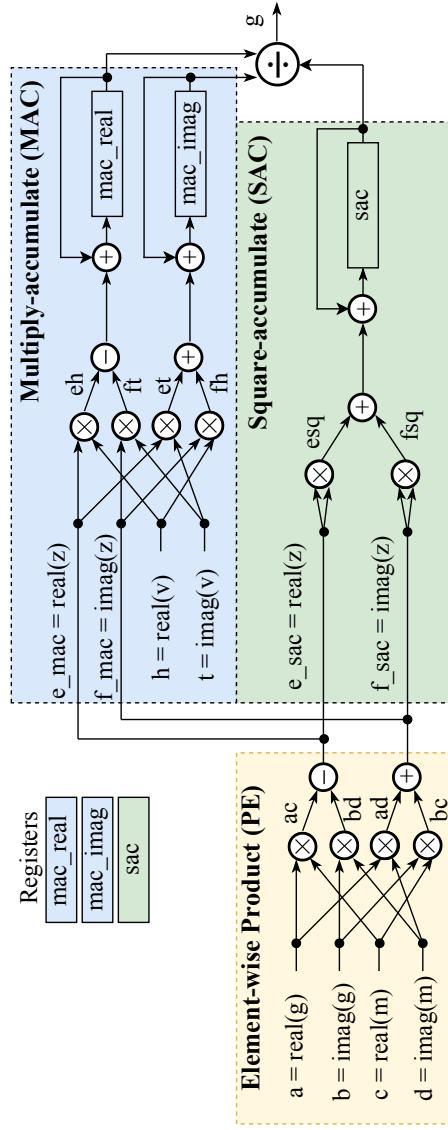


Figure 3.8: Signal flow of least squares algorithm in radio astronomy calibration processing.

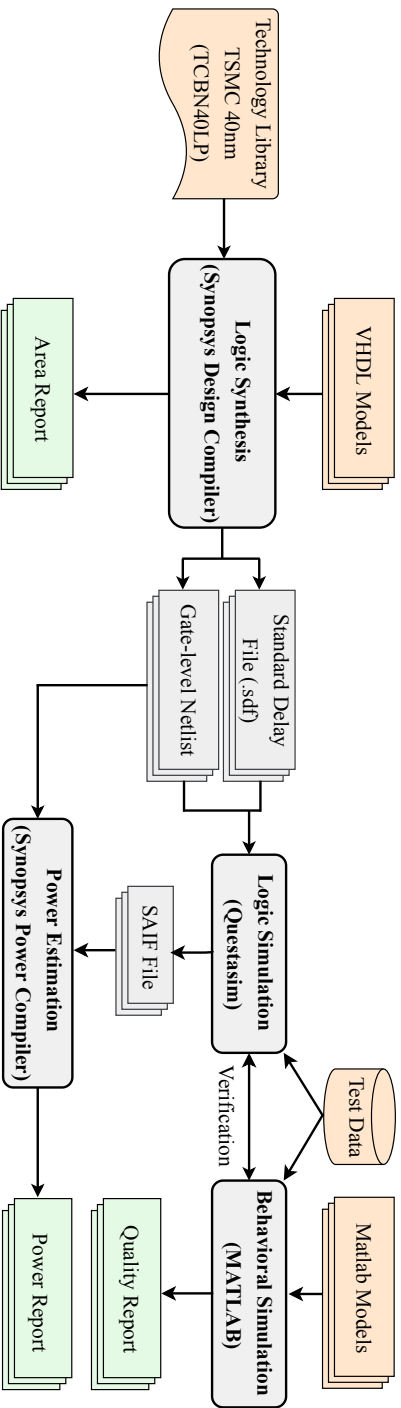


Figure 3.9: Our experimental setup to assess chip-area and power consumption of the considered designs along with their output-quality.

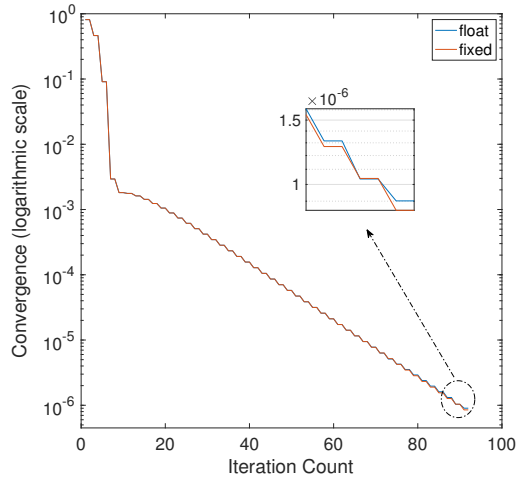
Table 3.1: Bit widths of the StEFCal algorithm for accurate LS core and the truncation levels to achieve an optimal approximate LS core.

Signal name	Accurate Core (bit-width)	Approximate Core (bits truncated)
h	18	0
t	18	0
e_sac	21	8
f_sac	20	8
e_mac	23	8
f_mac	24	12

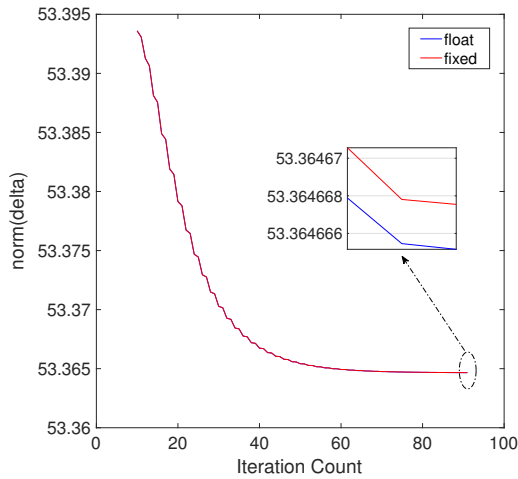
less or equal to 28 bits. Fig. 3.10 shows the comparison between the optimized fixed-point (fixed) and double-precision floating point (float) computation of the StEFCal algorithm. As shown in Fig. 3.10a, both computations converge in 92 iterations. Fig. 3.10b illustrates the behavior of $\text{norm}(\delta) = \|\mathbf{V} - \mathbf{GMG}^H\|_F$, which shows that the minimization of the difference is also achieved for the accurate core (fixed). The gain values are shown in Fig. 3.11. As the gain values of the accurate core also satisfy Eq. 3.14, *quality acceptance* is achieved for the accurate core.

As discussed in Section 3.3.1, our proposed LS accelerator design utilizes an approximate core additionally and the overall energy savings can be maximized by optimizing the approximate core in such a way that Eq. 3.13 is maximized. This means that a design space exploration is to be carried out to find a trade-off point where P_{ax} and N_{ax} values maximize Eq. 3.13. We consider truncation of inputs as a means of approximation in the approximate core, wherein four multipliers of the MAC and two squarers of the SAC have been approximated (see Fig. 3.8). Based on our design space exploration, the number of truncated bits to obtain the optimized approximate core are shown in Table 3.1, where the resulted $N_{ax} = 52$. The corresponding input widths of the accurate core are also shown as a reference.

Fig. 3.12 shows the output-quality comparison of our proposed methodology with that of the accurate core. The *quality acceptance* is achieved as we process the first 52 iterations on the approximate core, and the rest of the iterations (40) on the accurate core. After switching to the accurate core, two phenomena can be noticed: (1) the precision-oriented metric, i.e., convergence, experiences jumps because of the increase (change) in computation precision, see Fig. 3.12-left. (2) the deviation from the accurate solution gradually decreases, see Fig. 3.12-right. Overall, the solution converges to an acceptable value in the same number of iterations, i.e., 92.



(a)



(b)

Figure 3.10: Comparison between the double-precision floating-point (float) and optimized fixed-point (fixed) StEFCal processing, the latter is referred to as the *accurate core*.

Energy Efficiency Improvements

Table 3.2 shows the chip-area and power consumption of the accurate and approximate cores. It can be seen that the approximate core offers 41% power reduction as compared to that of accurate core ($P_{ax} = 2.08mW$ and $P_{acc} = 3.55mW$). As

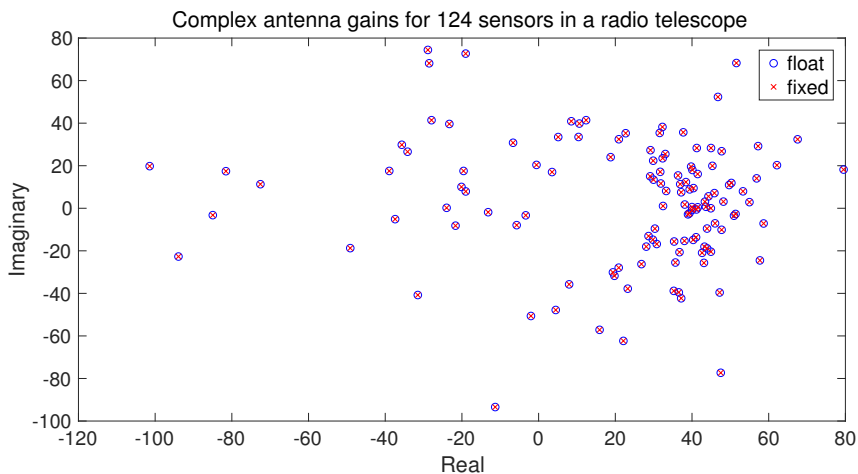


Figure 3.11: Gains computed by double-precision floating-point (float) and optimized fixed-point (fixed) processing.

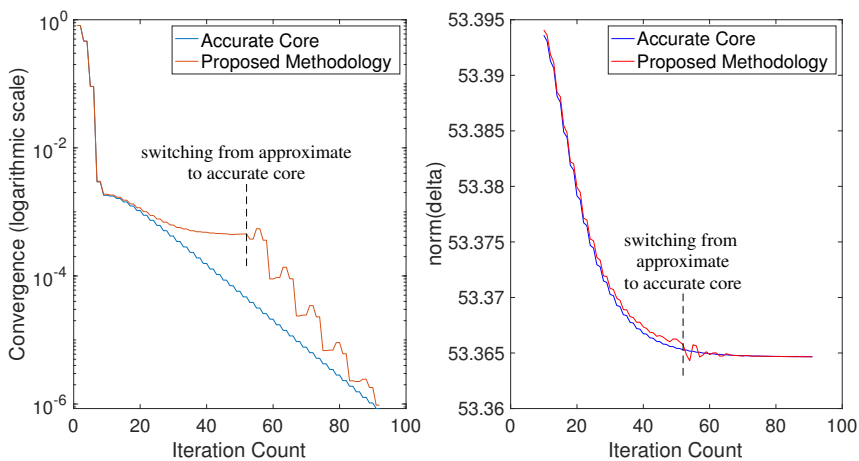


Figure 3.12: StEFCal processing based on our proposed heterogeneous methodology as compared to that of an accurate design. Our proposed methodology processes the first 52 iterations on the approximate core and the rest on the accurate core.

$N_{ax} = 52$ and $N_{acc} = 92$, the overall energy saving using Eq. 3.13 for our proposed LS accelerator design is 23.4%. The aforesaid energy saving is obtained while comparing the accurate single-core design with our proposed two-core design assuming that only one core is switched on at a certain period of time.

Table 3.2: Area and Power comparison of the accurate and approximate LS cores for TSMC 40nm Low Power (TCBN40LP) technology.

LS Core Type	Area (μm^2)	Power (mW)
Accurate Core	27023	3.55
Approximate Core	20604	2.08

Discussion and Future work

While comparing the chip-area of our proposed accelerator design (a two-core architecture) with that of an accurate accelerator design (a single-core architecture), the area overhead is 76%. This area overhead is due to the addition of the approximate core. However, if a CPU (see Fig. 3.6) can utilize both cores simultaneously for independent processes, the area overhead can be translated into throughput increase while having the same energy efficiency benefits for each process.

A case study of radio astronomy calibration (StEFCal) processing has been discussed to show how to employ the proposed heterogeneous accelerator by using a single time slot of LOFAR [124] data. Nevertheless, an increased data set would better provide the allowable number of iterations to run on an approximate core, therefore, a better estimate of energy savings that can be achieved using the proposed accelerator architecture.

3.4 CONCLUSIONS

The proposed adaptive statistical approximation model (Adaptive-SAM) has shown improvements in the error resilience analysis of iterative algorithms. It quantifies the number of resilient iterations in addition to statistical analysis parameters. Moreover, we have shown that the quality function must be reconsidered in the error resilience analysis process as the original quality metric of an iterative algorithm (convergence criterion) might not be necessarily sufficient. In which case, an additional quality metric has to be defined for reliable quality assessment to establish the promising approximate computing strategies.

A heterogeneous architecture for Least Squares (LS) acceleration has been presented targeting energy-efficiency. We have shown how a combination of optimized-precision (accurate) and reduced-precision (approximate) computing cores can be utilized to provide acceptable quality output while reducing energy consumption as compared to that of an accurate optimized architecture. Our design methodology exploits the inherent error-resilience of an iterative workload to leverage an approximate computing core for processing the initial iterations of the LS algorithm. A case study of radio astronomy calibration processing has shown 23.4% of energy savings as compared to that of the accurate counterpart. However, it is to be noted that the proposed methodology is independent of the application, provided that the computation pattern is iterative in nature.

We have utilized input truncation as the means of approximations within an approximate core. However, our methodology can utilize any approximation technique that is promising for the target application.

4

ERROR CANCELLATION IN ACCUMULATION BASED APPROXIMATE ACCELERATORS

ABSTRACT – The conventional approximate computing methodology restricts the introduction of errors to avoid a high loss in quality. However, this limits the computing efficiency and the number of pareto-optimal design alternatives for a quality-efficiency trade-off. This chapter presents a Self-Healing (SH) methodology for an accumulation based approximate accelerator, namely: Square-Accumulate (SAC). SAC refers to a hardware architecture that computes the inner product of a vector with itself. SH exploits the algorithmic error resilience of the SAC structure to ensure an effective quality-efficiency trade-off, wherein the squarer is regarded as an approximation stage and the accumulator as a healing stage. We propose to deploy an approximate squarer mirror pair, such that the error introduced by one approximate squarer mirrors the error introduced by the other, i.e., the errors generated by the approximate squarers are approximately additive inverse of each other. This helps the healing stage (accumulator) to automatically average out the error originated in the approximation stage, and thereby to minimize the quality loss. Our experiments corroborate that the proposed SH methodology provides a more effective quality-efficiency trade-off as compared to the conventional approximate computing methodology.

Approximate computing strives to achieve the highest performance, area-, and power-/energy-efficiency for a given quality constraint and vice versa. The state-of-the-art approximate design methodologies suggest utilizing *fail-rare*, *fail-small*, and *fail-moderate* approaches, where the approximations are restricted in terms of introducing errors and thereby limiting the computing efficiency. The fail-rare

strategy suggests that an approximation technique should result in a low error rate but may exhibit high error magnitudes [26]. On the other hand, fail-small refers to introducing low error magnitudes with possibly high error rates [26]. Fail-moderate suggests to utilize an additional design space of approximations, wherein the errors introduced may also exhibit moderate error rates and moderate error magnitudes [88]. It is important to note that the aforesaid strategies limit the design space as they do not allow approximations that introduce high error rates with high error magnitudes. The reason is obvious that this threatens the quality hugely in case of general algorithms, and if employed naively.

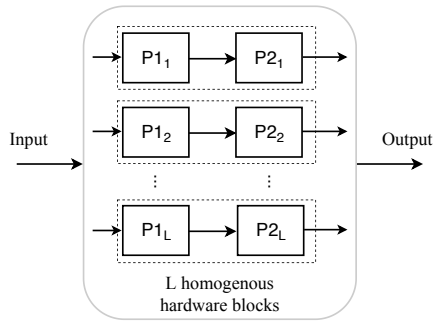
However, for accumulation based algorithms like square-accumulate (SAC) and multiply-accumulate (MAC), the approximations introducing simultaneous high error magnitudes and high error rates can also be utilized provided that the errors originated in various sub-components are potentially canceled out. We refer to this as a *fail-balanced* approach that helps to minimize the overall quality loss while achieving a high computing efficiency. This also increases the design space by introducing a high number of pareto-optimal approximate design alternatives to help effectively exploit the quality-efficiency trade-off.

This chapter presents a Self-Healing¹ (SH) methodology that exploits the fail-balanced approach for an accumulation based approximate accelerator. Specifically, the analysis and design of an approximate square-accumulate (SAC) architecture is discussed. SAC refers to a hardware architecture that computes the inner product of a vector with itself. Therefore, it is a special case of a multiply-accumulate (MAC), where both inputs of the multiplier are equal. It is one of the computationally expensive components of the Least Squares (LS) algorithm in general and radio astronomy calibration [107] in particular. LS is also employed in other Digital Signal Processing (DSP) applications like medical [89], synthetic aperture radar [20] and radioastronomical [86, 107] image-reconstruction.

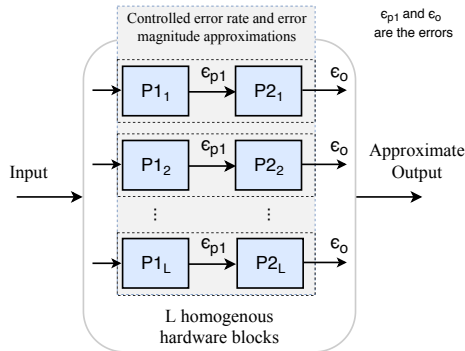
Consider an example of a parallel computing architecture, Fig. 4.1a, where the input stream is processed by L homogeneous (identical) hardware blocks to generate the output stream. Each hardware block consists of two processing elements, P₁ and P₂. The processing elements can be considered as arithmetic elements like multipliers and adders. The conventional way of approximating such an architecture is to employ approximate circuits for P₁ and P₂ in such a way that the error magnitudes and error rates are restricted to avoid an unacceptable loss in quality as shown in Fig. 4.1b.

The primary contribution of this chapter is a *Self-Healing* (SH) methodology that aims to utilize a fail-balanced approach wherein an architecture is divided into two types of stages, namely an *approximation stage* and a *healing stage*. Approximations are employed at the approximation stage in such a way that the

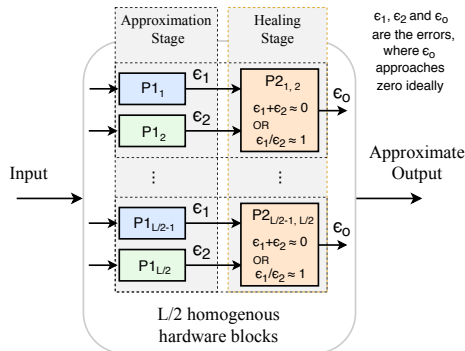
¹In this thesis, the term *self-healing* differs from that of [59]. In [59], in-circuit error detection and correction are targeted. However, in our self-healing methodology, we introduce approximations in a circuit in such a way that their effects are canceled out to minimize the quality-degradation.



(a) An example of a parallel computing architecture.



(b) Conventional approximate computing methodology.



(c) Proposed approximate computing methodology.

Figure 4.1: An overview of the conventional and the proposed approximate computing methodologies for parallel architectures. The proposed Self-Healing (SH) methodology does not restrict the approximations based on an error profile but provides the opportunity for error cancellation to achieve an effective quality-efficiency trade-off.

resulting errors have the potential to be healed up at the healing stage. Therefore, we propose to approximate P1 elements in pairs, such that each P1 in a pair generates an error that is the *mirror* (approximately additive or multiplicative inverse) of the other, see Fig. 4.1c. This minimizes the output error ϵ_0 to zero in some cases and to a lower value (as compared to the conventional methodology) in other cases and provides an effective quality-efficiency trade-off. The hardware cost of a pair is considered to be twice as that of a single P1 element, but the overall hardware cost of a parallel architecture remains the same as we require L/2 hardware blocks instead of L for the same throughput. To elaborate on the SH concept, this chapter presents the following:

- » Related terminology and application of SH for an approximate square-accumulate (SAC) design (Section 4.2).
- » Analysis of an approximate SAC design utilizing a truncated squarer that establishes the foundation for the SH methodology (Section 4.3).
- » An $n \times n$ approximate-squarer (AxSq) *mirror pair*, wherein the error introduced by one AxSq (ϵ_{s_1}) is an additive inverse of the error introduced by the other (ϵ_{s_2}), i.e., $\epsilon_{s_1} = -\epsilon_{s_2}$ (Section 4.4).
- » Design methodology for building an optimal SAC accelerator that utilizes the pareto-optimal AxSq *mirror pairs* (Section 4.5).
- » Quality-efficiency trade-off comparison of the proposed self-healing and conventional methodologies for random input data and radio astronomy calibration processing (Sections 4.5 and 4.6).

4.1 RELATED WORK

Approximate designs for adders [6, 38, 39, 53, 74, 97, 114, 130] and multipliers [11, 48, 49, 63, 65, 69, 75, 83, 101, 120, 123] have been researched for their indispensable role in Digital Signal Processing (DSP). Kulkarni *et al.* presented a low power approximate 2×2 multiplier and showed its efficacy in constructing higher-order ($n \times n$) multipliers, which can trade a bearable quality loss with improved computing efficiency [63]. In Fig. 4.2, the approximate 2×2 multiplier is shown as M1. It outputs only one error case (3×3 becomes 7 instead of 9) out of sixteen possible cases. M1 is referred to as a conventional approximate design in this chapter because it follows the fail-rare technique.

Another approximate 2×2 multiplier (M2) is discussed in Chapter 2. M2 follows the conventional fail-small technique as it brings three error cases with a small error magnitude (error magnitude= 1). In this chapter, we will not use the M2 design to demonstrate and compare our proposed fail-balanced technique. However, this design will be used in Chapter 5 for comparison purposes. To achieve pareto-optimality, a design space exploration of $n \times n$ approximate multipliers is presented in [101], which considers various 2×2 approximate multipliers to search for an optimized design.

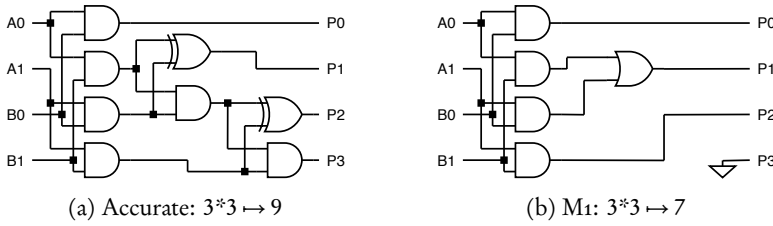


Figure 4.2: Logic diagrams of 2×2 multipliers; (a) accurate design; (b) an approximate design M_1 [63].

Deploying truncated multiplication in a MAC architecture attracted researchers in the last decade, where the objective was to limit the bit-width of multipliers and lower the error introduced due to truncations [62, 94]. Recent design approaches for approximate MAC present the introduction of an offset to compensate the inaccuracies of the approximate multiplier stage [35] and the utilization of hybrid redundant adders [31]. The aforesaid works presented techniques to approximate the MAC architecture. However, no exploitation of the self-healing approach has been studied to the best of our knowledge. Moreover, despite the importance of the SAC architecture in DSP, approximate SAC designs have not been researched yet.

The key limitation of conventional approximate computing techniques deals with its restricted design space, which does not allow employing approximate design alternatives (circuit-, architectural-, algorithm-level) that produce high error rates and high error magnitudes simultaneously. This restriction on design alternatives limits the achievable computing efficiency gains, hindering the exploitation of the quality-efficiency trade-off effectively.

4.2 SELF-HEALING METHODOLOGY FOR APPROXIMATE SQUARE-ACCUMULATE (SAC)

This section explains the Self-Healing (SH) concept by discussing its utilization for an approximate SAC architecture, wherein the squarer stage is regarded as *approximation stage*, and the accumulator as a *healing stage*. First, we define the terminology related to SH that will be used in the subsequent sections.

4.2.1 TERMINOLOGY

We define the *Mirror Error Effect* (MEE) as an introduction of errors (ϵ_1, ϵ_2) in a pair of approximate components that has the potential of cancellation (completely or partially) at a subsequent healing stage. For instance, ϵ_1 and ϵ_2 have opposite signs (healing stage: adder) or a common factor (healing stage: divider). Such a pair of components is (proposed to be) called *absolute approximate mirror*

pair provided that $\epsilon_1 = -\epsilon_2$ or $\epsilon_1/\epsilon_2 = 1$, otherwise *approximate mirror pair* where the errors are canceled out partially.

It is to be noted that the word "absolute" is used in the term *absolute approximate mirror pair* to provide a notion that the errors are canceled out completely instead of partially. Moreover, each module (approximate component) in an *absolute approximate mirror pair* is referred to as an *absolute approximate mirror* of the other.

In case of the SAC architecture where an accumulator is regarded as a healing stage, a pair of approximate multipliers complying to the MEE is referred to as an *Absolute Approximate Multiplier Mirror Pair* (AAMMP) if it exhibits $\epsilon_1 = -\epsilon_2$, otherwise it is called an *approximate multiplier mirror pair*. Similarly, a pair of approximate squarers complying to the MEE is referred to as an *Absolute Approximate Squarer Mirror Pair* (AASMP) if it exhibits $\epsilon_1 = -\epsilon_2$, otherwise it is called an *approximate squarer mirror pair*.

4.2.2 EMPLOYING SELF-HEALING FOR APPROXIMATE SAC ARCHITECTURE

With reference to Fig. 4.1a, consider a parallel architecture that computes a Square-Accumulate (SAC) operation where each hardware block is referred to as a SAC architecture. Now P1 can be regarded as a squarer and P2 as an accumulator, illustrated in Fig. 4.3a. Let \mathbf{a} be an input vector of length N and A_i be the i^{th} element of \mathbf{a} . The SAC operation computes:

$$\sum_{i=1}^N (A_i * A_i) \quad (4.1)$$

we consider an even number of elements in the input vector to ease the discussion, i.e., $N \in 2\mathbb{Z}_{>0}$. To design an approximate SAC in a self-healing fashion, we propose to approximate the squarer by deploying a pair of approximate squarers (e.g., AASMP) that introduce errors, $+\delta$ and $-\delta$ at Sq1 and Sq2 respectively, and utilize an accurate accumulator to cancel out the errors introduced in the squarer stage (Fig. 4.3b).

Eq. (4.1) can be re-written as,

$$\sum_{j=1}^{N/2} \{(A_{2j-1} * A_{2j-1}) + (A_{2j} * A_{2j})\} \quad (4.2)$$

Eq. (4.2) shows a pair of squarers that can be regarded as Sq1 and Sq2 as illustrated in Fig. 4.3b. Given the same input distribution for Sq1 and Sq2 (ideal case), the error at the approximate SAC output will approach zero for an infinite number of inputs. However, for non-ideal (real-world) cases, the error is minimized due to partial cancellation. It can be noted that the proposed approximate SAC hardware block (Fig. 4.3b) doubles the circuit area as compared to a reference

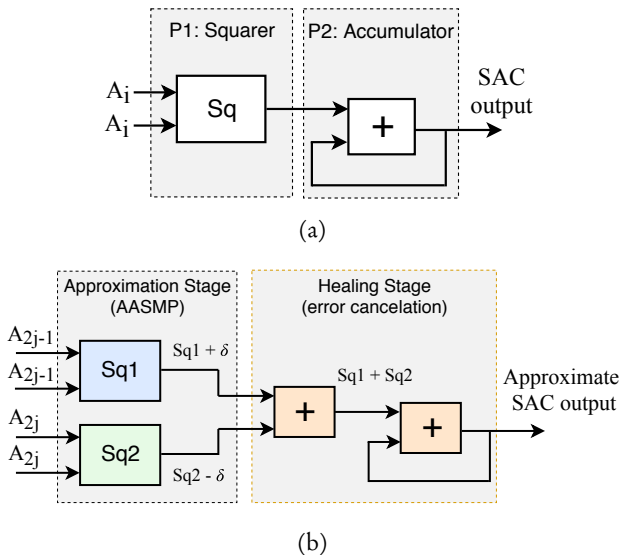


Figure 4.3: Square-accumulate (SAC) architectures; (a) accurate; (b) proposed approximate SAC utilizing an Absolute Approximate Squarer Mirror Pair (AASMP). Given the same input distribution for Sq1 and Sq2, the errors ($+\delta$ and $-\delta$) originated at the approximation stage are canceled out at the healing stage.

SAC hardware block (Fig. 4.3a), however, the overall circuit area of a parallel architecture remains the same for the proposed architecture as we require $L/2$ SAC hardware blocks instead of L for the same throughput.

4.3 ANALYSIS OF APPROXIMATE SAC COMPOSED OF TRUNCATED SQUARER

This section presents the mathematical analysis, and simulation of various truncated SAC alternatives that establish the basis for Self-Healing (SH). For a pair of approximate squarers, we present two ways by which the MEE can be achieved. In this section we discuss truncated squaring and in Section 4.4 we discuss logic pruning (reducing the number of logic gates) as a means of approximation.

4.3.1 MATHEMATICAL ANALYSIS OF TRUNCATED SQUARING

We consider the SAC operation executed on n bit words and the t Least Significant Bits (LSBs) are truncated. Here we consider the signed numbers represented in 2's complement. Let A be defined as a random number,

$$-2^{n-1} \leq A < 2^{n-1} \quad A \in \mathbb{Z}$$

For the input pairs (A, A) , we can describe all numbers except the number 0 (zero) in the n -bit range by either having $(+a, +a)$ for a positive input and $(-a, -a)$ for a negative input, where $a = |A|$. We have two possible cases for the SAC input pairs,

$$\begin{aligned}(A, A) &= (+a, +a) \text{ and} \\ (A, A) &= (-a, -a)\end{aligned}$$

Let α_t represent the input number with the t LSBs truncated. Truncation of t LSBs means that t LSBs are removed from an n -bit number. Therefore, α_t is an $n - t$ bit number and its value is given as:

$$\alpha_t = A - m \quad 0 \leq m < 2^t, m \in \mathbb{Z}$$

where m is the truncation error. For the input pair $(+a, +a)$ the product of the truncated factors is given by:

$$\begin{aligned}\alpha_t \cdot \alpha_t &= (a - m_1)(a - m_1) \quad 0 \leq m_1 < 2^t, m_1 \in \mathbb{Z} \\ &= aa - 2am_1 + m_1m_1\end{aligned}\tag{4.3}$$

In Eq. (4.3) the product aa is the accurate result of the squaring. The terms $2am_1$ and m_1m_1 are errors introduced by the truncation.

Let's assume: $m_1 \ll a$, i.e., the truncation error is much smaller than the original value of the number that was truncated. The product can now be approximated by:

$$\alpha_t \cdot \alpha_t \approx aa - 2am_1\tag{4.4}$$

For the input pair $(-a, -a)$,

$$\begin{aligned}\alpha_t \cdot \alpha_t &= (-a - m_2)(-a - m_2) \quad 0 \leq m_2 < 2^t, m_2 \in \mathbb{Z} \\ &= aa + 2am_2 + m_2m_2 \\ &\approx aa + 2am_2\end{aligned}\tag{4.5}$$

Eq. (4.4) and Eq. (4.5) show that one case introduces a negative error (approximately $-2am_1$) and the second case introduces a positive error (approximately $+2am_2$). Assuming that these two cases have equal probability and $m_1 \approx m_2$, we can conclude that the error of the first case approximately cancels the error of the second case at the accumulation stage.

This is an interesting property of the truncated squarer in a SAC architecture, for squaring signed numbers, where the errors higher and lower than zero have a potential to be canceled out at the healing stage (accumulator).

In case of squaring, the product is always a positive number. Therefore, we have a choice to invert the signs of inputs (multiplicands) without affecting the output. The MEE can be achieved by utilizing an approximate squarer pair (Sq1 and Sq2), where Sq1 squares the truncated input numbers that are made positive (before truncation) while Sq2 squares the truncated input numbers that are made negative (before truncation) to form a *mirror pair* as,

$$\text{Sq1: } A \rightarrow (|A|_t)^2$$

$$\text{Sq2: } A \rightarrow (|-A|_t)^2$$

where A is an input, and the subscript t denotes a truncation operation. The motivation behind this proposal is that on average the amount of positive number squarings equals the amount of negative number squarings which provides an opportunity to cancel out the truncation error.

A motivational example of 1-bit truncated squaring of an 8-bit integer ($n = 8$, $t = 1$) is shown in Fig. 4.4, where the errors are shown for squaring the integer as a positive $((+25)_{10})$ and a negative $((-25)_{10})$ number after truncation. The truncated squaring produces the output as a 14-bit integer that is shifted left (2-bit) to achieve a 16-bit output. The errors ($\epsilon_1 = -49$ and $\epsilon_2 = 51$) show the mirror effect that have a potential to be approximately canceled out at the accumulator.

In the above example, $(00)_2$ is appended to the least significant position after squaring in order to produce the 16-bit output. However, any 2-bit combination can be hard-wired to the least significant position targeting the lowest error output. Below we consider design alternatives for these least significant positions in case of 1-bit truncated squaring of 8-bit input integers ($n = 8$, $t = 1$) and compare their output quality in a SAC architecture.

Actual_s: A conventional approximate design that squares the truncated operands as they are originally fed (without changing sign) and appends 00 to form a 16-bit output.

Pos_{xx}: A design alternative that makes every input a positive integer before truncating and computing the square operation. The example (Fig. 4.4) illustrates that truncated squaring of positive integers produces an output smaller than (or in some cases equal to) the exact result. For appending two bits to the 14-bit squarer output, five options are considered: **Pos₀₀**, **Pos₀₁**, **Pos₁₀**, **Pos₁₁**, and **Pos_{LL}** that append $(00)_2$, $(01)_2$, $(10)_2$, $(11)_2$ and $(LL)_2$ respectively to produce a 16-bit output. Here $(LL)_2$ is the two times repetition of the truncated 1-bit.

Neg₀₀: A design alternative that makes every input a negative integer before truncating, and appends 00 to produce the 16-bit output. As the example (Fig. 4.4) shows that truncated squaring of negative integers produces an output larger

	$(25)_{10}$	$(-25)_{10}$
Input (2's complement)	00011001 $(25)_{10}$	11100111 $(-25)_{10}$
1 bit truncation	0001100 $(12)_{10}$	1110011 $(-13)_{10}$
Squaring	$(12)_{10} * (12)_{10}$	$(-13)_{10} * (-13)_{10}$
14-bit output	00000010010000 $(144)_{10}$	00000010101001 $(169)_{10}$
Append 00 for 16-bit output	0000001001000000 $(576)_{10}$	0000001010100100 $(676)_{10}$
Error	$\mathcal{E}_1: (576)_{10} - (625)_{10} = (-49)_{10}$	$\mathcal{E}_2: (676)_{10} - (625)_{10} = (51)_{10}$

Figure 4.4: Truncated squaring of a number as a positive integer and as a negative integer demonstrates the Mirror Error Effect (MEE). The subsequent accumulation can cancel out the errors partially and improves the overall output quality.

than (or in some cases equal to) the exact result, we do not consider appending 01, 10, 11, and LL because it will increase the error.

MEE_x: Mirror error effect designs (MEE₁ and MEE₂) square the truncated operand in the proposed self-healing fashion, where half of the inputs are made positive integers and the other half are made negative integers before truncating and squaring. It is to be noted that Sq1 and Sq2 in MEE_x designs provide errors in opposite signs but not in equal magnitudes. Therefore, swapping Sq1 with Sq2 brings different quality for finite-length input vectors. For that reason, we consider both design alternatives, **MEE₁**: odd-indexed elements of an input vector are considered as positive and even-indexed as negative integers, **MEE₂**: odd-indexed elements of an input vector are considered as negative and even-indexed as positive integers.

Fig. 4.5 shows a quality comparison for the above design choices in a SAC architecture. Inputs are considered as signed numbers represented in 2's complement. Two input distributions have been assessed: uniform (Uniform) and normal (Norm_x). Each distribution has 1000 vectors ($v = 1000$) of 10,000 elements each. We have utilized the Mean Square Error (MSE) metric² [62, 94] to compute the error in dB as in [9],

$$\text{MSE (dB)} = 10 * \log_{10} \left[\sum_{i=1}^v (Ex_i - Ax_i)^2 / v \right] \quad (4.6)$$

where Ex_i is the exact SAC output and Ax_i is the approximate counterpart for the i^{th} vector; and v is the number of test vectors. The normal distribution has been analyzed with standard deviation of 15.9 and various mean values: 0, 10,

²We assume that several SAC outputs are independent of each other, i.e., several inner products are independent of each other. Therefore, we use MSE for quality comparison.

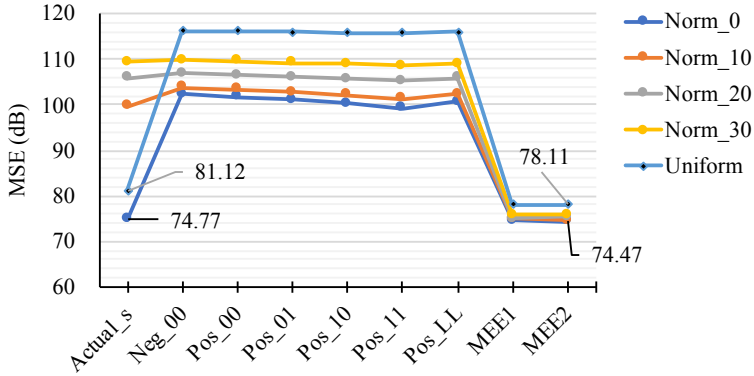


Figure 4-5: Quality analysis for approximate SAC utilizing various truncated squaring strategies. For every considered input distribution, MEE designs are outperforming because of error cancellation at the accumulator stage.

20 and 30. For instance, Norm_0 refers to a normal distribution with 0 mean and Norm_30 to a mean of 30 in Fig. 4-5.

It can be noted that self-healing based designs utilizing the MEE (MEE₁ and MEE₂) bring the best quality for all considered input distributions. In case of a Norm_0 input, MEE_x designs have slightly better quality as compared to Actual_s design. However, as the mean is shifted away from zero (Norm_10, Norm_20, Norm_30), the advantage of MEE_x designs is quite significant. This is because, a normal distribution with 0 mean (ideal case) is more likely to have an equal number of positive and negative integers which inherently produces the *mirror error effect* within a conventional design. However in general cases, where input distributions of various mean values can be possible, only self-healing based designs like MEE₁ and MEE₂ ensure efficient error cancellation to provide the best quality output.

The above analysis provides the foundation of the self-healing methodology that shows a better quality output as compared to the conventional way of applying approximations. However, it is important to note that for truncated designs, the MEE is achieved in the sign of errors, i.e., the errors have opposite signs, while the magnitudes are still unequal. Therefore, an *approximate mirror pair* can be formed that only partially cancels out the error. Secondly, inverting signs of input operands may render hardware costs depending upon the data representation. In the following section, we discuss how to achieve the MEE for logic-pruned (approximate) $n \times n$ squarers that utilize 2×2 squarers and 2×2 multipliers to produce *absolute approximate mirror pairs*, which cancel out the error originated within the pairs completely.

4.4 ABSOLUTE APPROXIMATE SQUARER MIRROR PAIR (AASMP)

As discussed in Chapter 2, an $n \times n$ recursive multiplier can be constructed from $(n/2)^2$ elementary (2×2) multipliers, where n indicates the width of input operands A and B in bits, $n \in \{4, 8, 16, 32, \dots\}$. These 2×2 elementary multipliers form the partial products and summing the bit-shifted partial products produces the overall product by utilizing adder trees. Any number out of the 2×2 partial products and/or adders can be approximated to achieve an approximate $n \times n$ multiplier [101].

In case of an $n \times n$ squarer, the number of required elementary (2×2) modules is less than $(n/2)^2$ due to the repetition of a few partial products with the same inputs. The only exception is a 2×2 squarer ($n = 2$). Without loss of generality, here we consider an 8×8 unsigned squarer (Sq8x8) to design an approximate *mirror pair*, wherein the error introduced by one approximate squarer is additive inverse (opposite in sign and equal in magnitude) of the other. Therefore, the pair is called AASMP. To achieve an approximate Sq8x8, we have employed approximations in 2×2 partial products constructs only, not for the adder trees.

First we discuss the construction of an accurate Sq8x8 module based on elementary (2×2) modules. Let $\text{Sq8x8} = A * A$, where $A = a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0$ is an 8-bit unsigned number and a_7 and a_0 are the most significant and least significant bits respectively. Fig. 4.6 illustrates the Sq8x8 computation, which shows that a total of four 4×4 partial products is required. Out of the four 4×4 partial products, two compute 4-bit square (Sq4x4) operations and the other two compute 4-bit multiply (P4x4) operations. However, the two P4x4 operations multiply the same (equal) inputs as shown in Fig. 4.6 (left). Therefore, an Sq8x8 block requires a P4x4 and two Sq4x4 blocks along with an adder tree.

Similarly, each 4×4 partial product can be computed by utilizing four 2×2 partial products as shown in Fig. 4.6 (right). In case of an Sq4x4 operation, two out of the four 2×2 partial products compute 2-bit square (Sq2x2) operations and the other two compute 2-bit multiply (P2x2) operations. However, both P2x2 multiply the same (equal) inputs, Fig. 4.6 (right). Therefore, an Sq4x4 hardware block requires one P2x2 and two Sq2x2 elementary modules along with an adder tree. On the other hand, a P4x4 hardware block requires four P2x2 elementary modules along with the adder tree. This explains why we require less number of elementary (2×2) modules for an $n \times n$ squarer as compared to a general multiplier that requires $(n/2)^2$ elementary modules. Fig. 4.7 illustrates the construction of an Sq8x8 architecture that requires ten elementary (2×2) modules. Out of the ten elementary modules, four compute 2×2 square operation (Sq2x2) and 6 compute 2×2 multiply operation (P2x2), see Appendix A for the details.

To achieve an approximate Sq8x8, any number out of the ten elementary modules can be approximated based upon the error tolerance of an application. To design an 8×8 AASMP, we propose to utilize approximate P2x2 and Sq2x2 modules

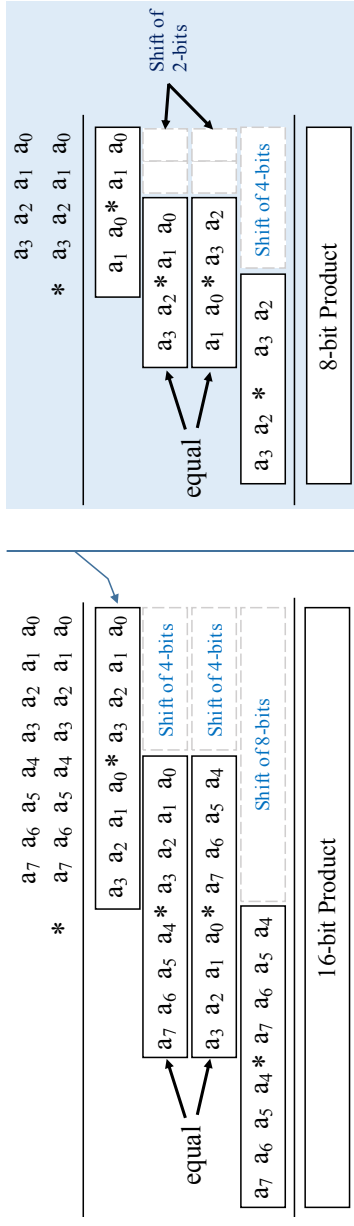


Figure 4.6: Sq8x8 computation utilizing 4×4 partial products (left), each 4×4 partial products can be computed by deploying 2×2 partial products (right).

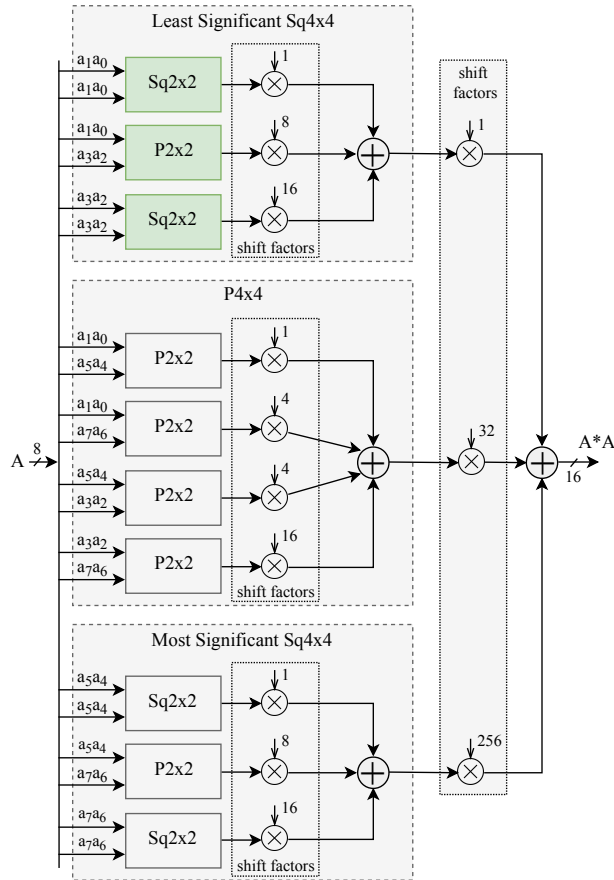


Figure 4.7: An 8×8 Squarer (Sq8x8) construction utilizing ten elementary (2×2) modules, with four squarers (Sq2x2) and six general multipliers (P2x2).

in one Sq8x8 and to utilize the *mirror* approximate counterparts in the other Sq8x8 to form a pair. Therefore, to design an 8×8 AASMP, we first discuss how to design 2×2 AASMP and 2×2 AAMMP.

4.4.1 DESIGN OF 2×2 ABSOLUTE APPROXIMATE MIRROR PAIRS

Several choices can be made to design an Sq2x2 AASMP or a P2x2 AAMMP. Keeping in view the truth tables, any output (except zero) can be approximated with a $\pm\delta$ error within a pair, i.e., $+\delta$ for one approximate 2×2 module and $-\delta$ for the other to form an *absolute approximate mirror pair*. For instance, a P2x2 AAMMP can utilize design choices such as: $2*1 \approx 1$ and 3 ($\epsilon = \pm 1$), $3*2 \approx 5$ and 7 ($\epsilon = \pm 1$), or 4 and 8 ($\epsilon = \pm 2$). Comprehensive design space exploration to get an optimal 2×2 *absolute approximate mirror pair* is beyond the scope of this

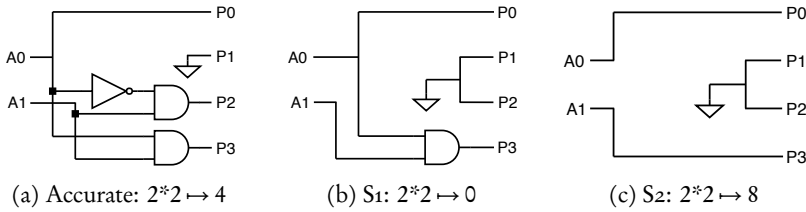


Figure 4.8: Logic diagrams of 2×2 squarers; (a) accurate design; (b) proposed approximate design S_1 ; (c) proposed *absolute approximate mirror* of S_1 .

work, we only intend to show here how it can be designed. Noteworthy, only those 2×2 *absolute approximate mirror pairs* are interesting that provide better hardware efficiency as compared to that of an accurate pair.

2×2 AASMP Design We propose an approximate 2×2 squarer: $Sq_{2 \times 2}$ (S_1) which introduces an error for one out of four possible input combinations: $2*2 \approx 0$, instead of 4 ($\epsilon_{S_1} = -4$). An *absolute approximate mirror* of S_1 is S_2 , which computes $2*2 \approx 8$ ($\epsilon_{S_2} = +4$). The logic diagrams of the accurate $Sq_{2 \times 2}$, S_1 and S_2 are shown in Fig. 4.8. Combining S_1 and S_2 in a pair forms a 2×2 AASMP which provides a better hardware efficiency as compared to that of an accurate pair since the number of gates is less.

2×2 AAMMP Design Consider M_1 as shown in Fig. 4.2b, which is an approximate 2×2 multiplier ($P_{2 \times 2}$) that introduces an error for one out of sixteen possible input combinations: $3*3 \approx 7$, instead of 9 [63], for the error: $\epsilon_{M_1} = -2$. To design M_p , which is an *absolute approximate mirror* of M_1 , we propose $3*3 \approx 11$, $\epsilon_{M_p} = +2$. Combining M_1 and M_p in a parallel pair produces our proposed 2×2 AAMMP. The logic diagram and truth table of M_p are shown in Fig. 4.9.

It can be seen from the logic diagrams that M_p (Fig. 4.9) requires more gates as compared to that of M_1 (Fig. 4.2b) while providing the same error rate and (absolute value of) error magnitude. This means that only using M_p as an approximate 2×2 multiplier is not recommended. However, the pair of M_p and M_1 that forms an AAMMP is proposed that provides a better quality-efficiency trade-off as compared to using two M_1 multipliers, see Section 4.5 and Section Section 4.6.

4.4.2 8×8 AASMP DESIGN

As discussed earlier, an $Sq_{8 \times 8}$ architecture is composed of six $P_{2 \times 2}$ and four $Sq_{2 \times 2}$ elementary modules (Fig. 4.7). Approximate elementary modules like M_1 , M_p , S_1 and S_2 can be utilized to design an approximate $Sq_{8 \times 8}$. In the case of an 8×8 AASMP design, M_1 and S_1 modules can be utilized for one $Sq_{8 \times 8}$ ($\epsilon_1 = -\delta$), and the M_p and S_2 modules for the other $Sq_{8 \times 8}$ ($\epsilon_2 = +\delta$) to form

A \ B	00	01	10	11
00	0000	0000	0000	0000
01	0000	0001	0010	0011
10	0000	0010	0100	0110
11	0000	0011	0110	1011

(a) Truth table of Mp.

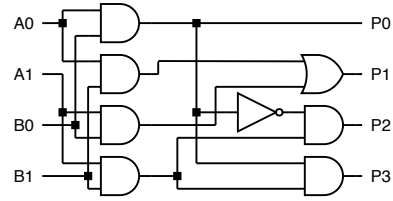
(b) Mp: $3 \times 3 \mapsto 11$

Figure 4.9: Logic diagram and truth table of the proposed 2×2 multiplier (Mp), which is an *absolute approximate mirror* of M1.

a pair. Thus, for a given input distribution, each Sq8x8 in an AASMP provides an error that is equal in magnitude and opposite in sign (additive inverse) as compared to that of the other Sq8x8.

4.4.3 $n \times n$ AASMP DESIGN

We have elaborated on the design of an 8×8 AASMP so far. In a similar fashion, an $n \times n$ AASMP can be designed by utilizing the negative error elementary 2×2 designs ($\epsilon = -\delta$, like M1 and S1) for one $n \times n$ squarer and the positive error elementary designs ($\epsilon = +\delta$, like Mp and S2) for the other $n \times n$ squarer to form a pair.

4.5 DESIGNING AN OPTIMAL APPROXIMATE SAC ACCELERATOR

In this section, we present a design flow for building an optimal approximate square-accumulate (SAC) accelerator. Fig. 4.10 illustrates the proposed design methodology.

The first step (Stage 1) performs a design space exploration of an approximate $n \times n$ squarer designed in a conventional way. The conventional way utilizes the approximate 2×2 designs that are based on the conventional methodology, i.e., M1 and S1 as approximate 2×2 multiplier and squarer respectively. It does not utilize the *mirror* approximate designs (Mp and S2). The design space exploration considers all the possible configurations of an $n \times n$ squarer, where each 2×2 multiplier and squarer can be filled-in with an accurate or an approximate design.

Based on input statistics, the error metric (or conversely: quality) is computed for each possible $n \times n$ configuration, see Appendix B for details. Also, the hardware cost (or conversely: efficiency) is computed for each possible $n \times n$ configuration. Then the Pareto-optimal $n \times n$ configurations are selected that provide an optimal trade-off between a defined error metric and an efficiency target (area/power/performance). We present the design space exploration of an approximate 8×8 squarer (Sq8x8) as shown in Fig. 4.11, wherein the Mean

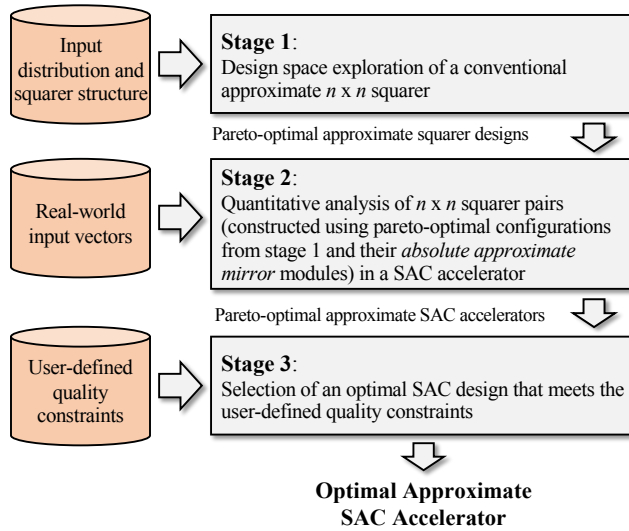


Figure 4.10: Design methodology for building an optimal approximate Square-Accumulate (SAC) accelerator. The conventional approximate squarer designs (Stage 1) are based on the conventional approximate computing methodology. These designs are paired with their *absolute approximate mirror pairs* and are tested for pareto-optimality in a SAC accelerator (Stage 2). Finally, an optimal approximate SAC accelerator is chosen based on the user-defined quality constraints (Stage 3).

Error (ME) is plotted against the chip-area (Area) for uniform and normal input distributions. We assume the efficiency target is chip-area of an 8×8 squarer, which is computed by adding the chip-areas of individual 2×2 multiplier/squarer modules. The chip-areas based on TSMC 40nm Low Power (TCBN40LP), synthesized at 1.43GHz, are shown in Table 4.1.

It is to be noted that the mean error (ME) is an important error metric for the output-quality analysis of an approximate squarer in a SAC architecture. When an approximate squarer produces a low ME (or ideally a zero ME) the subsequent accumulator produces a small overall error (or ideally a zero error) by averaging out the errors. Therefore, we have utilized the (normalized) ME metric. Let n_{ec} be the number of input error-combinations of an approximate squarer configuration, i.e., the number of input combinations that generate an error in an approximate squarer configuration. Let \mathbf{e} be a vector containing the error values corresponding to each input error-combination. For a given input distribution, let \mathbf{p}_e be a vector containing the corresponding probabilities of each input error-combination. This means that the size of vector \mathbf{e} is n_{ec} , which is equal to the size of vector \mathbf{p}_e . The normalized ME metric can be computed as

Table 4.1: Area of elementary 2×2 multipliers (P2x2) and squarers (Sq2x2).

Multiplier/Squarer	Area (μm^2)
Accurate P2x2	9.65
Conventional Approx P2x2 (M1)	7.05
Accurate Sq2x2	3.53
Conventional Approx Sq2x2 (S1)	2.12

follows,

$$ME[\text{normalized}] = \left(\sum_{i=1}^{n_{ec}} \mathbf{e}(i) \times \mathbf{p}_e(i) \right) / (2^{2n} - 1) \quad (4.7)$$

where $2n$ is the number of output bits. Based on the available design space, the pareto-optimal configurations (that provide an optimal trade-off between Area and ME as shown in Fig. 4.11) are selected for later stages (Stage 2 and Stage 3) of the design process. Each pareto-optimal point in Fig. 4.11 represents an approximate Sq8x8 configuration based on six 2×2 multipliers and four 2×2 squarers (see Fig. 4.7). Each 2×2 multiplier is selected from one of P2x2 and M1. Each 2×2 squarer is selected from one of Sq2x2 and S1. For example, the pareto-optimal point shown at the top-left position in Fig. 4.11a represents the following configuration (from the least significant (left) to the most significant (right) 2×2 module):

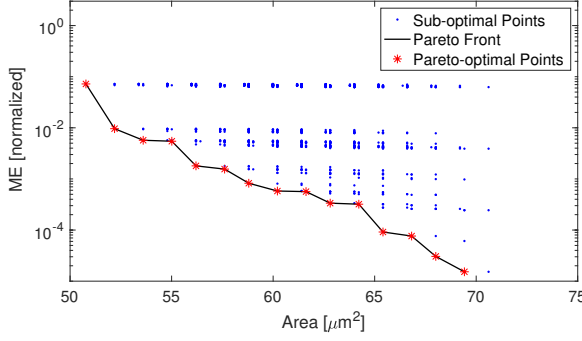
S1 M1 S1 M1 M1 M1 M1 S1 M1 S1

The above pareto-optimal configuration has an area consumption of $50.8 \mu m^2$ and all of its 2×2 modules are approximate. Therefore, this configuration has the lowest computation cost (or the highest efficiency) and the maximum ME (or the lowest quality) as compared to the other pareto-optimal points. On the other hand, the pareto-optimal point at the bottom-right position in Fig. 4.11a represents the following configuration (from the least significant (left) to the most significant (right) 2×2 module):

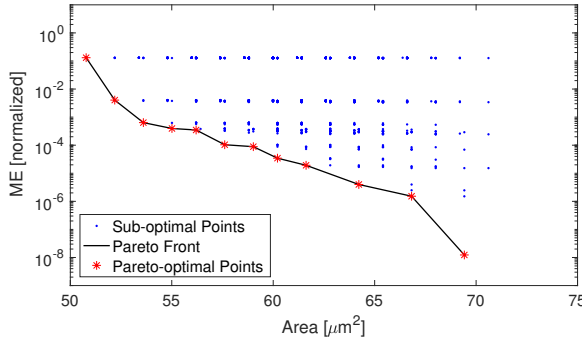
Sq2x2 M1 Sq2x2 P2x2 P2x2 P2x2 P2x2 Sq2x2 P2x2 Sq2x2

The above pareto-optimal configuration consumes $69.4 \mu m^2$ of the chip-area and has only one 2×2 approximate module, the rest of the nine 2×2 modules are accurate. Therefore, this configuration has the maximum computation cost (or the lowest efficiency) and the minimum ME (or the highest quality) as compared to the other pareto-optimal points.

Although, Stage 1 provides us with the configurations that offer an optimal quality-efficiency trade-off, it is to be noted that a significant quality degradation will occur if we employ higher error rate and higher error magnitude approximations in the elementary 2×2 modules (aiming at higher efficiency). However, if we introduce approximations in the proposed self-healing manner such that the mean error of two modules forming a *mirror pair* are additive inverse of each



(a) ME vs. Area for uniformly distributed inputs.



(b) ME vs. Area for normally distributed inputs.

Figure 4.11: Design space of an approximate Sq8x8 module constructed using elementary 2×2 modules: accurate P2x2, M1, accurate Sq2x2 and S1. The normal input distribution has $\mu = 128$ and $\sigma = 22.5$. Pareto-optimal points are chosen that provide best efficiency for a given quality constraint and vice versa.

other, we have:

$$E(\text{ME}_{\text{total}}) = E(\text{ME}_1) + E(\text{ME}_2) = 0 \quad (4.8)$$

where $E(\text{ME}_1)$ and $E(\text{ME}_2)$ are the expected ME values of the two squarers in a *mirror pair*, and $E(\text{ME}_{\text{total}})$ is the overall expected ME value of the approximate *mirror pair*. Eq. (4.8) shows that the overall expected ME value of a squarer *mirror pair* is zero because $E(\text{ME}_1) = -E(\text{ME}_2)$, which will result in an overall zero error after the accumulation stage in a SAC architecture. This will also hold true for high error rate and high error magnitude approximate elementary 2×2 designs in a *mirror pair*. This motivates the Stage 2, where the $n \times n$ pareto-optimal squarer configurations (generated in Stage 1) are paired with their *absolute approximate mirror* modules as discussed in Section 4.4.

Table 4.2: Area of 2×2 multiplier (P2x2) and squarer (Sq2x2) pairs. M1 is a conventional approximate multiplier [63]. Mp is the proposed *absolute approximate mirror* of M1. S1 and S2 are the proposed approximate squarers that are *absolute approximate mirrors* of each other.

Multiplier/Squarer Pair	Area (μm^2)
Pair of two accurate P2x2	19.29
Pair of two M1	14.11
AAMMP (pair of M1 and Mp)	15.52
Pair of two accurate Sq2x2	7.05
Pair of two S1	4.24
AASMP (pair of S1 and S2)	2.82*

* S1 and S2 consume $2.12 \mu m^2$ and $0.7 \mu m^2$ of the chip-area, respectively.

Nevertheless, $E(ME_{total})$ is not necessarily zero in case of random *finite-length* input distributions for the self-healing case as in Eq. (4.8). This brings the importance of finite-length random input analysis for quality-efficiency trade-off evaluation and comparison of the self-healing methodology with the conventional methodology. Therefore, at Stage 2, the quantitative analysis of the pairs (AASMPs) is performed by considering the real world scenarios that involve random input vectors of finite (limited) length. We present these simulations for both self-healing and conventional methodologies to compare their overall quality-efficiency trade-off. We consider uniform and normal ($\mu = 128$ and $\sigma = 22.5$) distributions, where each distribution has two finite-lengths: a small data size with 100 vectors of 124 elements each and a large data size with 1000 vectors of 10,000 elements each. The small data size resembles one channel of radio astronomy calibration data (LOFAR facility [124]). On the other hand, the large data size represents the considered probability distribution relatively better than the small data size.

As in Stage 1, chip-area is considered as efficiency target. However, here we consider a pair of 8×8 squarers utilizing 2×2 multiplier (P2x2) and 2×2 squarer (Sq2x2) pairs in a conventional and the proposed way. In case of the conventional P2x2 pair, both P2x2 are M1 ($3 \times 3 \approx 7$). But when we make an *absolute approximate multiplier mirror pair* (AAMMP), one of the two P2x2 is M1 ($3 \times 3 \approx 7$) and the other is Mp ($3 \times 3 \approx 11$). Likewise, both Sq2x2 of a conventional Sq2x2 pair are S1, while an *absolute approximate squarer mirror pair* (AASMP) utilizes S1 and S2 to form a proposed Sq2x2 pair. Table 4.2 shows the area of aforesaid pairs, synthesized at 1.43GHz for TSMC 40nm Low Power (TCBN40LP) technology. The area cost of each 8×8 squarer design is estimated by adding the areas of 2×2 constructs only, not the adder trees. This estimation is plausible for comparison purpose because the adder trees remain accurate in all designs. However, in Section 4.6 we will present synthesis results of complete designs, including the adder trees.

Fig. 4.12 shows the quality-efficiency trade-off of approximate Sq8x8 pairs

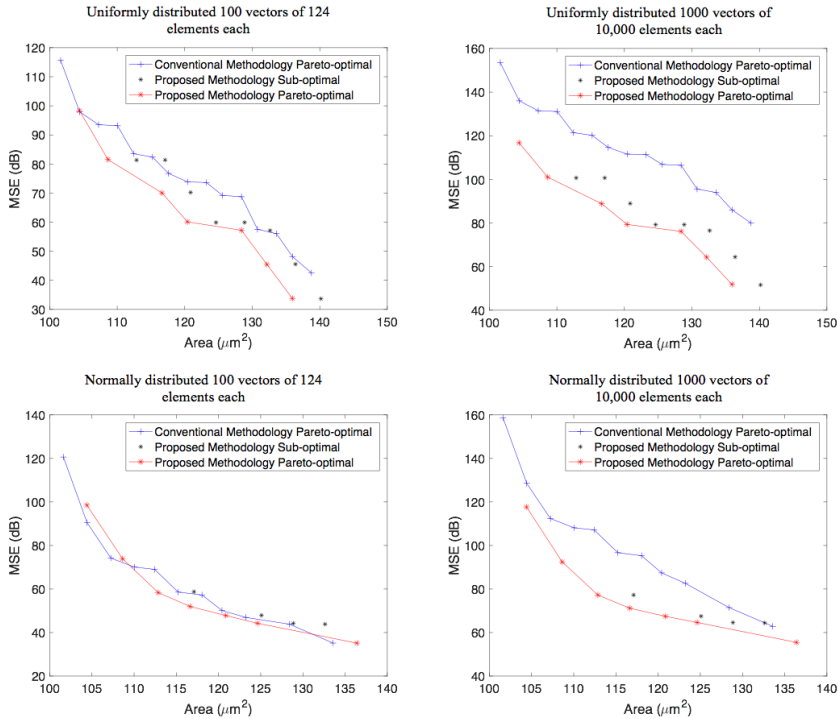


Figure 4.12: Quality-efficiency trade-off of pareto-optimal designs for the conventional and the proposed SH methodologies. Various finite-length randomly distributed inputs are considered. The proposed SH methodology provides a more effective trade-off for all considered input distributions, which is quite significant in case of larger vector inputs.

(AASMPs) in a SAC accelerator for random finite-length inputs. We also compare the conventional methodology pareto-optimal designs with that of our proposed SH methodology. The error metric (MSE) is computed at the output of the accumulator using Eq. (4.6). Noteworthy, because the overall error of a SAC (inner product) operation depends upon several outputs of the squarer, we have utilized mean error metric for designing an approximate squarer. However, on the other hand, several SAC outputs are independent of each other, i.e., several inner products are independent of each other. Therefore, we have utilized the MSE metric for the overall SAC output.

As expected, the quality-efficiency trade-off of the conventional methodology in Fig. 4.12 follows the same trend as of Fig. 4.11 for both uniform and normal input distributions. However, for normally distributed inputs, eleven pareto-optimal points (designs) are shown in Fig. 4.12 for the conventional methodology while there are twelve in Fig. 4.11. This is because the lowest approximation level

(maximum area) pareto-optimal point in Fig. 4.11 has generated zero error for finite-length input due to absence of the error case³, which brings an MSE(dB) value of $-\infty$ ($\log_{10}(0) = -\infty$). Therefore, this design point is not shown in Fig. 4.12.

It is to be noted that a low MSE value and a small area are desired. Therefore, in Fig. 4.12, a pareto front that is near to the origin of the graph is a better option. It follows from the quality-efficiency trade-off illustrated in Fig. 4.12 that self-healing improves the pareto front (the optimal designs) for almost all considered input cases. For large data sizes (large vector inputs), the pareto front of the proposed self-healing methodology completely outperforms the conventional counterpart, because the large random vectors have more tendency towards an ideal input distribution (uniform or normal in this case). However, in case of smaller data sizes (small vector inputs), randomly generated vectors have a relatively higher deviation from the ideal distribution, which results in less error cancellation in case of the proposed self-healing methodology. Nevertheless, self-healing still improves the trade-off for smaller data sizes by introducing several better pareto-optimal designs as shown in Fig. 4.12.

It should be noted that the smallest area (highest approximation level) of the proposed self-healing methodology is greater than the smallest area of the conventional methodology. This is due to the fact that a 2×2 AAMMP has a larger area as compared to an approximate conventional pair (the pair of two M_1) as shown in Table 4.2. Specifically, in case of a conventional approximate multiplier pair, we have two M_1 , while in case of an AAMMP, we have M_1 and M_p . As the area of M_p is larger than M_1 , we have more area cost for a 2×2 AAMMP. Moreover, because of a lower probability of error at a 2×2 approximate multiplier compared to an approximate squarer⁴, the pareto-optimal designs tend to utilize more 2×2 AAMMPs as compared to 2×2 AASMPs. Consequently, the 8-bit approximate squarer pairs have a higher area cost for the self-healing methodology as compared to conventional methodology. However, as we see the complete trade-off, the proposed self-healing methodology increases the design space (i.e., offers additional pareto-optimal designs), and provides overall a more effective quality-efficiency trade-off.

It is to be noted that other efficiency targets can also be considered, e.g., power, performance, or energy to find the pareto-optimal designs from Stage 1 and Stage 2 (see Fig. 4.10), where the relevant cost functions can be utilized like power consumption, delay, or power-delay-product respectively. Subsequently, while having a clear quality-efficiency trade-off (as in Fig. 4.12), an optimal SAC

³The configuration related of this pareto-optimal point has only one approximate 2×2 module: the most significant 2×2 multiplier. To generate an error case for such a configuration, the 8-bit input (see Fig. 4.7) should not be less than $(240)_{10}$, which is more than 4σ away from the mean in our considered normally distributed input ($\mu = 128$ and $\sigma = 22.5$). Therefore, the probability of an error case for such a configuration is extremely low, which did not show up in our finite-length data.

⁴For example, the probability of error for M_1 is $1/16$ and the probability of error for S_1 is $1/4$ for a uniformly distributed input.

accelerator design can be chosen based on user-defined quality constraints for the given input distribution, which accomplishes Stage 3 of the design process.

4.6 EXPERIMENTAL SETUP AND RESULTS

77

To compare the conventional and the proposed self-healing designs, a quality-efficiency trade-off study based on estimated area has been discussed in Section 4.5. In this section, we consider some selected designs to quantify and validate the efficiency benefits of self-healing over the conventional methodology based on the synthesis of designs. We present results of a quality analysis and a hardware synthesis of the proposed and conventional approximate Sq8x8 designs deployed in square-accumulate accelerators for random finite-length input vectors. Moreover, for the radio astronomy calibration processing case study, we analyze the quality impact of self-healing and compare it with an equivalent efficiency design⁵ utilizing the conventional methodology.

4.6.1 EXPERIMENTAL SETUP FOR QUALITY-EFFICIENCY TRADE-OFF STUDY

Our experimental setup to study the quality-efficiency trade-off is illustrated in Fig. 3.9 (Chapter 3). Quality analysis has been performed in Matlab utilizing behavioral models of approximate multiplier/squarer designs. We used Synopsys tools: Design Compiler and Power Compiler to assess hardware costs, i.e., area and power, for the TSMC 40nm Low Power (TCBN40LP) technology library.

4.6.2 QUALITY-EFFICIENCY TRADE-OFF OF 8×8 SQUARER PAIRS IN A SAC ACCELERATOR

In this section, we present the comparison of some design alternatives of Sq8x8 pairs in a SAC accelerator to quantify the quality-efficiency benefits. We present the first synthesis of Sq8x8 pairs that are not necessarily pareto-optimal. The main purpose here is to provide a comparison at a coarse-level based on synthesis of designs and quality analysis. In Chapter 5, however, we present the synthesis results of pareto-optimal designs. The following designs are considered in this section,

Accu An accurate Sq8x8 pair, where both Sq8x8 squarers in a pair are composed of accurate 2×2 elementary modules (P2x2 and Sq2x2).

Convent1 An Sq8x8 pair designed utilizing the conventional approximation approach where the least significant P2x2 multiplier (shown in green in Fig. 4.7) is approximated as $M_1 (3^*3 \approx 7)[63]$ for both Sq8x8s in a pair.

⁵Equivalent efficiency designs refer to the designs that provide the same hardware efficiency in terms of the desired form, e.g., area and/or power efficiency.

Table 4.3: Computational cost comparison of accurate and approximate Sq8x8 pairs. Convent1 and Convent3 are based on conventional approximation methodology; while SH1, SH3 and SH7 are self-healing based approximate designs.

Parameters	Accu	Convent1	SH1	SH3	Convent3	SH7
Area _{_1} (μm^2)	642	617	626	599	582	436
Power _{_1} (μW)	460	437	445	423	426	399
Area _{_2} (μm^2)	1000	925	935	897	855	729
Power _{_2} (μW)	1079	1051	1048	979	977	795

Area_{_1} and Power_{_1} at 1GHz, and Area_{_2} and Power_{_2} at 1.43GHz.

SH1 An Sq8x8 pair designed utilizing the proposed self-healing approximation approach where the least significant P2x2 is approximated as M1 ($3*3 \approx 7$)[63] in one Sq8x8, and the least significant P2x2 is approximated as Mp ($3*3 \approx 11$) in the other Sq8x8. This forms an AASMP.

Convent3 In addition to Convent1 approximations, the two least significant Sq2x2 squarers are approximated as S1 for both Sq8x8s in an approximate pair. Therefore, approximating the three least significant 2×2 elementary modules (shown in green in Fig. 4.7) in a conventional fashion.

SH3 In addition to SH1 approximations, the two least significant Sq2x2 are approximated as S1 ($2*2 \approx 0$) for one Sq8x8, and approximated as S2 ($2*2 \approx 8$) for the other Sq8x8 to form an AASMP. Therefore, approximating the three least significant 2×2 elementary modules (shown in green in Fig. 4.7) in a self-healing fashion.

SH7 In addition to SH1 and SH3 approximations, all four elementary modules (P2x2) of P4x4 (see Fig. 4.7) are approximated as M1 for one Sq8x8 and as Mp for the other Sq8x8 to form an AASMP. Therefore, approximating seven least significant 2×2 elementary modules in self-healing fashion.

We compare the computational costs in terms of chip-area and power consumption of the above Sq8x8 pairs at the operating frequencies: 1GHz (Area_{_1}, Power_{_1}) and 1.43GHz (Area_{_2}, Power_{_2}) as shown in Table 4.3. Normally distributed input vectors have been utilized to estimate power consumption. For each approximate computing methodology (conventional and self-healing), Table 4.3 shows an increase in computational efficiency as the approximations are increased, i.e., a higher number of elementary 2×2 modules are approximated. For instance, the Convent3 design has a higher area and power efficiency as compared to that of the Convent1 design, and likewise, the SH7 design has a higher area and power efficiency as compared to that of the SH3 design.

For a quality comparison, the Mean Square Error (MSE) is computed at the SAC output as in Eq. (4.6). The result is shown in Fig. 4.13. We have analyzed

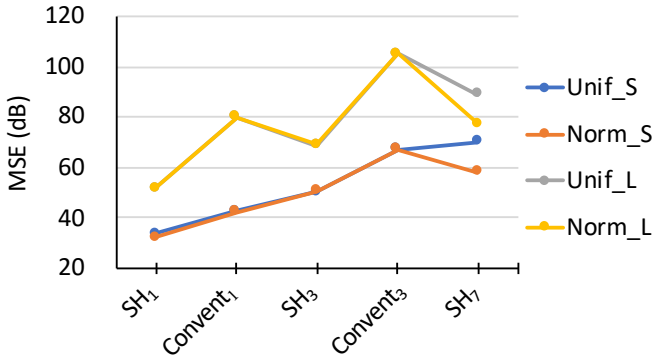


Figure 4.13: Quality comparison for various SAC designs utilizing Sq8x8 pairs. SH3 provides better quality as compared to conventional equivalent efficiency design (Convent3), while SH7 outperforms Convent3 both in quality (mostly) and efficiency.

Uniform (Unif_x) and normal (Norm_x) 8-bit unsigned input distributions, namely: Unif_S, Unif_L, Norm_S, Norm_L, where S stands for a small data size (100 vectors of 124 elements each) and L stands for a larger data size (1000 vectors of 10,000 elements each). We consider Norm_x distributions with $\mu = 128$ and $\sigma = 22.5$. In case of small data sizes (Unif_S and Norm_S), SH1 provides better quality as compared to the conventional approach (Convent1). Even though the computational efficiency of SH1 is lower than that of the Convent1 design, it introduces an additional design alternative in the trade-off. Moreover, SH3 brings higher efficiency at a lower quality as compared to Convent1 and also contributes to an additional design alternative. Interestingly, SH3 provides much better quality as compared to the Convent3 design while providing an almost equivalent power efficiency. This is due to the error cancellation opportunity brought by the self-healing approximation methodology. Also, SH7 provides better efficiency as compared to that of Convent3 design while also providing a better quality output for three out of four input cases.

For large data sizes with uniform (Unif_L) and normal (Norm_L) distributions, self-healing designs SH1 and SH3 show a better quality than the conventional design Convent1. Interestingly, SH3 deploys higher area- and power-efficiency as compared to Convent1 and still provides better quality due to error cancellation. Similar to small data sizes, self-healing designs SH3 and SH7 provide better quality as compared to the Convent3 design for large data sizes. SH7 shows better efficiency (up to 25% better area efficiency and up to 18.6% better power efficiency, see Table 4.3) as compared to the conventional design (Convent3) with a better quality output.

Therefore, we can conclude that in comparison to the conventional design approach, the self-healing approach increases the pareto-optimal design options

in some cases and brings better (higher efficiency and higher output quality) designs in the other cases to provide an effective quality-efficiency trade-off.

4.6.3 RADIO ASTRONOMY CALIBRATION PROCESSING – A CASE STUDY

As discussed in Chapter 3, radio astronomy calibration processing (StEFCal [107]) employs a Least Squares (LS) algorithm that takes sky visibilities as input and utilizes the model visibilities to estimate sensor gains for a given radio telescope configuration. It has three computationally expensive components: square-accumulate (SAC), multiply-accumulate (MAC) and element-wise product (PE), see Fig. 3.7 in Chapter 3. Here we only present the simulation results for the SAC accelerator that computes,

$$\sum_{j=1}^N \{(\mathbf{z}r_j * \mathbf{z}r_j) + (\mathbf{z}i_j * \mathbf{z}i_j)\} \quad (4.9)$$

where $\mathbf{z}r$ is the real part and $\mathbf{z}i$ is the imaginary part of a complex vector \mathbf{z} . Each squaring in Eq. (4.9) can be rewritten as a squarer pair to employ self-healing,

$$\sum_{k=1}^{N/2} \{[(\mathbf{z}r_{2k-1} * \mathbf{z}r_{2k-1}) + (\mathbf{z}r_{2k} * \mathbf{z}r_{2k})] + [(\mathbf{z}i_{2k-1} * \mathbf{z}i_{2k-1}) + (\mathbf{z}i_{2k} * \mathbf{z}i_{2k})]\} \quad (4.10)$$

We have utilized the radio astronomy calibration data of the LOFAR facility [124]. Table 4.4 shows the quality analysis of various design choices as discussed in Section 4.6.2 (Accu, Convent₁, SH₁, SH₃, Convent₃). For the MSE (SAC error), both self-healing designs (SH₁ and SH₃) provide better quality as compared to their conventional counterparts (Convent₁ and Convent₃ respectively). It is important to note that the conventional designs produce higher errors because of lack of error cancellation. However, the self-healing based designs can cancel out the error partially or fully (based on the input distribution). For that matter, we can see that SH₃ produces 46.7% better quality as compared to the conventional counterpart (Convent₃) while providing approximately the same computing efficiency as discussed in Section 4.6.2.

4.6.4 DISCUSSION AND FUTURE WORK

To quantify the quality-efficiency benefits of our proposed SH methodology, we have utilized a few pareto-optimal designs (in this section) obtained from the design space exploration (Section 4.5). These designs have been synthesized to estimate area and power consumption using Synopsys tools. For power consumption estimation, where switching activity is needed, we have only utilized a normally distributed input. Also, these designs have been simulated for radio astronomy calibration processing to assess their behavior in a real application

Table 4.4: Quality analysis of radio astronomy calibration for employing various approximate SAC alternatives. SH₃ and Convent₃ designs provide almost equal power efficiency, however, SH₃ brings 46.7% better quality with reference to Convent₃.

Design Alternatives	SAC Error (MSE)
Accu	0
Convent ₁	7.8341e-07
SH ₁	7.7470e-07
Convent ₃	2.25e-02
SH ₃	1.20 e-02

scenario. The benefits (area-/power-efficiency and quality improvements) reported in this section are not very precise because only a few designs have been tested. All the pareto-optimal designs should be tested for better quantification of advantages offered by the proposed methodology. By testing, we mean that the designs are synthesized for area consumption and their power consumption is estimated based on their target input distribution (e.g., uniform or normal), and their models are simulated in a real application scenario.

Secondly, to compare our proposed methodology with the conventional error restricted methodology, we only utilized a fail-rare based conventional approximate 2×2 multiplier (M₁) that provides only one error case out of sixteen possible combinations. As discussed in Chapter 2, a fail-small based approximate 2×2 multiplier (M₂) is also reported in the literature, which provides three error cases out of the total sixteen cases but with a relatively small magnitude of the error. However, in the comparison, we did not utilize M₂ because its *absolute approximate mirror* is not efficient. Nevertheless, in Chapter 5, we will discuss the internal-self-healing methodology (that utilizes the self-healing principle, internally, within an approximate $n \times n$ multiplier) and we will compare both the fail-rare and fail-small based designs with our proposed methodology.

Finally, it should be noted that there is a chance of overflow when $+\delta$ approximate designs are generated, see Chapter 5 for details. Although there is a low probability, some of the 8×8 squarer designs presented in this chapter may overflow based on the input distribution. To guarantee that none of the pareto-optimal designs overflows, the worst-case scenario has to be tested during the design space exploration to detect and exclude the overflow designs. In Chapter 5, where we will discuss $+\delta$ approximate $n \times n$ multipliers, we will utilize overflow detection and exclusion within the design space exploration process.

4.7 CONCLUSIONS

A Self-Healing (SH) methodology to enable efficient and systematic approximate computing has been presented. Our analysis has shown that exploit-

ing healing stages of an accumulation based algorithm in general, and of the square-accumulate (SAC) accelerator in particular, provides an effective quality-efficiency trade-off. We have shown how SH can be employed to truncation and logic pruning approximate computing techniques. We discussed the randomly distributed finite-length input analysis and a case study of radio astronomy calibration processing for an approximate SAC accelerator that showed a more effective quality-efficiency trade-off utilizing SH as compared to the conventional approximation methodology. For random input vectors, SH demonstrates up to 25% and 18.6% better area and power efficiency respectively with a better-quality output as compared to the conventional approximate computing methodology. As a case study, SH is applied to one of the computationally expensive components (square-accumulate) of the radio astronomy calibration application, where it shows up to 46.7% better quality for equivalent computing efficiency as that of conventional methodology.

Nevertheless, a comprehensive design space exploration—based on input distribution—is required to ensure the highest efficiency for a given quality constraint within radio astronomy calibration processing. We have shown how *absolute approximate mirror pairs* are designed for unsigned logic-pruned multipliers and squarers, and their utilization in SAC accelerators for an effective quality-efficiency trade-off. However, the utilization of SH for the signed multiplier case and Multiply-Accumulate (MAC) accelerators for attaining an effective quality-efficiency trade-off are indicated as future directions of research.

5

INTERNAL-SELF-HEALING METHODOLOGY FOR ACCUMULATION BASED APPROXIMATE ACCELERATORS

ABSTRACT – The self-healing methodology is constrained to parallel implementations with similar modules (or parts of a datapath) in multiples of two through the pairing of mirror versions to achieve error cancellation. In this chapter, we propose a methodology for Internal-Self-Healing (ISH) that allows exploiting self-healing within a computing element internally without requiring a paired, parallel module. We employ our ISH methodology to design an approximate multiply-accumulate (xMAC), wherein the multiplier is regarded as an approximation stage and the accumulator as a healing stage. We propose to approximate a recursive multiplier in such a way that a near-to-zero average error is achieved for a given input distribution to cancel out the error at an accurate accumulation stage. To increase the efficacy of such a multiplier, we propose a novel 2×2 approximate multiplier design that alleviates the overflow problem within an $n \times n$ approximate recursive multiplier. The proposed ISH methodology shows a more effective quality-efficiency trade-off for an xMAC as compared to the conventional error-restricted techniques.

As discussed in Chapter 4, the conventional approximate computing methodology suggests utilizing *fail-small*, *fail-rare*, or *fail-moderate* strategies [26, 88], wherein the errors are restricted as per their magnitudes and rates to avoid high loss in the output-quality. This is referred to as the *conventional methodology* in this chapter. An important drawback of the conventional methodology is a limited design-space, which excludes the approximations that introduce high error magnitudes and high error rates. This limitation hinders the achievable

This chapter is based on [G:4].

efficiency gains for a given quality constraint and therefore limits the efficacy of the quality-efficiency trade-off.

The *fail-balanced* technique for approximate computing has alleviated the aforesaid limitation in the design space, see Chapter 4. This technique does not restrict the approximations based on their error profiles but provides an opportunity for the error cancellation to deliver an effective quality-efficiency trade-off. This is referred to as the *self-healing methodology* here. Consider an example of a computing architecture, composed of two computing elements: P1 and P2, as shown in Fig. 5.1. The input stream is fed to P1 while the output is obtained from P2. The conventional methodology suggests approximating both computing elements with controlled error rates and error magnitudes to avoid an unacceptable (high) loss in the output-quality; see Fig. 5.1a. On the other hand, the self-healing methodology considers P1 as an *approximation stage* and P2 as a *healing stage*. The approximations are applied at the *approximation stage* (approximate P1) in such a way that their corresponding error is canceled out (partially or fully) in the subsequent *healing stage* (accurate P2). To achieve this, a pair of approximate P1 elements is required with a *mirror error effect*, i.e., the error introduced by each P1 in a pair is an additive or multiplicative inverse of the other; see Fig. 5.1b.

A limitation of the self-healing methodology is that it can only be employed in parallel architectures that have similar computing elements (or parts of a datapath) in *multiples of two*, so that the *mirror error effect* is achieved by pairing the similar computing elements. However, in case of datapaths that do not have similar elements in *multiples of two*, an approximation methodology is required that can provide the *mirror error effect* within a single computing element, as targeted in this chapter.

The principal contribution of this work is an Internal-Self-Healing (ISH) methodology where the *approximation stage* (P1, see Fig. 5.1c) is designed for an internal *mirror error effect* without requiring a parallel paired computing element. To elaborate on the ISH methodology, the following is presented in this chapter,

- » The approximate multiply-accumulate¹ (xMAC) concept with ISH methodology (Section 5.2).
- » Design of an $n \times n$ recursive multiplier with near-to-zero mean error and its efficacy for xMAC (Section 5.2.1).
- » Overflow handling scheme for near-to-zero mean error recursive multipliers and design of a novel approximate 2×2 multiplier that alleviates the overflow problem (Section 5.2.2).
- » We compare the conventional and the proposed ISH methodologies for chip-area and power optimized designs considering data with uniform

¹In contrast to Chapter 4, here we use a case study of an approximate multiply-accumulate accelerator which is more generalized as compared to an approximate square-accumulate (SAC) accelerator. Noteworthy, the proposed ISH methodology can also be employed to approximate SAC accelerators.

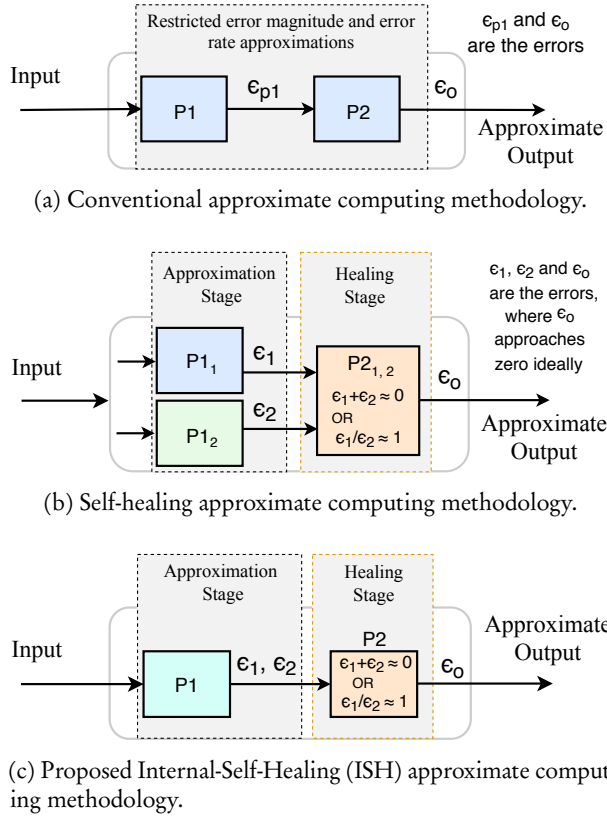


Figure 5.1: An overview of the conventional, the self-healing and the proposed approximate computing methodologies. The proposed ISH methodology does not require parallel computing elements but provides a *mirror error effect* within a single approximate element (P1).

and normal distributions and data obtained from a radio astronomy application (Section 5.3).

5.1 RELATED WORK

This section reviews the essential concepts concerning approximate multipliers, MAC, and the designs available in literature that correspond to the conventional and self-healing methodologies.

Approximate circuits for multipliers [42, 48, 63, 69, 75, 83, 85, 101, 123] and adders [6, 39, 53, 74, 97, 114, 130] have been investigated for their pivotal role in digital signal processing architectures. As discussed in Chapter 2, approximate recursive multipliers have been designed for their benefits of low power consumption

and the possibility of fine-grained optimization based on the input distribution [63, 75, 101].

An $n \times n$ recursive multiplier utilizes elementary 2×2 multiplier modules. An approximate 2×2 multiplier (M_1) [63] features a lower complexity of the circuit (see Fig. 5.2b) as compared to the accurate design (M), see Fig. 5.2a. This brings a smaller chip-area, lower power, and lower latency of M_1 as compared to M . However, M_1 brings one error case out of sixteen possible input combinations ($3 \times 3 \mapsto 7$), where the $\text{error_rate} = 1/16$ for a uniformly distributed input and the $\text{error_magnitude} = 2$. Another approximate design, M_2 [101], also provides a lower complexity of the circuit as compared to M , while producing three error cases (Fig. 5.2c, 5.2d) with $\text{error_rate} = 3/16$ for a uniformly distributed input and $\text{error_magnitude} = 1$. M_1 has a higher error magnitude and a lower error rate as compared to M_2 , therefore M_1 can be regarded as a *fail-rare* design while M_2 as a *fail-small* design, and M_1 and M_2 correspond to the conventional approximate computing methodology.

To enable self-healing (*fail-balanced* design), we proposed M_p in Chapter 4 which is referred to as M_3 here. Fig. 5.2e and Fig. 5.2f show the truth table and logic diagram of M_3 . Noteworthy, M_3 is a *mirror* of M_1 , i.e., it produces an error case ($\epsilon = +2$) which is an additive inverse of M_1 ($\epsilon = -2$). Although M_3 requires more hardware as compared to M_1 , combining M_1 and M_3 in a pair has shown an overall effective quality-efficiency trade-off for square-accumulate architectures, see Chapter 4 for details.

In case of approximate MAC (xMAC) accelerators, approximate multipliers that produce near-to-zero² mean error provide the opportunity of error cancellation at the accumulation stage. A related approximate multiplier, DRUM, has been demonstrated for producing a near-to-zero mean error for a uniformly distributed input, by optimizing the widths of the input operands of a multiplier [42]. However, the applications that exhibit other input distributions (e.g., Gaussian) cannot utilize DRUM. On the other hand, approximate recursive multipliers can be optimized based on the input distribution but they do not exhibit a near-to-zero mean error by original design [42]. Interestingly, we demonstrate in Section 5.2.1 that they can be re-designed to achieve a near-to-zero mean error profile while retaining their primary benefits.

Truncated multiplication in a MAC architecture has also been studied [62, 94], where the primary aim is to restrict the bit-width of multipliers and produce low error MAC computations by diminishing the effects of truncation. Another approach for approximate MAC utilizes an offset compensation at the accumulation stage to alleviate the inaccuracies of the approximate multiplier stage [35]. In the context of redundant arithmetic units, which are based on the redundant number system, an approximate hybrid redundant MAC has been proposed that utilizes hybrid redundant adders to achieve high performance [31]. On the other

²Near-to-zero mean error refers to the mean error value that is zero or approximately zero.

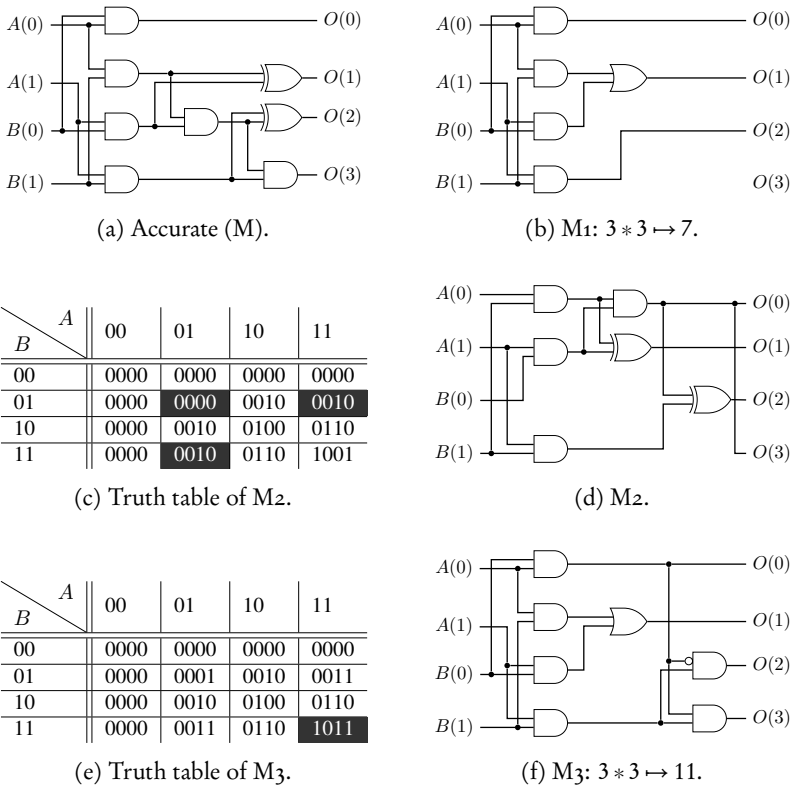


Figure 5.2: 2×2 Multiplier designs; M_1 [63] and M_2 [101] correspond to the conventional methodology, while M_3 corresponds to the self-healing methodology.

hand, however, the approximate hybrid redundant MAC has an overhead of binary-redundant and redundant-binary converters [31].

The key limitation of the above designs is that either they are based on the conventional error-restricted design methodology (e.g., M_1 and M_2 designs) or they have a restriction of parallel pairs (e.g., a pair of M_1 and M_3). As discussed earlier, the conventional error-restricted design methodology excludes the approximations that introduce high error magnitudes and high error rates, and therefore, limits the achievable efficiency gains. On the other hand, the restriction of utilizing parallel pairs limits the applicability of the designs when a datapath does not have similar computing elements in *multiples of two*. Moreover, in the context of approximate MAC accelerators, no exploitation of the self-healing methodology has been studied to the best of our knowledge.

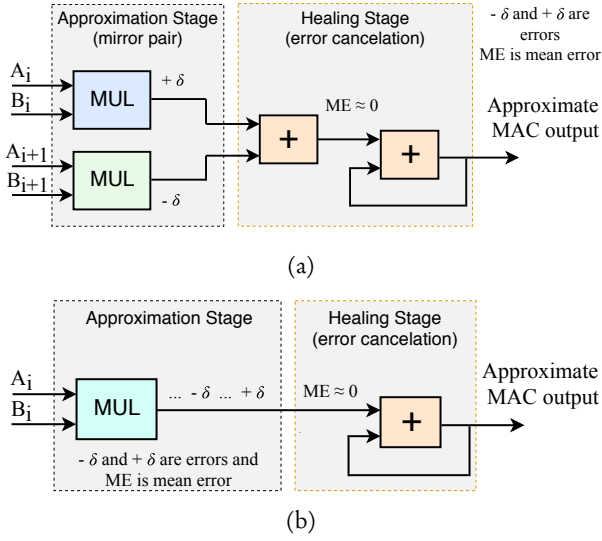


Figure 5.3: Approximate MAC designs, (a) utilizing the self-healing methodology, (b) utilizing the proposed ISH methodology, where approximation is achieved with $\pm\delta$ errors within a single multiplier module, which can be averaged out at the accumulator.

5.2 DESIGNING AN APPROXIMATE MAC WITH THE INTERNAL-SELF-HEALING (ISH) METHODOLOGY

A MAC operation computes,

$$\sum_{i=1}^N (A_i * B_i) \quad (5.1)$$

where A and B are the input vectors of length N . To design an approximate MAC (xMAC) in compliance with the self-healing methodology discussed in Chapter 4, the multiplication is considered as an *approximation stage* and the accumulation as a *healing stage*; see Fig. 5.3a. A pair of approximate multipliers is utilized such that they produce errors that are additive inverse of each other, i.e., $\epsilon_1 = +\delta$ and $\epsilon_2 = -\delta$, so that the expected value of the mean error approaches zero. This helps the accurate accumulator to cancel out the errors originated in the approximate multipliers. However, such a methodology is limited to architectures that have multiple multiplier pairs in parallel, which is not always the case as discussed earlier. Therefore, we propose an xMAC accelerator where an approximate multiplier can generate $+\delta$ and $-\delta$ errors internally, without requiring a parallel multiplier; see Fig. 5.3b. This relieves the restriction of *multiples of two* computing units and increases the applicability of the design.

5.2.1 APPROXIMATE MULTIPLIER FOR MAC

A key challenge in employing the ISH methodology for xMAC is to achieve an approximate multiplier that exhibits a near-to-zero mean error profile for a given input distribution, so that the subsequent accurate accumulator can average out the errors originated in the approximate multiplier. Here we discuss an approximate $n \times n$ unsigned recursive multiplier with the desired property, where n is the bit-width of input operands, $n \in \{2, 4, 8, 16, \dots\}$.

An $n \times n$ recursive multiplier is constructed using $(n/2)^2$ elementary (2×2) multipliers [63, 75, 101]. These 2×2 multipliers generate partial products. Summation of the bit-shifted partial products produce the overall output of an $n \times n$ recursive multiplier. Fig. 5.4 shows cases of 4×4 ($O_{4 \times 4}$) and 8×8 ($O_{8 \times 8}$) recursive multipliers that are composed of four and sixteen 2×2 multipliers, respectively. Any number out of the set of 2×2 multipliers and/or adders can be approximated to achieve an approximate multiplier [63, 101]. However, in this work we only apply approximations in the 2×2 multipliers (not the adders) as in Chapter 4. Therefore, any combination of approximate 2×2 multipliers, e.g., M_1 , M_2 and M_3 (Fig. 5.2), can be utilized to form an approximate $n \times n$ multiplier.

To achieve a near-to-zero mean error profile, the 2×2 multipliers that have equal numerical weights (see Fig. 5.4) can be approximated with $+\delta$ and $-\delta$ errors. For example, in case of a 4×4 multiplier, the output ($O_{4 \times 4}$) can be expressed as follows (see Fig. 5.4a),

$$O_{4 \times 4} = A_L * B_L + 4(A_L * B_H) + 4(A_H * B_L) + 16(A_H * B_H) \quad (5.2)$$

where the constants 4 and 16 are representing the shift factors. If M_1 is deployed for $A_L * B_H$, M_3 for $A_H * B_L$, and M for the other two, the expected mean error value of the multiplier (for uniformly distributed inputs) is zero. Therefore, an xMAC utilizing such an approximate multiplier has an expected error value of zero for uniformly distributed input vectors. Likewise, zero (or approximately zero) expected error value configurations can be chosen for other input distributions.

In the case of an 8×8 multiplier (see Fig. 5.4b), one possible way to achieve a zero mean error (for uniformly distributed inputs) is to approximate all the 2×2 multipliers of M_b with $+\delta$ (M_3) errors and all the 2×2 multipliers of M_c with $-\delta$ (M_1) errors. It is to be noted that there are multiple combinations of 2×2 multipliers in an $n \times n$ approximate multiplier that bring a zero (or approximately zero) mean error value. However, some of the combinations may overflow as discussed in the following section.

5.2.2 OVERFLOW HANDLING

A challenge for designing an $n \times n$ recursive multiplier with near-to-zero mean error is the requirement of positive error ($\epsilon = +\delta$) 2×2 approximate multipliers

$$\begin{array}{r}
 A: \underbrace{a_3 a_2}_{A_H} \underbrace{a_1 a_0}_{A_L} \\
 B: \underbrace{b_3 b_2}_{B_H} \underbrace{b_1 b_0}_{B_L}
 \end{array}
 +
 \begin{array}{c}
 \underbrace{O_{2 \times 2}} \\
 \begin{array}{cccc}
 \boxed{A_H * B_H} & 0 & 0 & 0 \\
 0 & \boxed{A_H * B_L} & 0 & 0 \\
 0 & \boxed{A_L * B_H} & 0 & 0 \\
 0 & 0 & 0 & \boxed{A_L * B_L}
 \end{array} \\
 \hline
 \underbrace{p_7 p_6 p_5 p_4 p_3 p_2 p_1 p_0}_{O_{4 \times 4}}
 \end{array}$$

(a) 4×4 recursive multiplication requires four 2×2 multipliers.

$$\begin{array}{r}
 A: \underbrace{a_7 a_6}_{A_{HH}} \underbrace{a_5 a_4}_{A_{HL}} \underbrace{a_3 a_2}_{A_{LH}} \underbrace{a_1 a_0}_{A_{LL}} \\
 B: \underbrace{b_7 b_6}_{B_{HH}} \underbrace{b_5 b_4}_{B_{HL}} \underbrace{b_3 b_2}_{B_{LH}} \underbrace{b_1 b_0}_{B_{LL}}
 \end{array}
 +
 \begin{array}{c}
 \underbrace{O_{2 \times 2}} \\
 \begin{array}{cccccccc}
 \boxed{A_{HH} * B_{HH}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & \boxed{A_{HL} * B_{HH}} & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & \boxed{A_{HH} * B_{HL}} & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & \boxed{A_{HL} * B_{HL}} & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & \boxed{A_{HH} * B_{LH}} & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & \boxed{A_{HH} * B_{LL}} & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & \boxed{A_{HL} * B_{LH}} & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & \boxed{A_{HL} * B_{LL}} & 0 & 0 \\
 0 & 0 & 0 & \boxed{A_{LH} * B_{HH}} & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & \boxed{A_{LH} * B_{HL}} & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & \boxed{A_{LL} * B_{HH}} & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & \boxed{A_{LL} * B_{HL}} & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & \boxed{A_{LH} * B_{LH}} & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & \boxed{A_{LH} * B_{LL}} & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{A_{LL} * B_{LH}} \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{A_{LL} * B_{LL}}
 \end{array} \\
 \hline
 \underbrace{p_{15} p_{14} p_{13} p_{12} p_{11} p_{10} p_9 p_8 p_7 p_6 p_5 p_4 p_3 p_2 p_1 p_0}_{O_{8 \times 8}}
 \end{array}$$

(b) 8×8 recursive multiplication requires sixteen 2×2 multipliers.

Figure 5.4: Recursive $n \times n$ multiplication utilizes elementary 2×2 multipliers. The same colors show equal numerical weight 2×2 multipliers that can be approximated with $+\delta$ and $-\delta$ errors to enable ISH.

like M_3 , which may result in the overall output exceeding the $2n$ bits. We define an *overflow configuration* as the configuration of an $n \times n$ multiplier (consisting of 2×2 multipliers) that overflows for at least one combination of n -bit inputs. Here we discuss how to identify the *overflow configurations* in order to discard them during design space exploration, and also propose a novel 2×2 approximate multiplier design that helps to alleviate the overflow problem.

Overflow Examples

Consider a 4×4 multiplication operation as shown in Fig. 5.4a. Let $A = (1111)_2$ and $B = (1111)_2$. This implies $A_H = A_L = B_H = B_L = (11)_2 = 3$, therefore Eq.

(5.2) becomes,

$$O_{4 \times 4} = 3 * 3 + 4(3 * 3) + 4(3 * 3) + 16(3 * 3)$$

assuming M_3 ($3 * 3 \mapsto 11$) is deployed for all 2×2 multipliers,

$$\begin{aligned} O_{4 \times 4} &= 11 + 4(11) + 4(11) + 16(11) \\ &= 275 = (1\ 0001\ 0011)_2 \end{aligned}$$

the output exceeds 8 bits ($2n$). Therefore the above example is an *overflow configuration* for a 4×4 multiplier, which is not desired. In case of a 4×4 multiplier, the overflow occurs as the value of the output is greater than 255, i.e., $2^{2n} - 1$. However, while constituting a higher order multiplier, say 8×8 multiplier, a 4×4 multiplier with an output value of less than 255 may also overflow the higher order multiplier. Note that 255 is still considerably larger than the maximum possible accurate output value of a 4×4 multiplier, which is 225 (i.e., $(2^n - 1)^2$). Consider an 8×8 multiplication (Fig. 5.4b), and let the constituting four 4×4 multiplications be represented by M_a , M_b , M_c and M_d such that the least significant multiplication is M_a while the most significant is M_d . The following expression represents the 8×8 computation,

$$O_{8 \times 8} = M_a + 16(M_b) + 16(M_c) + 256(M_d) \quad (5.3)$$

where the constants 16 and 256 are representing the shift factors. Let $A = (1111\ 1111)_2$ and $B = (1111\ 1111)_2$. Let M_3 ($3 * 3 \mapsto 11$), M_3 , M_1 ($3 * 3 \mapsto 7$) and M ($3 * 3 \mapsto 9$) compute the $A_L * B_L$, $A_L * B_H$, $A_H * B_L$ and $A_H * B_H$ partial products respectively for each of the 4×4 multipliers. Therefore, each of the 4×4 multipliers will generate,

$$O_{4 \times 4} = 11 + 4(11) + 4(7) + 16(9) = 227$$

and Eq. (5.3) becomes,

$$\begin{aligned} O_{8 \times 8} &= 227 + 16(227) + 16(227) + 256(227) \\ &= 65603 = (1\ 0000\ 0000\ 0100\ 0011)_2 \end{aligned}$$

the output exceeds 16 bits (i.e., $2n$), therefore this is an *overflow configuration*. So, even in cases where none of the 4×4 multipliers leads to overflow, the resulting 8×8 multiplier can cause overflow. In general, any $n \times n$ multiplier configuration that is not an overflow configuration in itself but has a maximum output value greater than $(2^n - 1)^2$, may overflow a higher order $2n \times 2n$ multiplier.

A Novel 2×2 Approximate Multiplier

To alleviate the overflow problem in an $n \times n$ approximate recursive multiplier, one way is to utilize an approximate 2×2 multiplier that provides a relatively

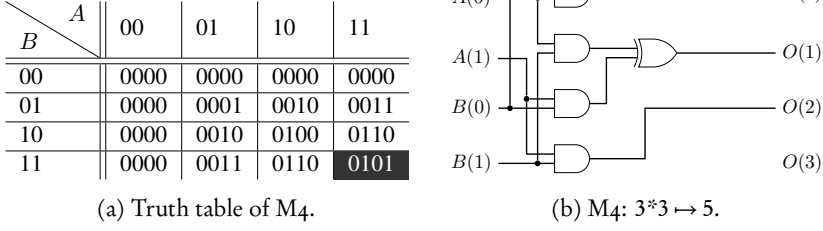


Figure 5.5: A proposed 2×2 approximate multiplier for overflow compensation.

larger negative error ($-\delta$). When such an approximate 2×2 multiplier is utilized, the maximum possible output value of an $n \times n$ approximate multiplier is reduced. In this regard, we propose an approximate 2×2 multiplier design (M_4), as shown in Fig. 5.5, which provides a larger negative error ($\epsilon = -4$) as compared to M_1 ($\epsilon = -2$). Note that M_4 can be balanced with two M_2 ($\epsilon = +2$) in order to achieve the internal-self-healing.

It should also be noted that M_4 is useful in the design of near-to-zero mean error recursive multipliers as it reduces the maximum possible output value of an $n \times n$ multiplier. For instance, if M_4 is employed to only $A_H * B_H$ in Eq. (5.2), it averts the possibility of overflow no matter which of the combination out of the given choices ($M/M_1/M_2/M_3/M_4$) is used for the other three 2×2 multipliers.

Overflow Handling Scheme

To identify the overflow configurations, we propose to assess each $n \times n$ multiplier configuration step-wise, from 4×4 constituting multipliers to an overall $n \times n$ multiplier. Consider an 8×8 recursive multiplication operation. For each 8×8 configuration, firstly, we need to check the possibility of an overflow for each of the four 4×4 multipliers. Let Γ_4 be the maximum possible output value of a 4×4 multiplier,

$$\Gamma_4 = \max(O_{4 \times 4}) \quad (5.4)$$

Then we need to check the following,

$$\Gamma_4 < 2^8 \quad (5.5)$$

If Eq. (5.5) fails for any of the four 4×4 multipliers, the configuration is discarded. Otherwise, we need to check the maximum possible output value of an overall 8×8 multiplier (Γ_8). Let $\Gamma_4(1)$, $\Gamma_4(2)$, $\Gamma_4(3)$, and $\Gamma_4(4)$ be the maximum possible output values of the constituting 4×4 multipliers and their respective shift factors be $S_4(1)$, $S_4(2)$, $S_4(3)$, and $S_4(4)$. Then the maximum possible output value of an 8×8 multiplier (Γ_8) is given as,

$$\Gamma_8 = \sum_{j=1}^4 [\Gamma_4(j) * S_4(j)] \quad (5.6)$$

now we need to check the following,

$$\Gamma_8 < 2^{16} \quad (5.7)$$

If Eq. (5.7) fails, the 8×8 multiplier is an overflow configuration. Likewise, additional steps can be added to identify overflow, or, to select non-overflow configurations for higher order recursive multipliers. For example, in the case of a 16×16 multiplier, Eq. (5.7) is to be checked for each constituting 8×8 multiplier and then the overall maximum possible output of the 16×16 multiplier is checked for an overflow.

To automate overflow handling for an $n \times n$ approximate multiplier configuration, a recursive function can be utilized (see Appendix C for details), where at each stage (n_r), the function checks the following condition for identifying valid configurations,

$$\Gamma_{n_r} < 2^{2n_r} \quad (5.8)$$

here n_r is the current recursive stage, $n_r \in \{4, 8, \dots, n/2, n\}$. The related maximum possible output value (Γ_{n_r}) can be computed as,

$$\begin{aligned} \Gamma_{n_r} &= \sum_{j=1}^4 [\Gamma_{(n_r/2)}(j) * S_{(n_r/2)}(j)] \\ &= \Gamma_{M_a} + 2^{n_r/2} \Gamma_{M_b} + 2^{n_r/2} \Gamma_{M_c} + 2^n \Gamma_{M_d} \end{aligned} \quad (5.9)$$

where $\Gamma_{M_a}, \Gamma_{M_b}, \Gamma_{M_c}$ and Γ_{M_d} are the maximum possible output values of the constituting $n_r/2 \times n_r/2$ multipliers (sub-multipliers).

5.2.3 COMPARISON OF THE PROPOSED ISH WITH THE CONVENTIONAL APPROXIMATE COMPUTING METHODOLOGY

Terminology and Notation

We follow the notation introduced in [G:7] and extend that to incorporate approximate recursive multipliers. Let \mathbb{I} be a set of inputs that is mapped to \mathbb{O} as the function f is executed in its exact form, i.e., $f : \mathbb{I} \mapsto \mathbb{O}$. Let $f^* : \mathbb{I} \mapsto \mathbb{O}^*$ and $f^{*'} : \mathbb{I} \mapsto \mathbb{O}^{*'}$ be the execution of the same function in approximate form by utilizing the conventional and the ISH methodologies, respectively. Let \mathfrak{D} be the design space offered by an approximate computing methodology, which is essentially a set of all possible design configurations offered by the respective methodology, i.e.,

$$\mathfrak{D} = \{C_1, C_2, C_3, \dots, C_g\} \quad (5.10)$$

here g is the number of design alternatives/configurations offered by the respective approximate computing methodology, and each C_i is a design configuration that characterizes a specific point: (e_i, q_i) in the quality-efficiency trade-off. Where e_i is efficiency and q_i is quality offered by C_i .

Supposing an $n \times n$ recursive multiplier, the function f corresponds to multiplication operation. Let \mathcal{D}^* and $\mathcal{D}^{*/}$ be the design spaces offered by the conventional and the ISH approximate computing methodologies respectively. The conventional approximate computing methodology utilizes the conventional error-restricted elementary (2×2) multipliers (M_1, M_2) along with the accurate version (M). Let \mathcal{R}^* be defined as the set of elementary multipliers utilized by the conventional approximate computing methodology, i.e., $\mathcal{R}^* = \{M, M_1, M_2\}$. On the other hand, the proposed ISH methodology utilizes the conventional and the proposed self-healing based elementary multipliers, therefore, $\mathcal{R}^{*/} = \{M, M_1, M_2, M_3, M_4\}$, where $\mathcal{R}^{*/}$ is a set of elementary multipliers utilized by the ISH methodology. It can be noted that all elements of \mathcal{R}^* are included in $\mathcal{R}^{*/}$, i.e., $\mathcal{R}^* \subset \mathcal{R}^{*/}$. Therefore,

$$\mathcal{D}^* \subset \mathcal{D}^{*/} \quad (5.11)$$

Comparison

To compare the trade-offs offered by two methodologies, we define *effectivity* (\mathcal{E}), such that \mathcal{E} is a function of quality and efficiency. A design methodology (with an effectivity of \mathcal{E}_1) is considered to be more effective than the other (with the effectivity of \mathcal{E}_2), i.e., $\mathcal{E}_1 > \mathcal{E}_2$, if and only if it provides a better efficiency (e) for a specific output quality (q), and a better quality (q) for a specific efficiency (e) target. As shown in Eq. (5.11), the design alternatives offered by the proposed ISH methodology include the design alternatives offered by the conventional methodology, and additionally, the ISH methodology offers new designs that help error cancellation. Consequently, the proposed ISH methodology provides a quality-efficiency trade-off that is always equally or more effective as compared to that of the conventional methodology counterpart, i.e.,

$$\mathcal{E}_{(f^{*/}:\mathbb{I} \rightarrow \mathbb{O}^{*/})} \geq \mathcal{E}_{(f^*:\mathbb{I} \rightarrow \mathbb{O}^*)} \quad (5.12)$$

Besides the overall trade-off, it is also important to analyze the error bounds of an approximate circuit that affect its feasibility for a target application. Marzek et al. [85] formalized the Worst Case Error (WCE) of a recursive multiplier as,

$$\text{WCE}_n = \text{WCE}_{M_a} + 2^{n/2} \text{WCE}_{M_b} + 2^{n/2} \text{WCE}_{M_c} + 2^n \text{WCE}_{M_d} \quad (5.13)$$

where WCE_n is the worst case error of an $n \times n$ recursive multiplier, and $\text{WCE}_{M_a}, \text{WCE}_{M_b}, \text{WCE}_{M_c}$ and WCE_{M_d} represent the worst case errors of the four constituting ($n/2 \times n/2$) multipliers (sub-multipliers) respectively. In case of an approximate multiplier that is designed in a conventional way, Eq. (5.13) represents the WCE that occurs when a worst case input triggers the error cases of all the approximate sub-multipliers. On the other hand, consider M_b and M_c are mirrored, such that they have error magnitudes that are additive inverse of each other, i.e., utilizing the proposed ISH methodology. If an input triggers an error case for each sub-multiplier, the second and third terms in Eq. (5.13) cancel

out. In fact, the WCE for such an ISH based approximate multiplier occurs when one of the M_b or M_c does not have an error triggering input and is given as,

$$\text{WCE}_n = \text{WCE}_{M_a} + 2^{n/2} \text{WCE}_{(M_b, M_c)} + 2^n \text{WCE}_{M_d} \quad (5.14)$$

where $\text{WCE}_{(M_b, M_c)}$ is the worst case error of M_b and M_c , which occurs when only one of them introduces an error, and the error has a same direction (sign) as that of M_a and M_d . Hence, the worst case error (WCE_n) of the ISH methodology can never be greater than that of the conventional methodology. Keeping in view the design space relation in Eq. (5.11), and the worst case errors for the conventional (see Eq. (5.13)) and the ISH (see Eq. (5.14)) methodologies, we have,

$$\text{WCE}_{(f^{**}: \mathbb{I} \rightarrow \mathbb{O}^{**})} \leq \text{WCE}_{(f^*: \mathbb{I} \rightarrow \mathbb{O}^*)} \quad (5.15)$$

From Eq. (5.12) and Eq. (5.15), it can be concluded that it is always beneficial to employ the proposed ISH methodology as compared to the error restricted conventional approximate computing methodology. It is to be noted that the aforesaid comparison is independent of the input distribution. Moreover, we quantify the benefits offered by the proposed ISH methodology in the subsequent sections.

5.3 EXPERIMENTAL RESULTS

To study the quality-efficiency trade-off for approximate MAC accelerators and to compare the proposed internal-self-healing (ISH) methodology with the conventional methodology, we have performed a design space exploration for area- and power-optimization for uniform and normal input distributions. As 8-bit architectures are widely used in signal processing applications [8, 52, 68, 111], our experiments are mainly focused on 8-bit designs. However, we also compare 4-bit and 16-bit designs to test the scalability of our methodology.

In Section 5.2.3, it has been shown that the proposed ISH methodology always provides better (or at least equivalent as a worst-case) designs as compared to the conventional approximate computing methodology. Here we present the cases that quantify the maximum benefits offered by the ISH methodology based on our design space exploration. This includes our case study of radio astronomy calibration processing (Section 5.3.4) and synthesis based comparison (Section 5.3.5) of quality-efficiency benefits.

5.3.1 EXPERIMENTAL SETUP

A quality analysis was performed using function-accurate behavioral implementations of accurate and approximate $n \times n$ recursive multipliers, and a hardware efficiency analysis was performed utilizing Synopsys Design Compiler and Power Compiler for the TSMC 40nm Low Power (TCBN40LP) technology library. A block diagram of our experimental setup is shown in Fig. 3.9 (Chapter 3).

To fix the latency budget of all the synthesized designs, a fixed operating frequency of 1 GHz has been utilized for hardware efficiency analysis. This legitimates the area and power comparison of various design alternatives to ensure a fair comparison. We have utilized the same compile command (*compile_ultra*³) for synthesizing all designs. Questasim has been utilized for functional verification and to generate the switching activity for power estimation. For normally distributed inputs, the following mean (μ) and standard deviation (σ) values have been considered⁴, 4-bit case: ($\mu = 8$, $\sigma = 1.5$), 8-bit case: ($\mu = 128$, $\sigma = 22.5$), and 16-bit case: ($\mu = 32768$, $\sigma = 6553$).

5.3.2 DESIGN SPACE EXPLORATION OF THE PROPOSED ISH METHODOLOGY

We have performed design space exploration as discussed in Appendix C to obtain the best designs offered by the ISH methodology. These best designs are referred to as the *pareto-optimal* configurations/designs, and the line joining the pareto-optimal points in the quality-efficiency trade-off is regarded as the *pareto front*.

One way of estimating the hardware costs of an $n \times n$ recursive multiplier is to add up the costs of the constituting sub-multipliers [G:3][85]. However, this ignores the hardware costs related to adder trees within an $n \times n$ multiplier. Therefore, the cost estimation proposed in [G:3][85] is useful for ranking purpose only, and has an underlying assumption that the costs of adder trees will follow the same trend as that of the sub-multipliers. In this chapter, we have utilized a better way of cost estimation that also includes the cost contributions of the adder trees related to the sub-multipliers. Firstly, we obtain the cost of an $n \times n$ multiplier composed of multiples of a unique 2×2 multiplier, say M_1 , using the Synopsys tool flow. Then we divide the cost of an $n \times n$ multiplier by the number of total 2×2 designs constituting an $n \times n$ multiplier. This includes the area/power costs of the 2×2 multipliers along with the related adders, and therefore provides a plausible estimation of hardware costs, or conversely: the hardware efficiency.

Table 5.1 shows the estimated hardware costs of the considered 2×2 multipliers that are utilized for estimating the costs of design configurations during the design space exploration. Note that an 8×8 multiplier needs relatively more adders as compared to a 4×4 multiplier to add the partial products. Therefore, each of the 2×2 multipliers in Table 5.1 has a lower cost estimate while constituting a 4×4 multiplier as compared to while constituting an 8×8 multiplier. Similarly,

³The compile command has been mentioned here for the reproducibility of our results. However, the conclusions of this chapter do not depend on this command.

⁴We mention the mean (μ) and standard deviation (σ) values for the reproducibility of our results. These μ and σ values have been chosen so that it is highly likely that the inputs remain less than 2^n , i.e., $\mu + 5\sigma < 2^n$. However, the conclusions of this chapter do not depend on these particular cases because using the same procedure the designs can be optimized based on the required μ and σ values, i.e., the input distribution of the target application.

Table 5.1: 2×2 multiplier cost (conversely: efficiency) estimation for TSMC 40nm Low Power library at 1 GHz. The estimation also includes the costs related to the adder trees within a higher order multiplier.

Design Type	4×4 Multiplier		8×8 Multiplier	
	Area (μm^2)	Power ^a	Area (μm^2)	Power ^a
M	21.52	13.41	32.43	27.59
M ₁	13.29	9.18	25.20	22.34
M ₂	19.17	10.16	31.11	22.06
M ₃	19.17	13.15	31.21	27.47
M ₄	16.76	10.27	27.36	22.66

^a Power(μW) estimates based on uniformly distributed input.

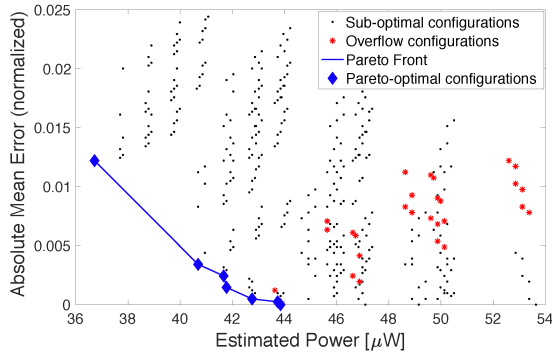
we have generated the results for a 16×16 case that has the same pattern in increase. We use these results in Section 5.3.3.

Fig. 5.6 shows the complete design space of a 4×4 recursive multiplier utilizing the five 2×2 multiplier options (M, M₁, M₂, M₃, and M₄), optimized for uniformly distributed input. The absolute mean error shown at the y-axis (in all our results of this Chapter) is normalized to the output range of the multiplier, i.e., 2^{2n} , where n is the bit-width of the input operands. Red asterisks represent the overflow configurations that are identified and discarded (using Eq. (5.8)).

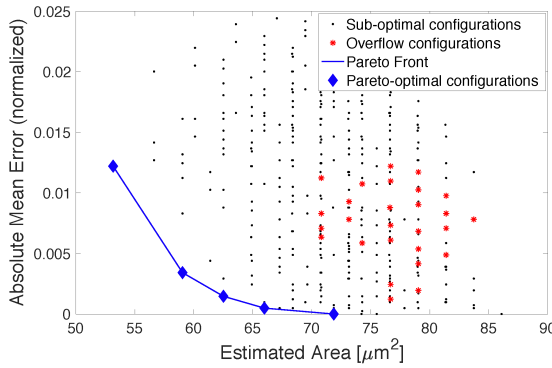
Table 5.2 shows the pareto-optimal configurations for a 4×4 recursive multiplier based on uniformly distributed input. The left column shows the power optimized pareto-optimal configurations and the right column shows the area optimized ones. Each configuration contains four 2×2 multipliers, e.g., M₁ M₁ M₁ M₁, where the left-most 2×2 multiplier is the Least Significant Multiplier (LSM) and the right-most one is the Most Significant Multiplier (MSM) of a 4×4 configuration. The hardware efficiency increases and the output-quality decreases as we go from the bottom row to the top row. It can be seen that most of the pareto-optimal configurations include the self-healing based designs like M₃ and M₄. This substantiates the importance of our proposed M₃ and M₄ designs. Similarly, we have generated the results for an 8×8 and a 16×16 case that have shown the same pattern and are discussed in the following sections.

5.3.3 SCALABILITY AND COMPARISON OF THE ISH WITH THE CONVENTIONAL METHODOLOGY

To compare the proposed ISH and the conventional approximate computing methodologies, we compare their pareto fronts for area- and power-optimized designs based on each input distribution (uniform and normal). As discussed in Section 5.2.3, all four approximate designs (M₁, M₂, M₃ and M₄) are considered (along with the accurate one (M)) as 2×2 multiplier options for the proposed



(a) Design space exploration for power optimization.



(b) Design space exploration for area optimization.

Figure 5.6: Quality-efficiency trade-off study of a 4×4 multiplier optimized for uniformly distributed inputs.

ISH methodology. However, only the conventional low error-rate (M_1) and low error-magnitude (M_2) approximate designs are considered (along with the accurate one (M)) for the conventional methodology.

Fig. 5.7 shows the pareto fronts for 4×4 recursive multipliers. It can be seen that the proposed ISH methodology presents many designs that have better efficiency for a given quality constraint and vice versa as compared to the conventional methodology counterparts. It should be noted that the additional design points (shown in Fig. 5.7b and 5.7d) are not worse as compared to the conventional methodology because they increase the number of design options in the pareto front. However, such designs may be ignored as they do not provide much efficiency benefits as compared to their decreased quality. Moreover, it should be noted that Fig. 5.7 shows pareto-optimal configurations based on exhaustive search, as the design space is small enough for a 4×4 multiplier case.

Table 5.2: Pareto-optimal configurations for a 4×4 recursive multiplier based on uniformly distributed input. The hardware efficiency increases and the output-quality decreases as we go from the bottom row to the top row.

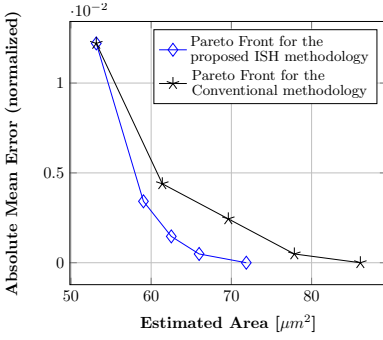
Power Optimization				Area Optimization			
LSM*	→	MSM*		LSM	→	MSM	
M ₁	M ₁	M ₁	M ₁	M ₁	M ₁	M ₁	M ₁
M ₁	M ₁	M ₁	M ₃	M ₁	M ₁	M ₁	M ₃
M ₁	M ₂	M ₁	M ₃	M ₁	M ₄	M ₁	M ₃
M ₁	M ₄	M ₁	M ₃	M ₁	M ₄	M ₄	M ₃
M ₁	M ₄	M ₂	M ₃	M ₄	M ₂	M ₄	M ₃
M ₂	M ₄	M ₂	M ₃	-	-	-	-
M ₄	M ₄	M ₂	M ₃	-	-	-	-

*LSM and MSM are the least significant and the most significant 2×2 multipliers respectively.

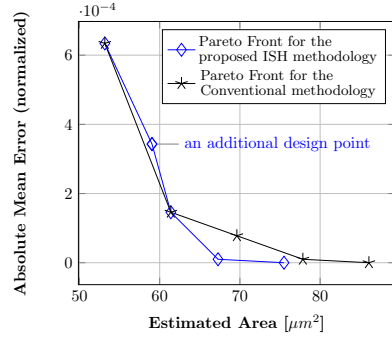
To verify the scalability of the proposed methodology, we also present a comparison for 8×8 and 16×16 recursive multipliers as shown in Fig. 5.8. Here the y-axis is shown in logarithmic scale to clearly illustrate the widely spread designs for comparison. It is to be noted that we have performed an exhaustive search for the 8×8 conventional methodology case, where the rest of the simulations for Fig. 5.8 (8×8 ISH and 16×16 conventional and ISH) have been performed by utilizing the intermediate-pruning technique discussed in Appendix C. It can be seen that the proposed ISH methodology clearly outperforms the conventional methodology for all considered input lengths by providing many designs that have better efficiency for a given quality constraint and vice versa.

In the case of the ISH methodology, it is noteworthy that the error drops relatively faster for increasing area/power costs in the beginning. This is because of error balancing that helps to reduce the error without using the accurate modules. However, at a certain stage, when the error is already very low, the rate of error drop (with respect to area/power) decreases (e.g., see Fig. 5.8b, designs: C6 and C7). This is because the usage of accurate modules is necessary to further reduce the error beyond this stage.

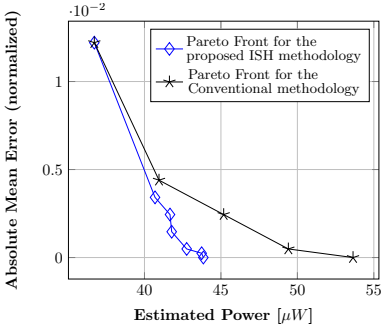
As discussed earlier, for 4×4 and 8×8 cases, exhaustive simulation is utilized (by considering all possible combinations of 2×2 multipliers) to obtain pareto-optimal designs based on the conventional methodology, which means no conventional design can be better than them. Although there are some approximations involved within the intermediate-pruning algorithm (see Appendix C) that may provide near-optimal (instead of optimal) designs in a rare case, it generates better ISH designs as compared to the conventional exhaustively searched designs. This substantiates the fact that the ISH methodology performs better than the conventional error-restricted methodology, including the higher order



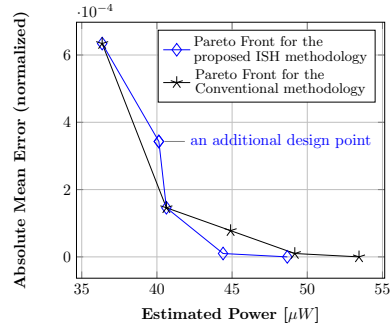
(a) Area optimization for uniformly distributed input.



(b) Area optimization for normally distributed input.



(c) Power optimization for uniformly distributed input.



(d) Power optimization for normally distributed input.

Figure 5.7: Comparison of the pareto-optimal designs of 4×4 multipliers based on ISH and the conventional approximate computing methodologies. The proposed ISH methodology outperforms for all considered optimization targets by providing better (or at least equal) efficiency designs for a given quality constraint and vice versa. For normally distributed input (b) and (d), additional design points are brought by the proposed ISH methodology, which are not worse as compared to the conventional methodology as they increase the number of design options in the pareto front.

input cases like 8-bit and 16-bit. Therefore we can conclude that the proposed ISH methodology provides a more effective quality-efficiency trade-off as compared to the conventional approximate computing methodology due to internal self-healing of the errors within the approximate modules, and this is independent of the target hardware efficiency (e.g., area or power), input width (e.g., 4-bit/8-bit/16-bit), and input distribution (e.g., uniform or normal).

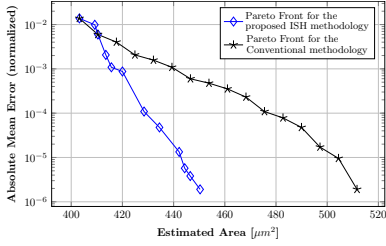
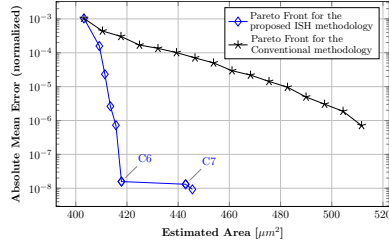
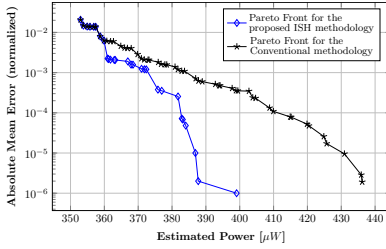
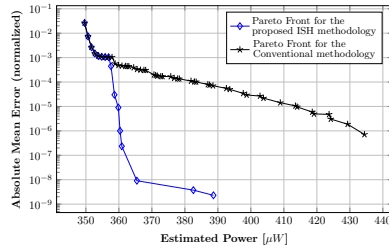
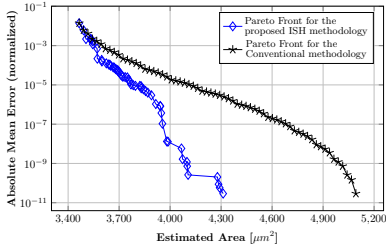
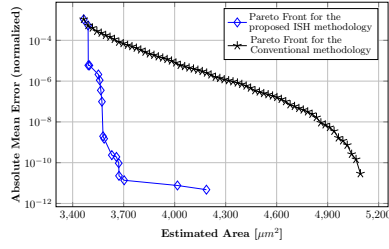
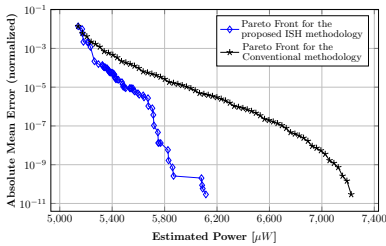
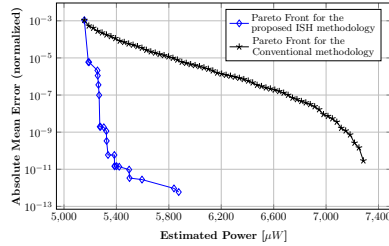
(a) Area optimization for uniformly distributed input (8×8).(b) Area optimization for normally distributed input (8×8).(c) Power optimization for uniformly distributed input (8×8).(d) Power optimization for normally distributed input (8×8).(e) Area optimization for uniformly distributed input (16×16).(f) Area optimization for normally distributed input (16×16).(g) Power optimization for uniformly distributed input (16×16).(h) Power optimization for normally distributed input (16×16).

Figure 5.8: Comparison of the pareto-optimal designs of 8×8 ((a)-(d)) and 16×16 ((e)-(h)) recursive multipliers based on the ISH and the conventional approximate computing methodologies. The proposed ISH methodology outperforms for all considered optimization targets.

So far, we have shown results for general input distributions in this Chapter. Here we present the improvements offered by the ISH methodology for an application. As discussed in Chapter 3, radio astronomy calibration estimates complex antenna gains within a radio telescope by utilizing an iterative method, known as StEFCal [107]. For a given configuration, i.e., the number of antenna elements and receiving channels in a radio telescope, StEFCal estimates the gains by utilizing measured visibilities (\mathbf{V}) and model visibilities (\mathbf{M}).

Considering a hardware accelerator design, StEFCal has three dominant kernels: complex-input element-wise product, complex-input square-accumulate and complex-input multiply-accumulate (MAC), see Fig. 3.7 in Chapter 3. Here we focus on the quality-efficiency trade-off of the complex-input MAC operation, which computes,

$$\sum_{j=1}^N \{z_j * v_j\} \quad (5.16)$$

where $v \in \mathbf{V}$. Also, $z \in \mathbf{Z}$ represents an element-wise product of model visibility vector and gain vector computed in the previous iteration, see Chapter 3 for details. We assume $N = 496$, which is the vector size for a radio telescope configuration of 124 antenna elements and 4 frequency channels.

It is to be noted that each complex multiplication requires four real-input multiplications. Therefore, in order to study the quality-efficiency trade-off, we have utilized pareto-optimal multipliers of the ISH and the conventional methodologies for all the four multiplications. Keeping in view the feasibility of 8-bit architectures in radio astronomy processing [111], we considered the MAC operation utilizing 8×8 multipliers to process StEFCal. We have utilized the radio astronomy calibration data of the LOFAR facility [124].

As shown in Eq. (5.12), the ISH methodology always provides better (or at least equivalent as a worst case) designs as compared to the conventional methodology. Therefore, here we present the cases that quantify the maximum benefits offered by the ISH methodology. Table 5.3 shows equivalent-efficiency⁵ designs for the area and power optimization. It can be seen that the area-optimized design of the ISH methodology (ISH_A) brings 27% improvement of the Mean Square Error (MSE) as compared to the equivalent-efficiency conventional methodology design (Conven_A). For power optimized designs, ISH methodology (ISH_P) offers 55% improvement in quality as compared to the conventional methodology counterpart (Conven_P).

Table 5.3 also shows the power costs of area-optimized designs (Conven_A and ISH_A), and area costs for power-optimized designs (Conven_P and ISH_P).

⁵The equivalent-efficiency designs have equal (or approximately equal) targeted efficiency, or inversely, the computational cost. For example, in the case of area optimized designs, Conven_A and ISH_A are equivalent-efficiency designs because they both consume equal chip-area, see Table 5.3.

Table 5.3: Employing equivalent-efficiency approximate MAC alternatives in radio astronomy calibration. The proposed designs (ISH_A and ISH_P) exhibit 27% and 55% better quality as compared to the conventional methodology counterparts (Conven_A and Conven_P).

Design Alternatives	MAC Error (MSE)	Hardware Cost*
Accurate	0	A = 519, P = 439
Conven_A	1.96e-02	A = 447 (P ^s = 389)
ISH_A	1.44e-02	A = 447 (P ^s = 384)
Conven_P	2.01e-02	P = 383 (A ^s = 445)
ISH_P	9.07e-03	P = 383 (A ^s = 472)

* A and P are Area (μm^2) and Power (μW) estimates (respectively) of each multiplier in a complex-input MAC accelerator.

^s These A and P costs are not the primary optimization targets.

Although these power and area costs are not the *primary optimization targets*⁶, it is important to note that they can introduce an additional trade-off. For instance, the conventional methodology design Conven_A consumes slightly more power as compared to the ISH counterpart (ISH_A); on the other hand, the conventional methodology design Conven_P requires less area as compared to the ISH counterpart (ISH_P). Moreover, it is to be noted that in case of the ISH methodology, the power-optimized designs (e.g., ISH_P) tend to utilize more M2 multipliers as they are cheap in power, on the other hand, they require more area (see Table 5.1). Keeping in view the aforesaid comparisons, it is important to optimize a design based on the efficiency target defined by the requirement specifications of an application.

This case study shows that utilizing the ISH methodology brings better quality for a given hardware cost (or efficiency target) as compared to the conventional methodology. However, in order to employ approximate modules optimized for radio astronomy processing, a comprehensive study of error resilience and input distribution of representative data sets is required.

5.3.5 SYNTHESIS BASED COMPARISON

As yet, we have shown quality-efficiency improvements offered by the proposed ISH methodology based on estimated hardware cost models discussed in Section 5.3.2. Here we present the results to quantify maximum hardware improvements based on the synthesis of complete units. We have synthesized 8×8 pareto-optimal multiplier designs that are optimized for normally distributed input. Table 5.4 shows the area and power consumption of the considered designs while their design configurations are shown in Table 5.5. Conven_1 and Conven_2

⁶For example, area optimized designs have a primary optimization target of minimum chip-area, and power optimized designs have a primary optimization target of minimum power consumption.

Table 5.4: Synthesis based comparison of the considered pareto-optimal configurations for an 8×8 recursive multiplier. The proposed ISH methodology provides more effective quality-efficiency designs, e.g., ISH_1 shows 18% better area and 14% better power as compared to Conven_2, also with a better quality output.

Design Alternatives	Quality-Efficiency		
	Error*	Area*	Power*
Accurate	0	519	439
Conven_1	2.95e-5	455	387
Conven_2	1.87e-6	507	414
ISH_1	1.57e-8	415	356
ISH_2	9.26e-9	464	373

* Unit of Area is μm^2 . Power (μW) is obtained utilizing switching activity based on normally distributed input. Normalized absolute mean error is considered as an Error.

designs are based on the conventional approximate computing methodology. ISH_1 and ISH_2 designs are based on the proposed ISH methodology.

In Table 5.4, as can be expected, hardware efficiency increases as we compromise on the quality of output. For example, Conven_1 is more efficient (requires less area and power) as compared to Conven_2 and Accurate designs. On the other hand, Conven_2 and Accurate designs have a better quality of output as compared to Conven_1. Similarly, the efficiency of ISH_1 is higher than ISH_2. In Table 5.5, it can be seen that the designs based on the proposed ISH methodology tend to utilize M3 and M4 as approximate 2×2 multiplier options, which helps for error cancellation and avoiding overflow situation at the same time. It can also be noted that the approximate 8×8 designs tend to utilize an approximate 2×2 multiplier (M1/M4) at the most significant position. This is because of a very low probability of error for the considered normally distributed input ($\mu = 128$, $\sigma = 22.5$), i.e., for the aforesaid characteristics of the input, it is hardly likely that both inputs of a 2×2 multiplier (placed at the most significant position) are 3.

Interestingly, while comparing quality and efficiency of designs for the proposed ISH and the conventional methodologies (see Table 5.4), the ISH designs (ISH_1 and ISH_2) provide a higher area-/power-efficiency and a higher output quality as compared to that of the conventional counterparts (Conven_1 and Conven_2). Specifically, ISH_1 exhibits 18% better area and 14% better power as compared to Conven_2, and at the same time, it provides a better quality of output (less error) due to the error cancellation mechanism intrinsic to the ISH methodology.

Table 5.5: Pareto-optimal configurations for an 8×8 recursive multiplier considered in Table 5.4.

Design Alternatives	Multiplier configurations optimized for normally distributed input														MSM*			
	→																	
	LSM*														MSM*			
Accurate	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
Conven_1	M ₁	M ₁	M ₁	M	M ₁	M	M ₁	M	M ₁	M	M ₁	M ₁	M ₁	M ₁	M ₁	M ₁	M ₁	M ₁
Conven_2	M	M	M	M	M ₁	M	M	M	M	M	M	M	M	M	M	M	M	M ₁
ISH_1	M ₄	M ₁	M ₁	M ₁	M ₁	M ₁	M ₄	M ₁	M ₁	M ₁	M ₁	M ₁	M ₁	M ₁	M ₃	M ₄	M ₁	M ₄
ISH_2	M ₄	M ₁	M ₁	M	M ₄	M ₄	M ₃	M ₁	M ₁	M	M ₄	M ₁	M	M ₁	M	M ₁	M ₃	M ₁

* LSM and MSM are the least significant and the most significant 2×2 multipliers respectively.

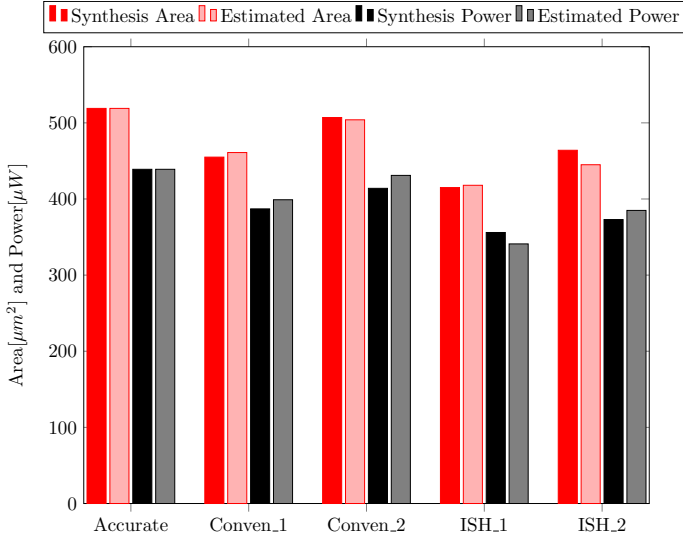


Figure 5.9: Comparison of the model based estimated cost and the synthesis based cost. The difference varies from 0-4% for both area and power costs.

5.3.6 DISCUSSION AND FUTURE WORK

In Section 5.3.2, we have performed a design space exploration to obtain the best designs offered by the conventional and the proposed ISH methodologies. For computing the computational cost (area and power consumption) of each considered design, we have utilized a cost estimation model discussed in Section 5.3.2. In Section 5.3.5, we have synthesized some designs obtained from the design space exploration as complete units. To assess the validity of the cost estimation model utilized in Section 5.3.2, we can compare its *estimated* costs with that of the *synthesis* based hardware costs. Fig. 5.9 shows the differences in area and power costs that vary from 0-4% for the considered designs. Although the number of cases considered is not sufficient to draw strong conclusions, Fig. 5.9 shows that the estimation model provides a useful approach to rank the designs cost-wise in order to find the pareto-optimal configurations during the design space exploration process. Especially because it's not possible to synthesize the millions of designs to obtain the synthesis-based hardware costs.

We have shown that utilizing the proposed ISH methodology provides more quality-efficiency benefits for the radio astronomy calibration processing as compared to the conventional approximate computing methodology. Nevertheless, before employing the proposed approximate computing designs to any application, it is important to assess the error resilience intrinsic to the application based on the error characteristics of the approximate designs. For example, the error profile of an approximate MAC accelerator is shown in Fig. 5.10, which

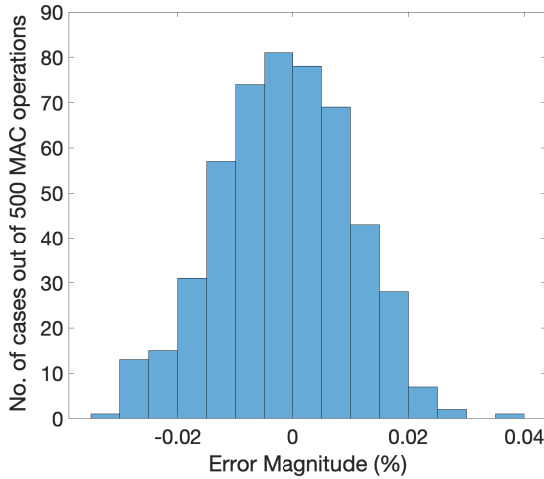


Figure 5.10: Error profile of an approximate MAC accelerator that employs the ISH₁ multiplier design. The error profile is generated based on normally distributed input ($\mu = 128$, $\sigma = 22.5$).

employs the ISH₁ multiplier design. It would be interesting to investigate the introduction and propagation of such an error profile in the signal processing pipeline of the target application, e.g., the science data processing (SDP) pipeline of the SKA radio telescope.

5.4 CONCLUSIONS

In this chapter, an Internal-Self-Healing (ISH) methodology is presented for approximate MAC accelerators, which utilizes the proposed approximate recursive multipliers with a near-to-zero mean error profile. In contrast to the self-healing methodology (discussed in Chapter 4), the proposed ISH methodology has shown an opportunity for error cancellation within approximate circuits, without requiring similar computing elements (or parts of a datapath) in *multiple of two*.

We have presented a quantitative analysis based on a design space exploration of 4-bit, 8-bit and 16-bit designs for area and power optimization, considering uniform and normal input distributions, and radio astronomy calibration processing. Our results showed a more effective quality-efficiency trade-off offered by the ISH methodology as compared to the conventional approximate computing methodology. By definition, the proposed ISH methodology also considers the conventional designs in addition to the novel designs (for error cancellation) during the design space exploration. Therefore, it always generates better or at

least equivalent designs with a higher *effectivity* (based on quality-efficiency trade-off) and a lower *worst-case error* as compared to the conventional methodology.

6

CONCLUSIONS AND RECOMMENDATIONS

Approximate computing techniques have helped designers to exploit error tolerance to achieve increased hardware efficiency, especially in combating power/energy-challenges. Nonetheless, the state-of-the-art techniques do not sufficiently address accelerator designs for iterative and accumulation based algorithms despite the fact that they are widely used in digital signal processing. This thesis is an effort to contribute towards high-efficiency accelerator designs for such algorithms. Based on our contributions, here we address the research questions raised in Chapter 1 and provide the future directions of our research.

6.1 CONTRIBUTIONS

6.1.1 ERROR RESILIENCE ANALYSIS OF ITERATIVE ALGORITHMS

Our first research question directs to investigate how to analyze iterative algorithms for error resilience, and how trustworthy is a precision-based quality metric (convergence) in the error resilience analysis process. As discussed in Chapter 3, error resilience analysis provides a means to assess if an algorithm is amenable to approximate computing techniques. The state-of-the-art Statistical Approximation Model (SAM) helps to inject Gaussian randomness during the processing of an algorithm. While employing SAM, the output quality of the algorithm is monitored to assess if it is error-resilient. In the case of iterative algorithms, our simulation results show that employing SAM is not sufficient. Our proposed Adaptive Statistical Approximation Model (Adaptive-SAM) provides an opportunity to divide an iterative algorithm into approximate and accurate parts. It quantifies the number of initial iterations that are error-resilient.

We have presented the error resilience analysis of radio astronomy calibration processing, which utilizes a Least Squares (LS) algorithm to estimate sensor gains within a radio telescope. While employing SAM, the algorithm demon-

strated resilience for an error magnitude that is 0.001% to 0.003% of the output values. Showing that there is a negligible error resilience intrinsic to the algorithm. However, while employing the proposed Adaptive-SAM, the algorithm showed resilience for an error magnitude that is 11% to 13% of the output values for the first 23% of the iterations. Therefore, we can conclude that our proposed Adaptive-SAM analysis reveals approximation opportunities in iterative algorithms.

We have investigated if the precision-based quality metric, the convergence criterion, is sufficient in the error resilience analysis process. While applying two different approximations in an iterative algorithm, our simulation results show the possibility that the solution is not acceptable while the convergence criterion is satisfied and vice versa. Therefore, the precision-based quality metric is not trust-worthy in the error resilience analysis process. To resolve this problem, we defined an additional accuracy-based quality metric to assess the viability of approximation techniques promising for an iterative algorithm.

6.1.2 EXPLOITING ERROR RESILIENCE OF ITERATIVE ALGORITHMS

Our second research question deals with investigating how to exploit the intrinsic error resilience of iterative algorithms effectively. In this regard, a heterogeneous architecture (composed of accurate and approximate computing cores) has been presented in Chapter 3. Our proposed approximate LS accelerator design processes the initial iterations on a low-precision (approximate) core and the later ones on an optimized-precision (accurate) core. In a case study of radio astronomy calibration processing, the proposed accelerator design has shown 23.4% of energy savings while providing an acceptable quality output within an equal number of iterations as compared to that of an accurate counterpart.

6.1.3 DESIGNING APPROXIMATE ACCELERATORS FOR ACCUMULATION BASED ALGORITHMS

Our third research question has led us to investigate how to design high-efficiency approximate accelerators for accumulation based algorithms. In accumulation based algorithms like MAC (or SAC), we have investigated the design of such approximate multipliers (or squarers) that bring an opportunity to cancel out errors within accumulation without the overhead of error correction circuitry. We have shown that high-efficiency approximate accelerators like square-accumulate (SAC) and multiply-accumulate (MAC) can be designed by utilizing the proposed approximate computing methodologies. Building on the observation that these accelerators accumulate the outputs of squarers and multipliers in the case of SAC and MAC respectively, we have proposed approximate squarers/multipliers that generate a near-to-zero mean error for a given input distribution. Hence providing an opportunity to cancel out the errors (originated in the approximate squarer/multiplier) at the accumulation stage. Noteworthy, this error

cancellation does not require the overhead of error detection and correction circuitry.

In Chapter 4, a self-healing (SH) methodology is proposed for designing a SAC accelerator. In the case of signed inputs, the conventional truncation techniques provide positive and negative errors for squarer/multiplier operations. However, we have shown that they do not provide a near-to-zero mean error if employed naively. In the case of squarers, the output is always a positive number, and therefore, the output quality is not compromised if we invert the signs of the inputs. We proposed to employ an approximate squarer *mirror pair*, composed of two squarers, S_1 and S_2 . Where S_1 squares the truncated input numbers that are made negative (before truncation) and S_2 squares the truncated input numbers that are made positive (before truncation). Our simulation results show that the proposed *mirror pair* provides significantly better quality for a SAC accelerator as compared to the conventional way of truncation.

For unsigned inputs, we proposed logic-pruned approximate squarers that cancel out the error completely (for a given input distribution) while forming a *mirror pair*. We proposed 2×2 squarers/multipliers that are utilized to construct a higher order ($n \times n$) squarer *mirror pair*. With randomly distributed finite-length input data, we have shown that an approximate SAC accelerator based on our proposed squarer *mirror pair* provides a more effective quality-efficiency trade-off as compared to that of the conventional error-restricted approximate computing methodology. Specifically, our proposed squarer *mirror pair* demonstrates up to 25% and 18.6% better area and power efficiency respectively with a better-quality output as compared to the conventional counterpart.

The proposed SH methodology is however restricted to *pairs of two* (or *multiples of two*) computing elements for error cancellation. In Chapter 5, we have presented an internal-self-healing (ISH) methodology for a MAC accelerator that introduces self-healing within a single multiplier. The ISH methodology relieves the *pairs of two* restriction for computing elements and enables error cancellation within a single computing element. Our MAC accelerator employs the proposed approximate recursive multipliers with a near-to-zero mean error profile. We have introduced novel 2×2 multipliers that allow the construction of $n \times n$ multipliers with characteristics of near-to-zero mean error while guaranteeing the non-overflow operation.

Our scalability analysis of designing 4×4 , 8×8 , and 16×16 approximate multipliers for a MAC accelerator shows that the proposed ISH methodology provides a more effective quality-efficiency trade-off as compared to that of the conventional approximate computing methodology. Noteworthy, input cases utilized for the aforesaid analysis include uniformly and normally distributed random data and radio astronomy calibration data. Moreover, we have shown that our ISH methodology provides a lower *worst-case error* as compared to the conventional methodology while offering an increase in *area efficiency* and *power efficiency*.

Our fourth research question has led us to perform a case study of radio astronomy processing, specifically to investigate how do the proposed approximate computing methodologies affect the quality and efficiency of the processing and what are the opportunities and challenges to embrace approximate computing principles for radio astronomy processing. As the Science Data Processing (SDP) pipeline of a radio telescope is dominated by accumulation based iterative processes that also utilize approximate algorithms like LS, our proposed methodologies provide opportunities for energy-/power-efficient radio astronomy processing. The heterogeneous architecture presented in Chapter 3 shows 23.4% of the reduction in energy consumption for radio astronomy calibration processing.

Moreover, the self-healing methodologies presented in Chapter 4 and Chapter 5 provide promising solutions for low-power SAC and MAC accelerators required for the SDP pipeline. By assessing equivalent-efficiency designs, we have shown that the proposed self-healing methodologies bring much better output quality for the radio astronomy calibration processing as compared to the conventional approximate computing methodologies. Nevertheless, to embrace approximate computing principles, the challenges include quantifying the error resilience intrinsic to the SDP pipeline based on a representative data set and investigating various components of the SDP (like gridding and de-gridding) as discussed in Section 6.2.

6.2 RECOMMENDATIONS FOR FUTURE WORK

Based on our research on iterative and accumulation based algorithms, we believe that the following research directions are interesting to further enhance/broaden the benefits of employing approximate computing techniques.

The self-healing (SH) methodology presented in Chapter 4 has been demonstrated by designing an approximate SAC accelerator. However, in principle, the SH methodology can also be applied to design an approximate MAC accelerator or an approximate MAC array processor having several parallel MAC operations, wherein an approximate multiplier *mirror pair* is utilized to introduce a near-to-zero mean error that can be compensated at the accurate accumulation stage. Similarly, the internal-self-healing (ISH) methodology presented in Chapter 5 can be applied to approximate SAC accelerators. Therefore, it would be interesting to compare the quality-efficiency trade-offs offered by the aforesaid methodologies for designing SAC and MAC accelerators.

In Chapter 4 and Chapter 5, we have shown how unsigned multipliers/squarers can be designed to introduce error cancellation within the accumulation based algorithms. Such designs can handle the applications that have unsigned inputs, and the applications that have signed inputs in signed-magnitude representation.

On the other hand, in case of applications having signed inputs in 2's complement representation, we have provided a proof of concept by analyzing their error behavior for a truncated squarer *mirror pair*. It is important to note that the truncated squarer *mirror pair* does not cancel out the error completely, see Fig. 4.4 for an example. Secondly, inverting signs of the input may introduce hardware overheads depending upon the data representation. Nevertheless, it would be interesting to investigate the quality-efficiency trade-off offered by the truncated signed squarers/multipliers within the accumulation based algorithms and compare it with that of the other approximate signed multiplier designs like hybrid high-radix encoded booth multipliers [69].

Although the proposed ISH and SH methodologies have been demonstrated by designing approximate recursive multipliers/squarers, potentially, they can be utilized to design/modify other multiplier structures like booth multipliers. Secondly, we have introduced approximations at the *logic level*, however, applying approximations at the *standard cell level* is also promising [27]. Therefore, it would be interesting to apply (and compare) the proposed error cancellation based approximations at standard cell level. Moreover, comparative studies have been done by considering the state-of-the-art approximate multipliers for general algorithms [50]. Similarly, it is important to carry out a comparative study of approximate multipliers for the subsequent accumulation, i.e., a comparative study of the approximate multipliers that are to be used in accumulation based algorithms.

In the context of radio astronomy processing, it is important to quantify the intrinsic error resilience present in the SDP pipeline to embrace the approximate computing principles discussed in Section 6.1. This requires to define a representative data-set of a radio telescope that can be analyzed for the introduction and propagation of errors within the SDP pipeline due to the employment of approximate computing techniques. For instance, an approximate unsigned multiplier design (ISH_1) utilizing the ISH methodology is presented in Chapter 5. It brings 19% better power efficiency as compared to an accurate counterpart. When utilized in a MAC accelerator, its error profile is shown in Fig. 5.10. It would be interesting to investigate if such an error profile is acceptable for the SDP pipeline to achieve the power efficiency benefits. Moreover, investigating ISH based signed multiplier designs would be useful for their direct application within the SDP pipeline.

Furthermore, the approximate LS accelerator has shown a reduction in energy consumption based on a limited data set of the LOw-Frequency ARray (LOFAR) [124]. As already discussed, a well-defined representative data set can be utilized to further tune the approximate core in the heterogeneous architecture to quantify the overall energy-efficiency gains. Noteworthy, although the accelerator designs presented in this thesis (MAC, SAC and LS) provide reasonable efficiency benefits for ASIC based implementations, they may not offer an equal extent of efficiency benefits for FPGA based architectures. In this regard, our designs may

need modifications for optimized FPGA based implementations. Or conversely, it would be interesting to investigate how the existing approximate computing techniques (optimized for FPGAs like in [32, 97]) can be modified to employ self-healing approximations for high-efficiency FPGA based accelerator designs targeting the iterative and accumulation based algorithms.

Finally, *gridding* and *de-gridding* are compute-intensive parts of the SDP pipeline, which help to map the non-uniform visibilities to the Fourier transform of the sky images [126]. These algorithms also require multiply-add operations that can be investigated for self-healing approximate computing techniques.

A

8 × 8 SQUARER CONSTRUCTION

As discussed in Chapter 4 (Section 4.4), an 8 × 8 squarer (Sq8x8) computes the A*A multiplication, where A is an 8-bit unsigned input. Let $A = a_7a_6 a_5a_4 a_3a_2 a_1a_0$, where a_7 is the most significant bit, and a_0 is the least significant bit. Therefore,

$$\begin{aligned} \text{Sq8x8} &= a_7a_6 a_5a_4 a_3a_2 a_1a_0 * a_7a_6 a_5a_4 a_3a_2 a_1a_0 \\ &= a_3a_2 a_1a_0 * a_3a_2 a_1a_0 + 16(a_3a_2 a_1a_0 * a_7a_6 a_5a_4) \\ &\quad + 16(a_7a_6 a_5a_4 * a_3a_2 a_1a_0) + 256(a_7a_6 a_5a_4 * a_7a_6 a_5a_4) \end{aligned} \quad (\text{A.1})$$

Eq. (A.1) shows that the total of four 4 × 4 partial products compute the Sq8x8 operation as illustrated in Fig. 4.6 (left). However, two of the 4 × 4 partial products multiply the same (equal) inputs. Therefore, Eq. (A.1) can be rewritten as,

$$\begin{aligned} &= a_3a_2 a_1a_0 * a_3a_2 a_1a_0 + 32(a_3a_2 a_1a_0 * a_7a_6 a_5a_4) \\ &\quad + 256(a_7a_6 a_5a_4 * a_7a_6 a_5a_4) \end{aligned} \quad (\text{A.2})$$

which means that three 4 × 4 partial products can compute the Sq8x8 operation, where the factors 32 and 256 are implemented as bit shifts as shown in Fig. 4.7. Moreover, each 4 × 4 partial product can be further decomposed into basic 2 × 2 partial product constructs as shown in Fig. 4.6 (right). Therefore, Eq. (A.2) becomes,

$$\begin{aligned} &= a_1a_0 * a_1a_0 + 4(a_1a_0 * a_3a_2) + 4(a_3a_2 * a_1a_0) + 16(a_3a_2 * a_3a_2) \\ &\quad + 32[a_1a_0 * a_5a_4 + 4(a_1a_0 * a_7a_6) + 4(a_5a_4 * a_3a_2) + 16(a_3a_2 * a_7a_6)] \\ &\quad + 256[a_5a_4 * a_5a_4 + 4(a_5a_4 * a_7a_6) + 4(a_7a_6 * a_5a_4) + 16(a_7a_6 * a_7a_6)] \end{aligned} \quad (\text{A.3})$$

This Appendix is based on [G:3].

By combining the same (equal) 2×2 partial product operations,

122

$$\begin{aligned}
 &= a_1 a_0 * a_1 a_0 + 8(a_1 a_0 * a_3 a_2) + 16(a_3 a_2 * a_3 a_2) + 32[a_1 a_0 * a_5 a_4 \\
 &\quad + 4(a_1 a_0 * a_7 a_6 + a_5 a_4 * a_3 a_2) + 16(a_3 a_2 * a_7 a_6)] \\
 &\quad + 256[a_5 a_4 * a_5 a_4 + 8(a_5 a_4 * a_7 a_6) + 16(a_7 a_6 * a_7 a_6)] \quad (A.4)
 \end{aligned}$$

Eq. (A.4) shows total of ten 2×2 partial products, wherein four are Sq2x2 and six are P2x2 (also depicted in Fig. 4.7). The adders and shift factors are implemented in the higher order blocks like P4x4, Sq4x4 and Sq8x8 shown in Fig. 4.7.

B

QUALITY EVALUATION FOR APPROXIMATE SQUARERS

To evaluate the quality of all the possible configurations of an $n \times n$ approximate squarer, a tool¹ is utilized that computes the *probability mass function of error* (PMFe) [74, 75] for an approximate squarer configuration, provided an input distribution. We consider an approximate $n \times n$ squarer (composed of elementary 2×2 multiplier and squarer modules) designed in a conventional (approximate) way. As discussed in Chapter 4, the conventional way utilizes M₁ as an approximate 2×2 multiplier option and S₁ as an approximate 2×2 squarer option, without the *mirror* approximate designs (Mp and S₂).

Let n be the number of input bits and $n \in \{4, 8, 16, 32, \dots\}$. Based upon the fact that the elementary modules used in this work are 2×2 multipliers and 2×2 squarers, the input can be divided into pairs of bits, starting from the LSB, as shown in Fig. B.1. Each input pair can attain one of three possible states, which defines whether the particular state will/can/will not lead to error(s) in the corresponding 2×2 approximate elementary module(s). The states are defined as follows:

- » State 0 : If in this state, the input pair will ensure that there is (are) no error(s) in the corresponding elementary module(s).
- » State 1 : If in this state, the input pair will generate an error in the corresponding approximate 2×2 squarer module only.
- » State 2 : If in this state, the input pair can only generate an error in the corresponding approximate 2×2 multiplier module(s), depending on the state of the other pair(s) of bits being used by the corresponding elementary module(s).

This Appendix is based on [G:3].

¹This tool has been developed with the CARE-Tech (ECS group), TU Wien where the implementation was done by that group.

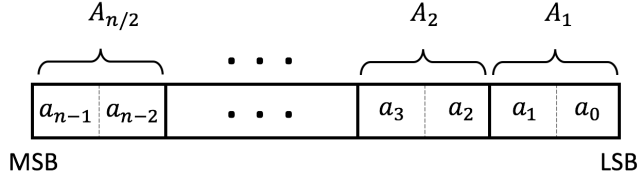


Figure B.1: Pairing of bits for an n -bit input. $A_1, A_2, \dots, A_{n/2}$ are utilized by elementary 2×2 modules in an $n \times n$ squarer.

It can be inferred from the designs of the approximate elementary modules (M_1 and S_1), presented in section 4.4, that an input pair will be considered in State 0 if it has either the value $(00)_2$ or $(01)_2$, as in all the designs these values do not lead to any approximation error. An input pair is considered in State 1 if it has the value $(10)_2$, as this will always generate an error in case of an approximate 2×2 squarer (S_1) being used at the corresponding location, see Section 4.4.1. Similarly, an input pair is considered in State 2 if it has the value $(11)_2$, as this might generate an error in a 2×2 approximate multiplier (M_1) module, depending on the state of the other pair of bits being input to the module.

Given an input probability distribution, the probabilities of each state can be computed as follows:

» State 0 :

$$P(s(A_i) = 0) = P(A_i == (00)_2) + P(A_i == (01)_2) \quad (\text{B.1})$$

» State 1 :

$$P(s(A_i) = 1) = P(A_i == (10)_2) \quad (\text{B.2})$$

» State 2 :

$$P(s(A_i) = 2) = P(A_i == (11)_2) \quad (\text{B.3})$$

Here $s(A_i) \in \{0, 1, 2\}$ defines the state of i^{th} input pair of bits (A_i), where $i \in \{1, 2, \dots, n/2\}$.

Using the probabilities of the individual states of the input pairs, assuming that they are independent, the probability of a state combination of input pairs can be computed as:

$$P(s(A_1) = s_1, s(A_2) = s_2, \dots, s(A_{n/2}) = s_{n/2}) = \prod_{i=1}^{n/2} P(s(A_i) = s_i) \quad (\text{B.4})$$

where $s_1, s_2, \dots, s_{n/2}$ is a state combination of $A_1, A_2, \dots, A_{n/2}$ input pairs. For example, in case of a 4-bit input the possible state combinations of A_1 and A_2 are $(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2), (2, 0), (2, 1)$, and $(2, 2)$. The probability of

a state combination, say $(0, 0)$, is given by $P(s(A_1) = 0) * P(s(A_2) = 0)$ based on Eq. (B.4).

To compute the probability of a state combination of input pairs more precisely, a modified version of the equation presented in [74] can be used. The equation for the given scenario can be reformulated as:

$$P(s(A_1) = s_1, s(A_2) = s_2, \dots, s(A_{n/2}) = s_{n/2}) = \sum_{\{a_{n-1}, \dots, a_0 \mid s(A_{n/2}) = s_{n/2} \wedge \dots \wedge s(A_1) = s_1\}} P((a_{n-1} \dots a_0)_2) \quad (\text{B.5})$$

For example, in case of a 4-bit input, the probability of the state combination $(0, 0)$ is given by,

$$\begin{aligned} P(s(A_1) = 0, s(A_2) = 0) &= \sum_{\{a_3, a_2, a_1, a_0 \mid s(A_2) = 0 \wedge s(A_1) = 0\}} P((a_3 a_2 a_1 a_0)_2) \\ &= P((a_3 a_2 a_1 a_0)_2 = (0000)_2) + P((a_3 a_2 a_1 a_0)_2 = (0001)_2) + \\ &\quad P((a_3 a_2 a_1 a_0)_2 = (0100)_2) + P((a_3 a_2 a_1 a_0)_2 = (0101)_2) \end{aligned} \quad (\text{B.6})$$

Note that the Eq. (B.5) considers the interdependency between pairs while Eq. (B.4) assumes that the pairs are independent, therefore, the results generated using Eq. (B.5) are more accurate than the results generated using Eq. (B.4). Although at first it seems that Eq. (B.5) will result in significant computational overheads, this is not the case for design space exploration. This is because of the fact that the probability of the state combinations of input pairs remain the same for all the configurations of the approximate squarer for a given input probability distribution, and is required to be computed only once. Therefore, the overhead when distributed over all the configurations results in insignificant overhead.

The PMFe for a given configuration can be computed by iterating over all the possible combinations of the input pairs. Algorithm 2 presents the pseudo-code for computing the PMFe of a given configuration of a squarer, provided an input distribution. The arrays V and P_v , returned by the algorithm, represent the PMFe where V stores the error values and P_v stores the corresponding error probabilities. Note that, although the computational complexity of the proposed algorithm is exponential, the design space exploration time has significantly been reduced by reducing the number of states per input pair from 4 (total number of input combinations, i.e., 2^2) to 3 (because of 3 states).

The error corresponding to the combination of input pairs can be computed by identifying the elementary module(s) that will generate error(s) and then adding the errors from all the modules together. Note that here by the error of an elementary module we mean the approximation error multiplied by the

Algorithm 2 Pseudo-code for computing PMFe

```

1: Input:
2:    $n$  : Number of input bits
3:   Config : Configuration of the approximate squarer
4:    $P(a_i)$  : Probabilities of individual bits of the input
5: Initialize:
6:    $V$  : Array for storing error magnitudes
7:    $P_v$  : Array for storing error probabilities
8:
9:   Compute  $P(s(A_i) = s_i) \forall A_i \in \{1, 2, \dots, n/2\}$  and  $s_i \in \{0, 1, 2\}$  using Eq. B.1, B.2,
   and B.3.
10:  Find  $C$ , i.e., All possible combinations of states of the pairs of input bits
11:  for  $c = \{1, \dots, \text{number of combinations in } C\}$  do
12:    Compute  $v_c$  (error value of the combination) using Algo. 3
13:    Compute  $\rho_c$  (probability of the combination) using Eq. (B.4) or Eq. (B.5)
14:    if  $v_c \in V$  then
15:       $P_v(v_c) = P_v(v_c) + \rho_c$ ;
16:    else
17:      Append  $v_c$  to  $V$  and place  $\rho_{v_c}$  in  $P_v(v_c)$ 
18:    end if
19:  end for
20:  RETURN  $V$  and  $P_v$ 

```

significance (shift factor) of the module. Algorithm 3 presents the pseudo-code for computing the error value (v_c) for a given combination of input pairs.

Note that Algorithm 2 provides the PMFe of a configuration which can be used to compute almost all the commonly used error metrics. Therefore, a complete design space, covering all the possible configurations of an approximate squarer, can be generated for a given input distribution to identify the pareto-optimal configurations which provides an optimal trade-off between a defined error metric and an efficiency target (area/power/performance).

Algorithm 3 Pseudo-code for computing the error value (v_c) for a given combination of input pairs

```

1: Input:
2:    $c$  : State combination that defines the state of each pair of input bits
3:   Config : Configuration of the approximate squarer
4: Initialize:
5:    $v_c = 0$ 
6:
7: for  $i = \{1, \dots, \text{length}(c)\}$  do
8:   if  $c(i) == 1$  & corresponding  $2 \times 2$  sq. is approx. then
9:      $v_c = v_c + \text{approximation error} \times \text{significance of the corresponding module}$ 
10:  else if  $c(i) == 2$  then
11:    for  $j = \{i + 1, i + 2, \dots, \text{length}(c)\}$  do
12:      if  $c(j) == 2$  then
13:         $v_c = v_c + \text{approximation error} \times \text{significance of the corresponding module}$ 
14:      end if
15:    end for
16:  end if
17: end for
18: RETURN  $v_c$ 

```

C

DESIGN SPACE EXPLORATION OF APPROXIMATE MULTIPLIERS FOR MAC

To quantify the gains offered by the ISH methodology as compared to the conventional methodology, we need to find the best quality-efficiency designs for each. Here we elaborate on the design space exploration tool¹ that leads us to such approximate recursive multiplier configurations for an approximate MAC unit. These designs are referred to as the *pareto-optimal* designs/configurations.

C.1 HUGE DESIGN SPACE - A CHALLENGE

The number of elementary multipliers increases rapidly with the increase in the number of bits per input (operand). The number of 2×2 elementary modules required for an $n \times n$ multiplier can be given as: $(n/2)^2$. The total number of possible configurations for an approximate multiplier directly depends on the number of elementary multipliers and the number of types that each can have. Assuming m as the number of types of elementary multipliers, the total number of possible configurations for an $n \times n$ multiplier can be given as:

$$\text{No. of configurations} = m^{(n/2)^2} \quad (\text{C.1})$$

As can be inferred from Eq. (C.1), the number of configurations grows rapidly both with m and n . To further highlight the requirement of a systematic design space exploration, Table C.1 presents the number of possible configurations for a few example cases with different m and n values. It can be seen that a huge design space has to be explored for a 16×16 multiplier case with only 5 options

This Appendix is based on [G:4].

¹This tool has been developed with the CARE-Tech (ECS group), TU Wien where the implementation was done by that group.

Table C.1: Number of configurations for a few example scenarios with different bit-widths (n) of multipliers and types of elementary 2×2 designs (m).

S. No.	n	m	No. of Configurations
1.	8	3	4.3×10^7
2.	8	5	1.53×10^{11}
3.	16	3	3.43×10^{30}
4.	16	5	5.42×10^{44}

for elementary 2×2 designs (S. No. 4). To tackle such an enormous design space, we have utilized a heuristic that prunes the search space in order to find the pareto-optimal configurations effectively.

C.2 DESIGN SPACE EXPLORATION

The design space exploration tool employs a recursive algorithm with intermediate pruning for fast exploration. The intermediate pruning is employed to prune less-effective parts of the overall design space at each intermediate stage to reduce the design space for the next subsequent stage. The overall flow is illustrated in Fig. C.1 and the related pseudo-code is given in [G:4]. The main steps of the tool-flow are as follows.

INITIALIZATION

In this step, we define a variable $E_Configs$ which stores the error and cost characteristics as well as the identities (IDs) of the elementary (2×2) multipliers. The error characteristics are stored in the form of an error map (E_Maps), which contains the output error for each possible input combination of a 2×2 multiplier. An example illustration of an E_Map is shown in Fig. C.2. The cost characteristics include the area and/or power values of the elementary multipliers.

STEP 1

Given the probability distributions of the input operands, i.e., ρ_x and ρ_y , the first step involves the input probability computation of all the individual elementary (i.e., 2×2 in our case) multipliers in an $n \times n$ multiplier. The input probability distribution of all the elementary modules is stored in a matrix ρ , where each entity of the matrix represents the input probability distribution of a single elementary multiplier and has a cumulative sum of 1.

To compute the input probability of all the elementary multipliers, we first independently compute the probability distribution of the pairs of bits of the input operands x and y which are the inputs to these elementary multipliers.

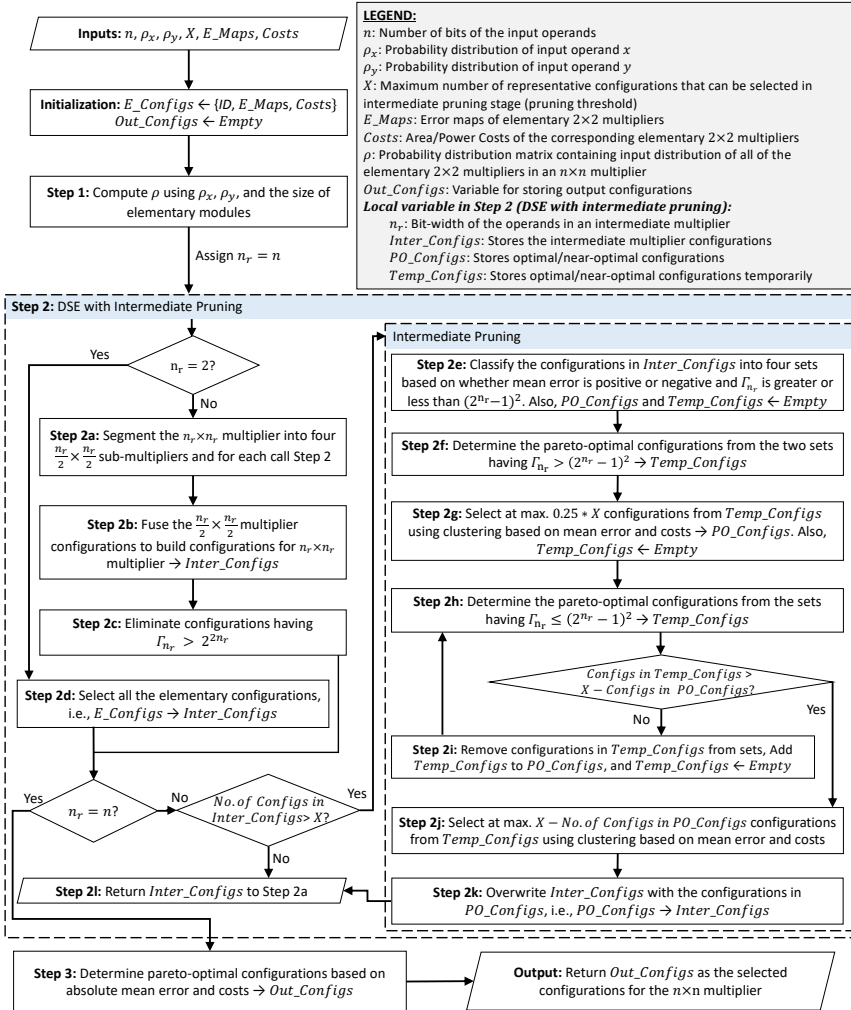


Figure C.1: Design space exploration tool for approximate recursive multipliers. The tool takes into account the bit-widths and the probability distributions of inputs, pruning threshold (X), and error & cost characteristics of the elementary multipliers to return pareto-optimal/near-optimal configurations while using limited computational and memory resources.

A \ B	0 = (00) ₂	1 = (01) ₂	2 = (10) ₂	3 = (11) ₂
0 = (00) ₂	0	0	0	0
1 = (01) ₂	0	0	0	0
2 = (10) ₂	0	0	0	0
3 = (11) ₂	0	0	0	-4

Figure C.2: An example of E_Map (error map) for the M_4 elementary multiplier shown in Fig. 5.5.

Similar to [75], the probability distribution of a pair of consecutive bits of the input operand x can be given as:

$$P_x\{i\}(k) = \sum_{q=0}^{2^{n-2i-2}-1} \sum_{p=0}^{2^{2i}-1} \rho_x(q \times 2^{2i+2} + k \times 2^{2i} + p) \quad (C.2)$$

where i defines the pair of bits in the input operand, i.e., the i^{th} pair consists of the bits at locations $2i$ and $2i + 1$, and k defines the combined decimal value of the bits ($i \in \{0, 1, 2, \dots, n/2 - 1\}$ and $k \in \{0, 1, 2, 3\}$). Similarly, the probability distribution of a pair of consecutive bits (defined by j) of the input operand y can be given as:

$$P_y\{j\}(l) = \sum_{r=0}^{2^{n-2j-2}-1} \sum_{s=0}^{2^{2j}-1} \rho_y(r \times 2^{2j+2} + l \times 2^{2j} + s) \quad (C.3)$$

where $j \in \{0, 1, 2, \dots, n/2 - 1\}$ and $l \in \{0, 1, 2, 3\}$. Using Eq. C.2 and Eq. C.3, and assuming x and y as independent random variables, the input probability distribution of a specific 2×2 multiplier (represented by $\rho\{i, j\}$) can be computed using the following equation:

$$\rho\{i, j\}(k, l) = P_x\{i\}(k) \times P_y\{j\}(l) \quad (C.4)$$

As an example, consider a 4×4 multiplier that consists of four 2×2 multipliers. The probability distributions of the four 2×2 multipliers are given as $\rho\{0, 0\}$, $\rho\{0, 1\}$, $\rho\{1, 0\}$, and $\rho\{1, 1\}$, where the probability distribution of each 2×2 multiplier, say $\rho\{0, 1\}$, has a probability-value for each input combination, i.e., $\rho\{0, 1\}(0, 0)$, $\rho\{0, 1\}(0, 1)$, $\rho\{0, 1\}(0, 2)$, $\rho\{0, 1\}(0, 3)$, $\rho\{0, 1\}(1, 0)$, ..., $\rho\{0, 1\}(3, 3)$.

STEP 2

The ρ computation step is followed by a recursive step where at each call the $n_r \times n_r$ multiplier is divided into four $n_r/2 \times n_r/2$ sub-multiplier units (see Fig. 5.4a for an example of a 4×4 multiplier) and for each sub-multiplier Step 2 is called again with the corresponding multiplier size and input distribution (Step

2a). Note that for the very first call to Step 2 (i.e., while moving from Step 1 to 2) the variable n_r is initialized with n where n_r represents a local variable that defines the bit-width of the inputs of the multiplier at a particular recursive stage.

From each intermediate stage, the step returns at maximum X number of highly-efficient configurations given a defined multiplier size and input probability distribution. Here X represents a parameter which defines the maximum number of representative configurations that can be selected from an intermediate recursive stage. It should be noted that a large number of representatives (a high value of X) results in a big design space when combined for larger multipliers and thus the design space exploration consumes more time and computational resources. On the other hand, a small value of X results in fewer pareto-optimal points at intermediate stages and, thereby, can result in sub-optimal results. Therefore, X should be chosen as high as possible/feasible for the simulation platform.

The received configurations for all the four sub-multipliers are then combined to generate the possible configurations for the $n_r \times n_r$ multiplier which are stored in *Inter_Configs* (Step 2b). At the same step, the mean error values, the maximum possible output values, and the costs of the generated configurations are also computed using the error and cost characteristics of the corresponding sub-multipliers. The mean error of a configuration of an $n_r \times n_r$ multiplier composed of four $n_r/2 \times n_r/2$ multipliers can be computed as,

$$ME_{n_r} = ME_a + 2^{n_r/2}ME_b + 2^{n_r/2}ME_c + 2^{n_r}ME_d \quad (C.5)$$

where ME_{n_r} represents the mean error of the $n_r \times n_r$ multiplier and ME_a , ME_b , ME_c and ME_d represent the mean error of the M_a , M_b , M_c and M_d sub-multipliers, respectively (see Fig. 5.4b for an example of an 8×8 multiplier). Similarly, the maximum possible output value of a configuration of an $n_r \times n_r$ multiplier can be computed using Eq. (5.9).

To compute the area and power costs of the generated configurations, we have utilized the model discussed in Section 5.3.2. The model estimates the costs of an overall multiplier by adding the costs of the corresponding sub-multipliers together with their contribution to the adder trees. Once the configurations have been generated and all the required characteristics have been computed, the configurations are then checked for the maximum possible output value to avoid overflow conditions (Step 2c), as mentioned in Section 5.2.2.

All the configurations having a maximum possible output value greater than or equal to 2^{2n_r} (for an $n_r \times n_r$ multiplier) are removed from the *Inter_Configs*. At this point, the value of n_r is compared with n and if it is equal, all the configurations are forwarded to Step 3. However, if n_r is not equal to n , the remaining number of configurations is checked and intermediate pruning (Step 2e-2k) is applied if it is greater than a pre-specified threshold, i.e., X . This is mainly done to reduce the number of possible configurations such that the design

space exploration can be performed using limited computational and memory resources and in a time-efficient manner.

The recursive function keeps on calling itself unless the size of the sub-multipliers is equivalent to 2×2 (i.e., $n_r = 2$), which acts as the termination point for the recursive calling. At this point, Step 2d is performed where *Inter_Configs* is initialized with *E_Configs* (i.e., all the elementary multiplier configurations) along with their mean errors and maximum possible output values. The mean error for each elementary multiplier is computed by taking the dot product of the error map (*E_Map*) of the corresponding elementary multiplier with the input probability distribution matrix. Note that Step 2d is called for each elementary module location in an $n \times n$ multiplier and the probability matrix used for computing the mean errors of the elementary multipliers is the one which contains the input probability distribution of that particular location.

INTERMEDIATE PRUNING (STEP 2E - 2K)

Whenever the number of intermediate configurations in *Inter_Configs* (after Step 2c or Step 2d) is greater than X and $n_r \neq n$, the intermediate pruning is called for choosing a subset of X effective configurations which can be used as the representatives of the complete design space of the sub-multiplier. To achieve this, at Step 2e, we classify the configurations into four sets based on the mean error and maximum possible output value of the configurations. Set 1 contains configurations having mean error > 0 and maximum output value $> (2^{n_r} - 1)^2$, Set 2 contains configurations having mean error ≤ 0 and maximum output value $> (2^{n_r} - 1)^2$, Set 3 contains configurations having mean error > 0 and maximum output value $\leq (2^{n_r} - 1)^2$, and Set 4 contains configurations having mean error ≤ 0 and maximum output value $\leq (2^{n_r} - 1)^2$. The configurations are divided into these four sets because of two main reasons: 1) So that different number of configurations can be selected from different sets based on their importance, for example, the configurations having maximum output value greater than $(2^{n_r} - 1)^2$ may result in overflow as mentioned in Section 5.2.2 and, therefore, should be given less importance; and 2) The configurations having positive and negative mean error should be given equal importance, as only in case configurations with both positive and negative mean error are available, the internal self-healing can be utilized to generate approximate configurations for larger multipliers that result in zero/near-to-zero mean error.

To select a subset of effective configurations, we first find pareto-optimal configurations from sets 1 and 2 based on absolute mean error and cost and store them temporarily in *Temp_Configs* (Step 2f). Then, in Step 2g, we check the number of pareto-optimal configurations. If it is greater than 25% of X , we first select the two extreme values from the pareto-optimal configurations, i.e., configurations having minimum and maximum absolute mean error, and then apply k-means clustering to find $0.25 * X - 2$ clusters using the rest of the pareto-optimal configurations based on mean error and cost. Here, k-means [76] is applied to group

configurations offering nearby error-cost points in the quality-efficiency trade-off. The configuration closest to the cluster centroid is then selected from each cluster as its representative. The selected configurations are then stored in a local variable *PO_Configs*. Moreover, if the number of pareto-optimal configurations in Step 2f is less than $0.25 * X$, all the configurations are selected and stored in *PO_Configs*. Also, in the same step the *Temp_Configs* is re-initialized to empty.

The remaining configurations are selected from sets 3 and 4 using Step 2h, 2i and 2j. In Step 2h, we find the pareto-optimal configurations from the sets based on the absolute mean error and the cost of the configurations and store them in *Temp_Configs*. If the selected number of configurations from these sets is greater than the remaining number of configurations (i.e., greater than $X - \text{No. of configurations in } PO_Config\text{s}$), we perform Step 2j to find the remaining required configurations from *Temp_Configs* using clustering (similar to Step 2g). However, if it is less, we perform Step 2i where we remove the configurations from the sets that are present in *Temp_Configs*, and we add the configurations of *Temp_Configs* to *PO_Configs*. Finally, we re-initialize *Temp_Configs* to empty before moving back to Step 2h.

Then, Step 2h is performed again using the modified sets to find near-optimal points. This cycle (Step 2h \rightarrow Step 2i \rightarrow Step 2h) is repeated until the condition is satisfied (or the sets are empty). This procedure ensures that we select the most effective (optimal/near-optimal) configurations from the sets as much as allowed by the X parameter. Afterwards, Step 2j is performed and the selected configurations are added to *PO_Configs*. The resultant configurations are forwarded to Step 2k where *Inter_Configs* is overwritten with the configurations in *PO_Configs*. Then the intermediate pruning function is returned to Step 2l, where the *Inter_Configs* are forwarded to the higher stage (Step 2a) for generating configurations of larger multipliers.

STEP 3

From the received configurations the pareto-optimal configurations are found using their absolute mean error and the area/power cost characteristics. The resultant configurations are then returned as the final configurations for the $n \times n$ multiplier.

C.3 VIABILITY OF OUR APPROACH

We utilized the above design space exploration tool for finding the pareto-optimal designs for 4-bit, 8-bit and 16-bit multipliers. Table C.2 shows the runtime of the simulations (with $X = 60$) using MATLAB (2017a) on an Intel Core i5-6600 CPU with 16 GB of RAM. We have also simulated the first case ($n = 8$ and $m = 3$) exhaustively which resulted in a simulation runtime of 43 seconds on our system. Interestingly, the pareto-optimal configurations for the aforesaid exhaustive simulation are exactly the same as of our algorithm at $X = 60$, which

Table C.2: Simulation runtime for the design space exploration of multipliers. While using a general purpose simulation platform, the design space exploration tool explores a huge design space (S. No. 4) in less than four minutes.

S. No.	n	m	No. of Configurations	Simulation Time (Seconds)
1.	8	3	4.3×10^7	5
2.	8	5	1.53×10^{11}	7
3.	16	3	3.43×10^{30}	138
4.	16	5	5.42×10^{44}	209

has a simulation runtime of 5 seconds. Moreover, the design space exploration tool enables us to explore a huge design space ($n = 16$ and $m = 5$) in less than four minutes using a general purpose computer system as a simulation platform.

ACRONYMS

A	AAMMP	absolute approximate multiplier mirror pair
	AASMP	absolute approximate squarer mirror pair
	Adaptive-SAM	adaptive statistical approximation model
	AER	adaptive error resilience
	ARC	application resilience characterization
	ASAC	automatic sensitivity analysis for approximate computing
	ASIC	application specific integrated circuit
C	CMOS	complementary metal-oxide semiconductor
	CPU	central processing unit
D	DoA	degree of approximation
	DRAM	dynamic RAM
	DVFS	dynamic voltage and frequency scaling
E	EM	error mean
	EP	error predictability
	ER	error rate
	ETA	error tolerant adder
F	FFT	fast Fourier transform
	FLOPS	floating-point operations per second
	FPGA	field programmable gate array
I	iACT	intel's approximate computing toolkit
	IC	integrated circuit
	ISH	internal-self-healing
L	LOFAR	The low frequency array
	LS	least squares
	LUT	look-up table
	LVA	load value approximation
M	MAC	multiply-accumulate
	ME	mean error
	MSE	mean square error
N	NTV	near-threshold voltage
P	PAC	program analysis for approximation-aware compilation
	PMFe	probability mass function of error

Q	QoS	quality of service
R	RAM	random-access memory
	RTL	register-transfer level
S	SAC	square-accumulate
	SAM	statistical approximation model
	SDP	science data processing
	SH	self-healing
	SKA	square kilometer array
	SNR	signal-to-noise ratio
	SRAM	static RAM
	StEFCal	statistically efficient and fast calibration
T	TSAM	technique specific approximation model
V	VLSI	very large scale integration
X	xMAC	approximate multiply-accumulate

BIBLIOGRAPHY

- [1] SKA Science. [Online] <https://www.skatelescope.org/science/>. Last accessed on June 18, 2020. (Cited on page 4).
- [2] IEEE standard glossary of computer hardware terminology. *IEEE Std 610.10-1994*, 1995. doi: 10.1109/IEEESTD.1995.79522. (Cited on page 13).
- [3] R. Airoidi, F. Campi, and J. Nurmi. Approximate computing for complexity reduction in timing synchronization. *EURASIP Journal on Advances in Signal Processing*, 2014(1):155, 2014. (Cited on page 9).
- [4] A. Alaghi and J. P. Hayes. Survey of stochastic computing. *ACM Transactions on Embedded computing systems (TECS)*, 12(2s):92:1–92:19, 2013. (Cited on pages 9, 10, and 29).
- [5] A. Alaghi, W. Qian, and J. P. Hayes. The promise and challenge of stochastic computing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2017. (Cited on page 10).
- [6] M. K. Ayub, O. Hasan, and M. Shafique. Statistical error analysis for low power approximate adders. In *Proceedings of the 54th Annual Design Automation Conference 2017*, page 75. ACM, 2017. (Cited on pages 58 and 87).
- [7] N. Banerjee, G. Karakonstantis, and K. Roy. Process variation tolerant low power dct architecture. In *2007 Design, Automation & Test in Europe Conference & Exhibition*, pages 1–6. IEEE, 2007. (Cited on page 20).
- [8] D. Bankman and B. Murmann. An 8-bit, 16 input, 3.2 pj/op switched-capacitor dot product circuit in 28-nm fdsoi cmos. In *2016 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, pages 21–24. IEEE, 2016. (Cited on page 97).
- [9] B. Barrois, O. Sentieys, and D. Menard. The hidden cost of functional approximation against careful data sizing: a case study. In *Proceedings of the Conference on Design, Automation & Test in Europe*, pages 181–186. European Design and Automation Association, 2017. (Cited on page 64).
- [10] C. H. Bennett and R. Landauer. The fundamental physical limits of computation. *Scientific American*, 253(1):48–57, 1985. (Cited on page 1).
- [11] K. Bhardwaj, P. S. Mane, and J. Henkel. Power-and area-efficient approximate wallace tree multiplier for error-resilient systems. In *Fifteenth International Symposium on Quality Electronic Design*, pages 263–269. IEEE, 2014. (Cited on page 58).
- [12] S. S. Bhattacharyya, E. Deprettere, R. Leupers, and J. Takala. *Handbook of signal processing systems*. Springer, 2018. (Cited on page 46).

- [13] M. Bohr. A 30 year retrospective on dennard's mosfet scaling paper. *IEEE Solid-State Circuits Society Newsletter*, 12(1):11–13, 2007. (Cited on page 1).
- [14] A.-J. Boonstra. *Radio frequency interference mitigation in radio astronomy*. PhD thesis, Delft University of Technology, 2005. (Cited on page 5).
- [15] A.-J. Boonstra and A.-J. Van der Veen. Gain calibration methods for radio telescope arrays. *IEEE Transactions on Signal Processing*, 51(1):25–38, 2003. (Cited on page 37).
- [16] D. Bortolotti, H. Mamaghanian, A. Bartolini, M. Ashouei, J. Stuijt, D. Atienza, P. Vanderghenst, and L. Benini. Approximate compressed sensing: ultra-low power biosignal processing via aggressive voltage scaling on a hybrid memory multi-core processor. In *2014 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 45–50. IEEE, 2014. (Cited on page 20).
- [17] R. Boyapati, J. Huang, P. Majumder, K. H. Yum, and E. J. Kim. Approx-noc: A data approximation framework for network-on-chip architectures. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pages 666–677, 2017. (Cited on page 19).
- [18] L. Breslau and S. Shenker. Best-effort versus reservations: A simple comparative analysis. In *Proceedings of the ACM SIGCOMM'98 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 3–16, 1998. (Cited on page 9).
- [19] P. Broekema et al. Exascale high performance computing in the square kilometer array. In *Proceedings of the Astro-HPC'12*, pages 9–16. ACM, 2012. (Cited on page 4).
- [20] M. D. Buhari, G. Y. Tian, R. Tiwari, and A. H. Muqaibel. Multicarrier sar image reconstruction using integrated music-lse algorithm. *IEEE Access*, 6:22827–22838, 2018. (Cited on pages 32 and 56).
- [21] B. Burke and F. Graham-Smith. *An Introduction to Radio Astronomy*. Cambridge University Press, Third Edition, 2010. (Cited on page 5).
- [22] L. N. Chakrapani, P. Korkmaz, B. E. Akgul, and K. V. Palem. Probabilistic system-on-a-chip architectures. *ACM Transactions on Design Automation of Electronic Systems*, 12(3):29, 2007. (Cited on page 11).
- [23] I. J. Chang, D. Mohapatra, and K. Roy. A priority-based 6t/8t hybrid sram architecture for aggressive voltage scaling in video applications. *IEEE transactions on circuits and systems for video technology*, 21(2):101–112, 2011. (Cited on page 20).
- [24] S. Cheemalavagu, P. Korkmaz, K. V. Palem, B. E. Akgul, and L. N. Chakrapani. A probabilistic cmos switch and its realization by exploiting noise. In *Proceedings of International Conference on VLSI*, pages 535–541. IFIP, 2005. (Cited on page 11).
- [25] V. K. Chippa, D. Mohapatra, A. Raghunathan, K. Roy, and S. T. Chakradhar. Scalable effort hardware design: Exploiting algorithmic resilience for energy efficiency. In *Proceedings of the 47th Design Automation Conference*, pages 555–560. ACM, 2010. (Cited on page 18).

- [26] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan. Analysis and characterization of inherent application resilience for approximate computing. In *Proceedings of the 50th Annual Design Automation Conference*, page 113. ACM, 2013. (Cited on pages 2, 3, 4, 14, 17, 18, 32, 34, 56, and 85).
- [27] S. De, J. Huisken, and H. Corporaal. An automated approximation methodology for arithmetic circuits. In *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 1–6. IEEE, 2019. (Cited on pages 25 and 117).
- [28] R. H. Dennard, F. H. Gaensslen, V. L. Rideout, E. Bassous, and A. R. LeBlanc. Design of ion-implanted mosfet’s with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5):256–268, 1974. (Cited on page 1).
- [29] J. A. Dickson, R. D. McLeod, and H. C. Card. Stochastic arithmetic implementations of neural networks with in situ learning. In *Proceedings of the International Conference on Neural Networks*, pages 711–716. IEEE, 1993. (Cited on page 10).
- [30] R. G. Dreslinski, M. Wiecekowsk, D. Blaauw, D. Sylvester, and T. Mudge. Near-threshold computing: Reclaiming moore’s law through energy efficient integrated circuits. *Proceedings of the IEEE*, 98(2):253–266, 2010. (Cited on page 20).
- [31] S. Dutt, A. Chauhan, R. Bhadoriya, S. Nandi, and G. Trivedi. A high-performance energy-efficient hybrid redundant mac for error-resilient applications. In *2015 28th International Conference on VLSI Design*, pages 351–356. IEEE, 2015. (Cited on pages 59, 88, and 89).
- [32] J. Echavarria, S. Wildermann, A. Becher, J. Teich, and D. Ziener. Fau: Fast and error-optimized approximate adder units on lut-based fpgas. In *2016 International Conference on Field-Programmable Technology (FPT)*, pages 213–216. IEEE, 2016. (Cited on pages 21, 25, and 118).
- [33] H. Esmailzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger. Dark silicon and the end of multicore scaling. In *2011 38th Annual international symposium on computer architecture (ISCA)*, pages 365–376. IEEE, 2011. (Cited on page 2).
- [34] H. Esmailzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger. Power challenges may end the multicore era. *Communications of the ACM*, 56(2):93–102, 2013. (Cited on page 1).
- [35] D. Esposito, A. G. Strollo, and M. Alioto. Low-power approximate mac unit. In *2017 13th Conference on Ph. D. Research in Microelectronics and Electronics (PRIME)*, pages 81–84. IEEE, 2017. (Cited on pages 59 and 88).
- [36] B. R. Gaines. Stochastic computing. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, pages 149–156. ACM, 1967. (Cited on page 10).
- [37] J. George, B. Marr, B. E. Akgul, and K. V. Palem. Probabilistic arithmetic and energy efficient embedded signal processing. In *Proceedings of the International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, CASES ’06, pages 158–168. ACM, 2006. (Cited on page 11).

- [38] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy. Impact: imprecise adders for low-power approximate computing. In *Proceedings of the 17th IEEE/ACM international symposium on Low-power electronics and design*, pages 409–414. IEEE Press, 2011. (Cited on page 58).
- [39] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy. Low-power digital signal processing using approximate adders. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(1):124–137, 2012. (Cited on pages 3, 21, 23, 58, and 87).
- [40] Q. Hang. Counting the floating point operations (FLOPS). [Online] <https://www.mathworks.com/matlabcentral/fileexchange/50608-counting-the-floating-point-operations-flops>. Last accessed on June 18, 2020. (Cited on page 38).
- [41] M. A. Hanif, R. Hafiz, and M. Shafique. Error resilience analysis for systematically employing approximate computing in convolutional neural networks. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 913–916. IEEE, 2018. (Cited on page 3).
- [42] S. Hashemi, R. Bahar, and S. Reda. Drum: A dynamic range unbiased multiplier for approximate applications. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 418–425. IEEE Press, 2015. (Cited on pages 22, 23, 32, 87, and 88).
- [43] K. He, A. Gerstlauer, and M. Orshansky. Circuit-level timing-error acceptance for design of energy-efficient dct/idct-based systems. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(6):961–974, 2013. (Cited on page 20).
- [44] J. L. Hennessy and D. A. Patterson. *Computer architecture: a quantitative approach*. Elsevier, 2011. (Cited on page 12).
- [45] H. Hoffmann, S. Misailovic, S. Sidiroglou, A. Agarwal, and M. Rinard. Using code perforation to improve performance, reduce energy consumption, and respond to failures. 2009. (Cited on page 18).
- [46] J. Hou, Y. Zhu, S. Du, S. Song, and Y. Song. Fpga-based scale-out prototyping of degridting algorithm for accelerating square kilometre array telescope data processing. *IEEE Access*, 8:15586–15597, 2020. (Cited on page 5).
- [47] D. Jeon, M. Seok, C. Chakrabarti, D. Blaauw, and D. Sylvester. A super-pipelined energy efficient subthreshold 240 ms/s fft core in 65 nm cmos. *IEEE Journal of Solid-State Circuits*, 47(1):23–34, 2011. (Cited on page 20).
- [48] H. Jiang, J. Han, F. Qiao, and F. Lombardi. Approximate radix-8 booth multipliers for low-power and high-performance operation. *IEEE Transactions on Computers*, 65(8):2638–2644, 2015. (Cited on pages 58 and 87).
- [49] H. Jiang, C. Liu, N. Maheshwari, F. Lombardi, and J. Han. A comparative evaluation of approximate multipliers. In *2016 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, pages 191–196. IEEE, 2016. (Cited on page 58).

- [50] H. Jiang, C. Liu, L. Liu, F. Lombardi, and J. Han. A review, classification, and comparative evaluation of approximate arithmetic circuits. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 13(4):60, 2017. (Cited on pages 3, 32, and 117).
- [51] R. Jongerius, S. Wijnholds, R. Nijboer, and H. Corporaal. An end-to-end computing model for the square kilometre array. *Computer*, 47(9):48–54, 2014. ISSN 0018-9162. (Cited on pages 5 and 32).
- [52] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pages 1–12. IEEE, 2017. (Cited on page 97).
- [53] A. B. Kahng and S. Kang. Accuracy-configurable adder for approximate arithmetic designs. In *Proceedings of the 49th Annual Design Automation Conference*, pages 820–825. ACM, 2012. (Cited on pages 21, 58, and 87).
- [54] S. Kamdar and N. Kamdar. big. little architecture: Heterogeneous multicore processing. *International Journal of Computer Applications*, 119(1), 2015. (Cited on page 32).
- [55] G. Karakonstantis, D. Mohapatra, and K. Roy. System level dsp synthesis using voltage overscaling, unequal error protection & adaptive quality tuning. In *2009 IEEE Workshop on Signal Processing Systems*, pages 133–138. IEEE, 2009. (Cited on page 20).
- [56] U. R. Karpuzcu, I. Akturk, and N. S. Kim. Accordion: Toward soft near-threshold voltage computing. In *High Performance Computer Architecture (HPCA), 2014 IEEE 20th International Symposium on*, pages 72–83. IEEE, 2014. (Cited on page 32).
- [57] J. Kim and S. Tiwari. Inexact computing using probabilistic circuits: Ultra low-power digital processing. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 10(2):16, 2014. (Cited on page 11).
- [58] Y.-C. Kim and M. A. Shanblatt. Architecture and statistical model of a pulse-mode digital multilayer neural network. *IEEE Transactions on Neural Networks*, 6(5):1109–1118, Sep 1995. ISSN 1045-9227. (Cited on page 10).
- [59] A. Kourfali and D. Stroobandt. Superimposed in-circuit debugging for self-healing fpga overlays. In *2018 IEEE 19th Latin-American Test Symposium (LATS)*, pages 1–6. IEEE, 2018. (Cited on pages 6 and 56).
- [60] A. Krapukhin. Approximate least squares accelerator. *Essay (M.Sc.), University of Twente*, 2019. (Cited on page 46).
- [61] P. K. Krause and I. Polian. Adaptive voltage over-scaling for resilient applications. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*, pages 1–6. IEEE, 2011. (Cited on pages 20, 31, and 33).

- [62] S. Kuang and J. Wang. Low-error configurable truncated multipliers for multiply-accumulate applications. *Electronics letters*, 42(16):1, 2006. (Cited on pages 59, 64, and 88).
- [63] P. Kulkarni, P. Gupta, and M. Ercegovac. Trading accuracy for power with an underdesigned multiplier architecture. In *2011 24th International Conference on VLSI Design*, pages 346–351. IEEE, 2011. (Cited on pages 3, 16, 24, 26, 28, 58, 59, 69, 74, 77, 78, 87, 88, 89, and 91).
- [64] P. Kulkarni, P. Gupta, and M. D. Ercegovac. Trading accuracy for power in a multiplier architecture. *Journal of Low Power Electronics*, 7(4):490–501, 2011. (Cited on page 26).
- [65] K. Y. Kyaw, W. L. Goh, and K. S. Yeo. Low-power high-speed multiplier for error-tolerant application. In *2010 IEEE International Conference of Electron Devices and Solid-State Circuits (EDSSC)*, pages 1–4. IEEE, 2010. (Cited on page 58).
- [66] M. S. e. a. Lau. Modeling of probabilistic ripple-carry adders. In *Proceedings of the Electronic Design, Test and Application, DELTA'10*. IEEE, 2010. (Cited on page 11).
- [67] S. Lee, L. K. John, and A. Gerstlauer. High-level synthesis of approximate hardware under joint precision and voltage scaling. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pages 187–192. IEEE, 2017. (Cited on page 26).
- [68] V. T. Lee, A. Alaghi, R. Pamula, V. S. Sathe, L. Ceze, and M. Oskin. Architecture considerations for stochastic computing accelerators. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2277–2289, 2018. (Cited on page 97).
- [69] V. Leon, G. Zervakis, D. Soudris, and K. Pekmestzi. Approximate hybrid high radix encoding for energy-efficient inexact multipliers. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26(3):421–430, 2017. (Cited on pages 58, 87, and 117).
- [70] C. Li, W. Luo, S. S. Sapatnekar, and J. Hu. Joint precision optimization and high level synthesis for approximate computing. In *Proceedings of the 52nd Annual Design Automation Conference*, pages 1–6, 2015. (Cited on page 26).
- [71] A. Lingamneni, C. Enz, J.-L. Nagel, K. Palem, and C. Piguet. Energy parsimonious circuit design through probabilistic pruning. In *2011 Design, Automation & Test in Europe*, pages 1–6. IEEE, 2011. (Cited on pages 11, 20, and 29).
- [72] S. Liu, K. Pattabiraman, T. Moscibroda, and B. G. Zorn. Flicker: saving dram refresh-power through critical data partitioning. In *Proceedings of the sixteenth international conference on Architectural support for programming languages and operating systems*, pages 213–224, 2011. (Cited on page 19).
- [73] I. L. Markov. Limits on fundamental limits to computation. *Nature*, 512(7513):147, 2014. (Cited on pages 1 and 2).

- [74] S. Mazahir, O. Hasan, R. Hafiz, M. Shafique, and J. Henkel. Probabilistic error modeling for approximate adders. *IEEE Transactions on Computers*, 66(3):515–530, 2016. (Cited on pages 58, 87, 125, and 127).
- [75] S. Mazahir, O. Hasan, R. Hafiz, and M. Shafique. Probabilistic error analysis of approximate recursive multipliers. *IEEE Transactions on Computers*, 66(11):1982–1990, 2017. (Cited on pages 26, 58, 87, 88, 91, 125, and 134).
- [76] J. Meng, S. Chakradhar, and A. Raghunathan. Best-effort parallel execution framework for recognition and mining applications. In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–12. IEEE, 2009. (Cited on pages 4, 9, 32, and 136).
- [77] J. Miao, K. He, A. Gerstlauer, and M. Orshansky. Modeling and synthesis of quality-energy optimal approximate adders. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, ICCAD '12*, pages 728–735. IEEE, 2012. (Cited on pages 20, 21, 29, and 32).
- [78] S. Misailovic, S. Sidiroglou, H. Hoffmann, and M. Rinard. Quality of service profiling. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering—Volume 1*, pages 25–34, 2010. (Cited on pages 3, 14, and 34).
- [79] S. Misailovic, S. Sidiroglou, and M. C. Rinard. Dancing with uncertainty. In *Proceedings of the 2012 ACM workshop on Relaxing synchronization for multicore and manycore scalability*, pages 51–60. ACM, 2012. (Cited on page 18).
- [80] A. K. Mishra, R. Barik, and S. Paul. iact: A software-hardware framework for understanding the scope of approximate computing. In *Workshop on Approximate Computing Across the System Stack (WACAS)*, 2014. (Cited on pages 3, 15, and 34).
- [81] S. Mittal. A survey of techniques for approximate computing. *ACM Computing Surveys (CSUR)*, 48(4):62, 2016. (Cited on pages 3 and 9).
- [82] D. Mohapatra, G. Karakonstantis, and K. Roy. Significance driven computation: a voltage-scalable, variation-aware, quality-tuning motion estimator. In *Proceedings of the 2009 ACM/IEEE international symposium on Low power electronics and design*, pages 195–200, 2009. (Cited on page 20).
- [83] A. Momeni, J. Han, P. Montuschi, and F. Lombardi. Design and analysis of approximate compressors for multiplication. *IEEE Transactions on Computers*, 64(4):984–994, 2014. (Cited on pages 58 and 87).
- [84] G. E. Moore. Cramming more components onto integrated circuits. *Electronics*, pages 114–117, 1965. (Cited on page 1).
- [85] V. Mrazek, Z. Vasicek, L. Sekanina, H. Jiang, and J. Han. Scalable construction of approximate multipliers with formally guaranteed worst case error. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, (99):1–5, 2018. (Cited on pages 26, 32, 87, 96, and 98).

- [86] S. Naghibzadeh, A. Sardarabadi, and A. van der Veen. Radioastronomical image reconstruction with regularized least squares. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3316–3320. IEEE, March 2016. (Cited on pages 32 and 56).
- [87] K. Nepal, S. Hashemi, H. Tann, R. I. Bahar, and S. Reda. Automated high-level generation of low-power approximate computing circuits. *IEEE Transactions on Emerging Topics in Computing*, 7(1):18–30, 2016. (Cited on page 25).
- [88] E. Nogues, D. Menard, and M. Pelcat. Algorithmic-level approximate computing applied to energy efficient hevc decoding. *IEEE Transactions on Emerging Topics in Computing*, 2016. (Cited on pages 56 and 85).
- [89] V. T. Olafsson, D. C. Noll, and J. A. Fessler. Fast spatial resolution analysis of quadratic penalized least-squares image reconstruction with separate real and imaginary roughness penalty: Application to fmri. *IEEE transactions on medical imaging*, 37(2):604–614, 2017. (Cited on pages 32 and 56).
- [90] K. V. Palem. Energy aware algorithm design via probabilistic computing: from algorithms and models to moore’s law and novel (semiconductor) devices. In *Proceedings of the 2003 international conference on Compilers, architecture and synthesis for embedded systems, CASES ’03*, pages 113–116. ACM, 2003. (Cited on page 10).
- [91] K. V. Palem. Energy aware computing through probabilistic switching: A study of limits. *IEEE Transactions on Computers*, 54(9):1123–1137, 2005. (Cited on pages 10 and 11).
- [92] D. Palomino, M. Shafique, A. Susin, and J. Henkel. Thermal optimization using adaptive approximate computing for video coding. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1207–1212. IEEE, 2016. (Cited on page 18).
- [93] A. Parsons, D. Backer, C. Chang, D. Chapman, H. Chen, P. Crescini, C. De Jesus, C. Dick, P. Droz, D. MacMahon, et al. Petaop/second fpga signal processing for seti and radio astronomy. In *2006 Fortieth Asilomar Conference on Signals, Systems and Computers*, pages 2031–2035. IEEE, 2006. (Cited on page 5).
- [94] N. Petra, D. De Caro, V. Garofalo, E. Napoli, and A. G. Strollo. Truncated binary multipliers with variable correction and minimum mean square error. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 57(6):1312–1325, 2009. (Cited on pages 32, 59, 64, and 88).
- [95] N. Pinckney, K. Sewell, R. G. Dreslinski, D. Fick, T. Mudge, D. Sylvester, and D. Blaauw. Assessing the performance limits of parallelized near-threshold computing. In *Proceedings of the 49th Annual Design Automation Conference*, pages 1147–1152, 2012. (Cited on page 20).
- [96] N. Pinckney, D. Blaauw, and D. Sylvester. Low-power near-threshold design: Techniques to improve energy efficiency. *IEEE Solid-State Circuits Magazine*, 7(2):49–57, 2015. (Cited on page 20).

- [97] B. S. Prabakaran, S. Rehman, M. A. Hanif, S. Ullah, G. Mazaheri, A. Kumar, and M. Shafique. Demas: An efficient design methodology for building approximate adders for fpga-based systems. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 917–920. IEEE, 2018. (Cited on pages 25, 32, 58, 87, and 118).
- [98] J. M. Rabaey, A. P. Chandrakasan, and B. Nikolić. *Digital integrated circuits: a design perspective*, volume 7. Pearson Education Upper Saddle River, NJ, 2003. (Cited on page 20).
- [99] A. Ranjan, A. Raha, S. Venkataramani, K. Roy, and A. Raghunathan. Aslan: Synthesis of approximate sequential circuits. In *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. IEEE, 2014. (Cited on page 25).
- [100] A. Ranjan, S. Venkataramani, X. Fong, K. Roy, and A. Raghunathan. Approximate storage for energy efficient spintronic memories. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2015. (Cited on page 19).
- [101] S. Rehman, W. El-Harouni, M. Shafique, A. Kumar, and J. Henkel. Architectural-space exploration of approximate multipliers. In *Proceedings of the 35th International Conference on Computer-Aided Design*, page 80. ACM, 2016. (Cited on pages 3, 16, 23, 24, 26, 28, 58, 66, 87, 88, 89, and 91).
- [102] L. Renganarayana, V. Srinivasan, R. Nair, and D. Prener. Programming with relaxed synchronization. In *Proceedings of the 2012 ACM workshop on Relaxing synchronization for multicore and manycore scalability*, pages 41–50. ACM, 2012. (Cited on page 18).
- [103] M. Rinard and P. Stanley-Marbell. Reducing serial i/o power in error-tolerant applications by efficient lossy encoding. In *2016 53rd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2016. (Cited on page 19).
- [104] A. Roldao-Lopes, A. Shahzad, G. A. Constantinides, and E. C. Kerrigan. More flops or more precision? accuracy parameterizable linear equation solvers for model predictive control. In *Field Programmable Custom Computing Machines, 2009. FCCM'09. 17th IEEE Symposium on*, pages 209–216. IEEE, 2009. (Cited on pages 4, 32, and 39).
- [105] P. Roy, R. Ray, C. Wang, and W. F. Wong. Asac: Automatic sensitivity analysis for approximate computing. In *ACM SIGPLAN Notices*, number 5, pages 95–104. ACM, 2014. (Cited on pages 15 and 34).
- [106] P. Roy, J. Wang, and W. F. Wong. Pac: program analysis for approximation-aware compilation. In *Proceedings of the 2015 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, pages 69–78. IEEE Press, 2015. (Cited on pages 15, 16, and 34).
- [107] S. Salvini and S. J. Wijnholds. Fast gain calibration in radio astronomy using alternating direction implicit methods: Analysis and applications. *Astronomy & Astrophysics*, 571:A97, 2014. (Cited on pages 4, 5, 32, 37, 38, 39, 56, 80, and 104).

- [108] A. Sampson. *Hardware and software for approximate computing*. PhD thesis, University of Washington, 2015. (Cited on pages 14, 32, and 34).
- [109] J. San Miguel, M. Badr, and N. E. Jerger. Load value approximation. In *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 127–139. IEEE, 2014. (Cited on page 19).
- [110] J. Schlachter, V. Camus, K. V. Palem, and C.ENZ. Design and applications of approximate circuits by gate-level pruning. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(5):1694–1702, 2017. (Cited on pages 11, 20, and 29).
- [111] M. L. Schmatz, R. Jongerius, G. Dittmann, A. Anghel, T. Engbersen, J. van Lunteren, and P. Buchmann. Scalable, efficient asics for the square kilometre array: From a/d conversion to central correlation. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7505–7509. IEEE, 2014. (Cited on pages 97 and 104).
- [112] G. Semeraro, G. Magklis, R. Balasubramonian, D. H. Albonese, S. Dwarkadas, and M. L. Scott. Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling. In *Proceedings Eighth International Symposium on High Performance Computer Architecture*, pages 29–40. IEEE, 2002. (Cited on page 20).
- [113] M. Shafique and S. Garg. Computing in the dark silicon era: Current trends and research challenges. *IEEE Design & Test*, 34(2):8–23, 2016. (Cited on page 2).
- [114] M. Shafique, W. Ahmad, R. Hafiz, and J. Henkel. A low latency generic accuracy configurable adder. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2015. (Cited on pages 3, 21, 58, and 87).
- [115] M. Shafique, R. Hafiz, S. Rehman, W. El-Harouni, and J. Henkel. Invited: Cross-layer approximate computing: From logic to architectures. In *Design Automation Conference (DAC), 2016 53rd ACM/EDAC/IEEE*, pages 1–6. IEEE, 2016. (Cited on pages 2, 3, 16, 32, and 33).
- [116] D. Shin and S. K. Gupta. Approximate logic synthesis for error tolerant applications. In *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*, pages 957–960. IEEE, 2010. (Cited on pages 20 and 29).
- [117] S. Sidiroglou-Douskos, S. Misailovic, H. Hoffmann, and M. Rinard. Managing performance vs. accuracy trade-offs with loop perforation. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pages 124–134. ACM, 2011. (Cited on page 18).
- [118] P. Stanley-Marbell and M. Rinard. Efficiency limits for value-deviation-bounded approximate communication. *IEEE Embedded Systems Letters*, 7(4):109–112, 2015. (Cited on page 19).
- [119] P. Stanley-Marbell, M. Rinard, et al. Error-efficient computing systems. *Foundations and Trends in Electronic Design Automation*, 11(4):362–461, 2017. (Cited on page 9).

- [120] M. B. Sullivan and E. E. Swartzlander. Truncated error correction for flexible approximate multiplication. In *2012 Conference Record of the Forty Sixth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, pages 355–359. IEEE, 2012. (Cited on page 58).
- [121] B. Thwaites, G. Pekhimenko, H. Esmailzadeh, A. Yazdanbakhsh, J. Park, G. Mururu, O. Mutlu, and T. Mowry. Rollback-free value prediction with approximate loads. In *2014 23rd International Conference on Parallel Architecture and Compilation Techniques (PACT)*, pages 493–494. IEEE, 2014. (Cited on page 19).
- [122] S. L. Toral, J. M. Quero, and L. G. Franquelo. Stochastic pulse coded arithmetic. In *IEEE International Symposium on Circuits and Systems.*, volume 1, pages 599–602, 2000. (Cited on page 10).
- [123] S. Ullah, S. Rehman, B. S. Prabakaran, F. Kriebel, M. A. Hanif, M. Shafique, and A. Kumar. Area-optimized low-latency approximate multipliers for fpga-based hardware accelerators. In *Proceedings of the 55th Annual Design Automation Conference*, page 159. ACM, 2018. (Cited on pages 58 and 87).
- [124] M. Van Haarlem, M. Wise, A. Gunst, G. Heald, J. McKean, J. Hessels, A. De Bruyn, R. Nijboer, J. Swinbank, R. Fallows, et al. Lofar: The low-frequency array. *Astronomy & Astrophysics*, 556:A2, 2013. (Cited on pages 5, 38, 46, 52, 74, 80, 104, and 117).
- [125] G. V. Varatkar and N. R. Shanbhag. Energy-efficient motion estimation using error-tolerance. In *Proceedings of the 2006 international symposium on Low power electronics and design*, pages 113–118, 2006. (Cited on page 20).
- [126] B. Veenboer, M. Petschow, and J. W. Romein. Image-domain gridding on graphics processors. In *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 545–554. IEEE, 2017. (Cited on pages 5 and 118).
- [127] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy, and A. Raghunathan. Salsa: systematic logic synthesis of approximate circuits. In *DAC Design Automation Conference 2012*, pages 796–801. IEEE, 2012. (Cited on page 25).
- [128] S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan. Approximate computing and the quest for computing efficiency. In *Proceedings of the 52nd Annual Design Automation Conference*, page 120. ACM, 2015. (Cited on page 2).
- [129] S. Venkataramani, A. Raghunathan, J. Liu, and M. Shoab. Scalable-effort classifiers for energy-efficient machine learning. In *Proceedings of the 52nd Annual Design Automation Conference*, page 67. ACM, 2015. (Cited on page 18).
- [130] A. K. Verma, P. Brisk, and P. Jenne. Variable latency speculative addition: A new paradigm for arithmetic circuit design. In *Proceedings of the conference on Design, automation and test in Europe*, pages 1250–1255. ACM, 2008. (Cited on pages 21, 58, and 87).
- [131] Q. Wu, Y. Zhu, X. Wang, M. Li, J. Hou, and A. Masoumi. Exploring high efficiency hardware accelerator for the key algorithm of square kilometer array telescope data processing. In *2017 IEEE 25th Annual International Symposium on*

Field-Programmable Custom Computing Machines (FCCM), pages 195–195. IEEE, 2017. (Cited on page 5).

154

[132] Q. Xu, T. Mytkowicz, and N. S. Kim. Approximate computing: A survey. *IEEE Design & Test*, 33(1):8–22, 2016. (Cited on pages 2 and 9).

[133] R. Ye, T. Wang, F. Yuan, R. Kumar, and Q. Xu. On reconfiguration-oriented approximate adder design and its application. In *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 48–54. IEEE, 2013. (Cited on page 21).

[134] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo, and Z. H. Kong. Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing. *IEEE transactions on very large scale integration (VLSI) systems*, 18(8):1225–1229, 2009. (Cited on pages 21 and 22).

LIST OF PUBLICATIONS

- [G:1] G.A. Gillani and A.B.J. Kokkeler. Improving error resilience analysis methodology of iterative workloads for approximate computing. In *14th ACM International Conference on Computing Frontiers*. ACM, 2017.
- [G:2] G.A. Gillani, A. Krapukhin, and A.B.J. Kokkeler. Energy-efficient approximate least squares accelerator. In *16th ACM International Conference on Computing Frontiers*. ACM, 2019.
- [G:3] G.A. Gillani, M.A. Hanif, M. Krone, S.H. Gerez, M. Shafique, and A.B.J. Kokkeler. Squash: Approximate square-accumulate with self-healing. *IEEE Access*, 6:49112–49128, 2018.
- [G:4] G.A. Gillani, M.A. Hanif, B. Verstoep, S.H. Gerez, M. Shafique, and A.B.J. Kokkeler. Macish: Designing approximate mac accelerators with internal-self-healing. *IEEE Access*, 7:77142–77160, 2019.
- [G:5] G.A. Gillani and A.B.J. Kokkeler. Go green radio astronomy: Approximate computing perspective: Opportunities and challenges: Poster. In *16th ACM International Conference on Computing Frontiers*. ACM, 2019.
- [G:6] L. Oudshoorn, G.A. Gillani, and A.B.J. Kokkeler. Impact of delay propagation on ntv pcmos design. *ICT-ENERGY Letters*, 2016.
- [G:7] P. Stanley-Marbell, A. Alaghi, M. Carbin, E. Darulova, L. Dolecek, A. Gerstlauer, G.A. Gillani, D. Jevdjic, et al. Exploiting errors for efficiency: A survey from circuits to applications. *ACM Computing Surveys (CSUR)*, 53(3), 2020.

THIS THESIS

```
@phdthesis{gillani2020:thesis,  
  author={Gillani, G.A.},  
  title={Exploiting Error Resilience For Hardware Efficiency -- Targeting  
        Iterative and Accumulation Based Algorithms},  
  school={University of Twente},  
  year={2020},  
  number={DSI Ph.D. Thesis Series No. 20-004},  
  issn={2589-7721},  
  isbn={978-90-365-5011-6},  
  doi={10.3990/1.9789036550116}  
}
```



ISBN 978-90-365-5011-6



9 789036 550116