Design of approximate and precision-scalable circuits for embedded multimedia and neural-network processing

Thèse N° 9144

Présentée le 1er février 2019

à la Faculté des sciences et techniques de l'ingénieur
Laboratoire de circuits intégrés
Programme doctoral en microsystèmes et microélectronique

pour l'obtention du grade de Docteur ès Sciences

par

Vincent Frédéric CAMUS

2019

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

I am just a child who has never grown up.
I still keep asking these ‘how’ and ‘why’ questions.
Occasionally, I find an answer.

— Stephen Hawking

Acknowledgements

First and foremost, I would like to thank my thesis advisor, Prof. Christian Enz, for his continuous support. Besides all his technical advices, he has always kept me motivated and has encouraged me to push myself and aim for the highest goals.

My sincere thanks also go to Prof. Marian Verhelst, who offered me an opportunity to join her team and to conduct thrilling studies in the field of Machine Learning. She has helped me a lot with insightful advices and the finest guidance towards new research directions.

I would also like to thank all the members of my thesis committee, Prof. Catherine Dehollain, Dr. Marc Pons Sole, Dr. Alain Vachoux, Prof. Marian Verhelst and Prof. Christian Enz for carefully reading my manuscript and for their constructive questions and comments.

Thank you to Lysiane Bourquin, Sandrine Piffaretti, Marcella Giovannini, Lucie Auberson and Marie Halm for their administrative support, to Dr. Alain Vachoux for his support with CAD tools, as well as Aymeric, Alex, Nadine, Patrick, Ryan and all the members of the MBC for their technical support.

A big thanks to my colleagues, Jérémy, Mattia, Vladimir, Raffaele, Francesco, Sammy, Camilo, Arnout, Chunmin, Alessandro, Farzan, Nino, Assim, Minhao, Daniel, Marta, Huaiqi, Claudio, Raghav, Anurag and Maria-Anna. They all made the time spent at work enjoyable, with fun or serious conversations, technical or non-technical discussions, and French or Italian food! I would like to dedicate a special shout-out to the colleagues and friends at LMTS (EPFL) and MICAS (KU Leuven) to have integrated me in the social life of their labs.

Finally I would like to thank my parents, my sister and my family, who always supported me in my life and in my studies, and all my friends who are bringing joy and fun to my life. Last but not least, I want to thank my wife Saki who always encourages me and continuously fills my life with love and happiness.

Vincent Camus
December 18, 2018

Abstract

Density, speed and energy efficiency of integrated circuits have been increasing exponentially for the last four decades following Moore's law. However, power and reliability pose several challenges to the future of technology scaling. Approximate computing has emerged as a promising candidate to improve performance and energy efficiency beyond scaling. Approximate circuits explore a new trade-off by intentionally introducing errors to overcome the limitations of traditional designs. This paradigm has led to another opportunity to minimize energy at run time with precision-scalable circuits, which can dynamically configure their accuracy or precision. This thesis investigates several approaches for the design of approximate and precision-scalable circuits for multimedia and deep-learning applications.

This thesis first introduces architectural techniques for designing approximate arithmetic circuits, in particular, two techniques called Inexact Speculative Adder (ISA) and Gate-Level Pruning (GLP). The ISA slices the addition operation into multiple shorter sub-blocks executed in parallel. It features a shorter speculative overhead and a novel error correction-reduction scheme. The second technique, GLP, consists in a CAD tool that removes the least-significant logic gates from a circuit in order to reduce energy consumption and silicon area. These conventional techniques have been successfully combined together or with overclocking.

The second part of this thesis introduces a novel concept to optimize approximate circuits by fabrication of false timing paths, i.e. critical paths that cannot be logically activated. Co-designing circuit timing together with functionality, this method proposes to monitor and cut critical paths to transform them into false paths. This technique is applied to an approximate adder, called the Carry Cut-Back Adder (CCBA), in which high-significance stages can cut the carry propagation chain at lower-significance positions, guaranteeing a high accuracy.

The third part of this thesis investigates approximate circuits within bigger datapaths and applications. The ISA concept is extended to a novel Inexact Speculative Multiplier (ISM). ISM, ISA and GLP techniques are then used to build approximate Floating-Point Units (FPU) taped-out in a 65nm quad-core processor. Approximate FPU circuits are validated through a High-Dynamic Range (HDR) image tone-mapping application. HDR imaging is a rapidly growing area in mobile phones and cameras extensively using floating-point computations. Results of the application show no visible quality loss, with image PSNR ranging from 76 dB using the pruned FPU to 127 dB using the speculative FPU.

The final part of this thesis reviews and complements precision-scalable Multiply-Accumulate (MAC) accelerators for deep learning applications. Deep learning has come with an enormous computational need for billions of MAC operations. Fortunately, reduced precision has demonstrated benefits with minimal loss in accuracy. Many works have recently shown configurable MAC architectures optimized for neural-network processing, either with parallelization or bit-serial approaches. In this thesis, the most prominent ones are reviewed, implemented and compared in a fair way. A hybrid precision-scalable MAC design is also proposed. Finally, an analysis of power consumption and throughput is carried out to figure out the key trends for reducing computation costs in neural-network processors.

Keywords

Approximate computing, approximate circuits, low-power digital circuits, approximate adders, speculative adders, approximate multipliers, floating-point unit (FPU), multiply-accumulate (MAC), false timing paths, timing errors, scalable precision, neural networks, high-dynamic range (HDR), image tone mapping.

Résumé

La densité, la vitesse et l'efficacité énergétique des circuits intégrés ont augmenté de façon exponentielle au cours des quatre dernières décennies, se conformant à la remarquable prédiction de Gordon Moore. Cependant, la consommation énergétique et la fiabilité des circuits posent plusieurs défis risquant de freiner cette progression. Le *calcul approximatif* (approximate computing) est apparu comme une méthode prometteuse pour continuer à réduire la consommation ou augmenter les performances malgré les difficultés liées à la miniaturisation. La conception de *circuits approximatifs* explore ce nouveau compromis en acceptant ou introduisant intentionnellement des erreurs pour surmonter les limites des circuits traditionnels. Ce paradigme a ouvert la voie à de nouvelles techniques de réduction dynamique de consommation, notamment grâce aux *circuits à précision variable*, qui permettent de configurer de manière dynamique leur exactitude ou précision. Cette thèse examine plusieurs approches de conception de circuits approximatifs et à précision variable pour applications en multimédia et en intelligence artificielle.

Cette thèse présente d'abord les techniques architecturales permettant de concevoir des circuits arithmétiques approximatifs, en particulier deux techniques appelées *Inexact Speculative Adder (ISA)* et *Gate-Level Pruning (GLP)*. La première technique, l'ISA, divise l'opération d'addition, c'est-à-dire la propagation de la chaîne de retenue, en plusieurs sous-blocs exécutés en parallèle en utilisant des retenues internes spéculées. L'ISA est doté d'un bloc de spéculation plus rapide et d'un nouveau mécanisme de réduction et correction d'erreurs. La deuxième technique, GLP, est un outil de CAO qui supprime les portes logiques les moins importantes d'un circuit afin d'en réduire la consommation d'énergie et la surface de silicium. Couplé à un modèle de prédiction d'erreurs de timing, l'ISA démontre une forte résistance à l'overclocking, ce qui offre un autre degré de liberté entre énergie et précision.

La deuxième partie de cette thèse introduit un nouveau concept permettant d'optimiser les circuits approximatifs en fabriquant de *faux chemins critiques*, c'est-à-dire des chemins critiques qui ne peuvent pas être logiquement activés. Optimisant à la fois les délais du circuit et sa fonctionnalité, cette méthode propose de surveiller et de couper les chemins critiques pour les transformer en faux chemins critiques. Un additionneur approximatif appelé *Carry Cut-Back Adder (CCBA)* est proposé, dans lequel certains calculs de retenue en aval de l'addition peuvent interrompre la chaîne de propagation de retenue à des positions en amont, de moindre importance, garantissant ainsi une grande précision tout en apportant d’énormes gains en énergie.

La troisième partie de cette thèse étudie l’intégration de techniques d’approximation au sein de plus grandes unités de calcul ainsi que leur application pour le traitement d’images. Le concept de l’ISA est d’abord étendu pour concevoir un multiplicateur spéculatif, l’*Inexact Speculative Multiplier (ISM)*. Les techniques ISM, ISA et GLP sont ensuite utilisées pour construire des unités à virgule flottante (FPU) approximatives, intégrées dans un processeur multi-cœur. Ces circuits FPU approximatifs sont validés par le biais d’une application embarquée de mappage ton local d’images HDR (à grande gamme dynamique). L’imagerie HDR est un domaine en pleine croissance dans les téléphones mobiles et les appareils photo utilisant de manière extensive les calculs en virgule flottante. Les résultats de l’application ne montrent aucune perte de qualité visible, avec un PSNR d’image allant de 76 dB avec la FPU prunée à 127 dB avec la FPU spéculative.

La dernière partie de cette thèse passe en revue, compare et complète les accélérateurs de multiplication-accumulation (MAC) à précision variable pour les applications d’apprentissage profond. L’apprentissage profond a engendré un énorme besoin en calcul pour des milliards d’opérations MAC. Heureusement, l’utilisation de précisions réduites a démontré de gros bénéfices avec une détérioration de qualité minime, ouvrant la voie à l’intelligence artificielle embarquée dans les appareils mobiles et l’Internet des Objets (IoT). De nombreux travaux ont récemment présenté des architectures MAC configurables à faible précision spécialement optimisées pour le traitement de réseaux de neurones, avec des capacités de parallélisation ou des approches sérielles. Dans cette thèse, les plus importantes sont passées en revue, mises en œuvre et comparées de manière équitable. Un MAC hybride sériel multi-bit est également proposé. Enfin, une analyse de la consommation d’énergie et du débit selon différents scénarios de précision est réalisée afin de réduire les coûts de calcul des réseaux de neurones.

Mots clés

Calcul approximatif, circuits approximatifs, circuits numériques à basse consommation, additioneurs approximatifs, additioneurs spéculatifs, multiplieurs approximatifs, unités à virgule flottante (FPU), multiplication-accumulation (MAC), faux chemins critiques, circuits à précision variable, réseaux de neurones, high-dynamic range (HDR), mappage ton local.

Contents

List of figures

List of tables

Nomenclature

ABPER	Average Bit-level Prediction Error Rate
ACA	Almost Correct Adder
ACAA	Accuracy-Configurable Approximate Adder
ADD	Adder or sub-adder block
AVPE	Average Value-level Predictive Error
BALL	Bit-Accurate Logic Level
CASSIS	Characterization with Adaptive Sample-Size Inferential Statistic
CCBA	Carry Cut-Back Adder
CCBM	Carry Cut-Back Multiplier
CI	Confidence Interval
CLA	Carry Look Ahead
CNN	Convolutional Neural Network
COMP	Error compensation block
CONV	Convolutional stage or layer
DNN	Deep Neural Network
DVAFS	Dynamic Voltage-Accuracy-Frequency Scaling
D&C	Divide and Conquer
E	Arithmetic error
ESA	Equal Segmentation Adder
ETAII	Error-Tolerant Adder type II
ETAIIM	Error-Tolerant Adder type II Modified
ETAIV	Error-Tolerant Adder type IV
ETBA	Error-Tolerant Balancing Adder
FPU	Floating-Point Unit
GCSA	Generate-signals-exploited Carry Speculation Adder
GLP	Gate-Level Pruning
GOPS	Giga Operations Per Second
HDR	High-Dynamic Range
ISA	Inexact Speculative Adder
ISM	Inexact Speculative Multiplier
LOA	Lower-part-OR Adder
LSB	Least Significant Bit

MAC	Multiply-Accumulate
MSB	Most Significant Bit
PAP	Power-Area Product
PDAP	Power-Delay-Area Product
PROP	Carry propagate block
PSNR	Peak Signal to Noise Ratio
PVD	Pixel Value Difference
QNN	Quantized Neural Network
RE	Relative Error
RFC	Random Forest Classification
RMS	Root Mean Square
RSD	Relative Standard Deviation
RTL	Register Transfer Level
SAP	Significance-Activity Product
SDF	Standard Delay Format
SNR	Signal to Noise Ratio
SPEC	Carry speculator block
STA	Static Timing Analysis
TOPS	Tera Operations Per Second

1 Introduction and related work

1.1 Introduction

By 2025, one trillion devices will be connected to the Internet and enhanced with embedded electronics and sensors [1]. As shown on Fig. 1.1 [2], the Internet of Things (IoT) is expected to deploy 1 trillion nodes with high sensing ability to interact with the physical world, but low computing abilities. This is unfortunate, as the fast growing trend of multimedia and machine-learning applications capable of mining through those collected data is computationally expensive for applications such as face recognition, augmented reality, keyword detection or other human activity recognition. One might think that data could be transferred to the Cloud in order to be analyzed. However, the power budget of IoT nodes does not allow such transfer by wireless communication that would require data rates of several Mbps in the case of vision and audio applications. Security, privacy, always-on or real-time would also challenge that strategy. This is why the trend is to look for solutions to increase computational efficiency of IoT nodes.

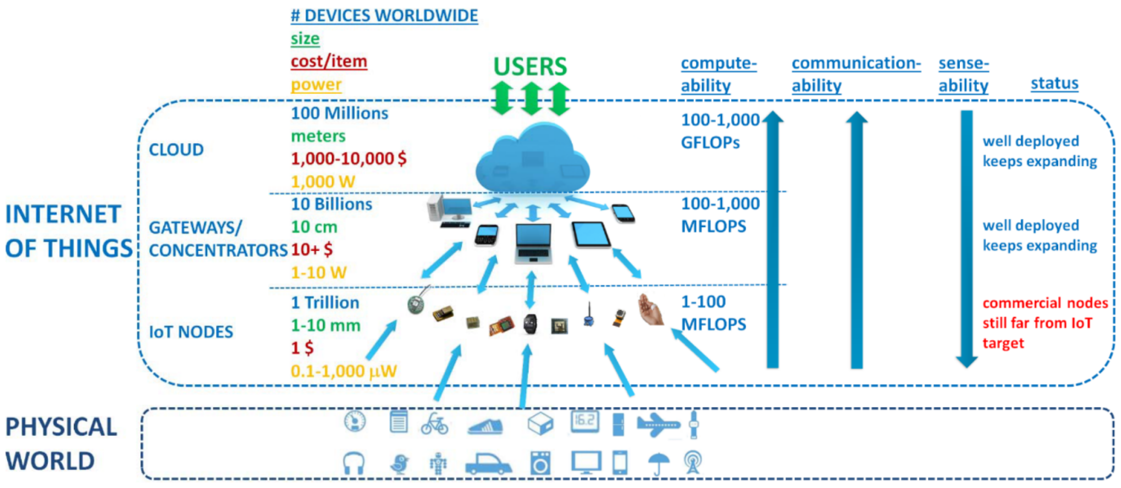


Figure 1.1 – Growth and architecture of the IoT expected for 2025 [2].

As illustrated in Fig. 1.2 [3], the power issue will not be solved with general-purpose processors, i.e. CPUs. The low power budget of battery-powered or energy-harvested IoT nodes requires dedicated hardware accelerators with much lower energy per operation.

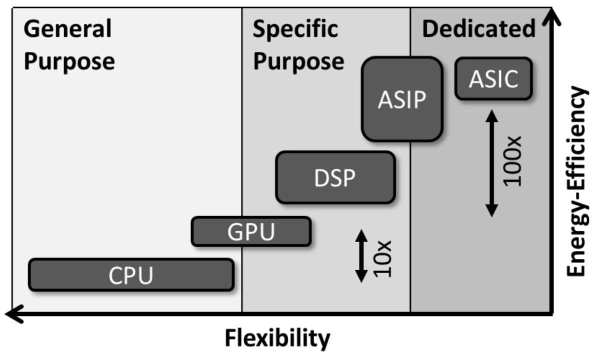


Figure 1.2 – Energy efficiency versus flexibility for different deep-learning platforms [3].

Traditional low-power design techniques cannot support the estimated growth of IoT objects and in the same time keep their energy consumption within sustainable bounds. Performance and energy efficiency of integrated circuits have increased exponentially for almost five decades, following Gordon Moore's remarkable prediction, but power and reliability pose several challenges to the future of technology scaling. Power has definitely emerged as a critical concern due to the poor scaling of V_{th}, traditional low-power design techniques cannot support the estimated growth of the IoT and in the same time keep their energy consumption within sustainable bounds. On the other side, transistor miniaturization reaching atomic scale has led to tremendous Process-Voltage-Temperature (PVT) variations and ever-increasing fabrication costs.

Unfortunately, achieving low power and robustness against variability requires complex and conflicting design constraints. For example, while power efficiency calls for voltage down-scaling and minimization of hardware, robustness demands higher voltage, larger transistors and additional correction or redundancy. As a result, designers are being pushed to seek new energy-efficient computing techniques to meet the increasing demand of data processing.

The concept of *error tolerance*, i.e. accepting error in a design to save resources, is well known in many abstraction layers and is already implicit in digital signal processing as the representation of real numbers is approximated due to the finite number of bits. With the exploding amount of data being processed in the IoT nodes and cloud, a wide range of applications can trade accuracy without compromising the functionality or the user experience. In multimedia applications, a small proportion of errors stays imperceptible to humans. Iterative applications like vision and tracking are inherently resilient to errors since those can be compensated in the succeeding frames or steps. Finally, machine-learning applications, executing statistical algorithms or neural networks, show a high degree of resilience to faults or approximations.

In that context, *approximate computing* has emerged as a new design paradigm, embracing inexactness and trading off a controllable amount of accuracy for significant power savings or performance boost beyond technology scaling. With the expected end of Moore's Law, it has become a leading trend in many abstraction layers, and particularly in circuit design. Designing *approximate circuits* explores a new trade-off by intentionally introducing errors to overcome the limitations of traditional circuit architectures. *Precision-scalable circuits* have appeared as another approach to minimize energy at run time by allowing dynamic reconfiguration of their accuracy or precision.

This thesis investigates several approaches for the design of approximate and precision-scalable circuits for multimedia and deep-learning applications. The detailed organization is as follow:

- Chapter 2 presents architectural techniques for designing approximate arithmetic circuits, in particular, two techniques called Inexact Speculative Adder (ISA) and Gate-Level Pruning (GLP).
- Chapter 3 introduces a novel method to optimize arithmetic circuits by artificially inserting and exploiting false paths, i.e. critical paths that cannot be logically activated. This concept is applied to approximate arithmetic circuits, in particular the Carry Cut-Back Adder (CCBA), which is then exhaustively compared to state-of-the-art circuits.
- Chapter 4 investigates the application of approximate circuit techniques within a bigger accelerator. Approximate Floating-Point Units (FPU) are integrated and taped-out in a multi-core processor, and validated through a High-Dynamic Range (HDR) image tone-mapping application.
- Chapter 5 finally presents different precision-scalable Multiply-Accumulate (MAC) units. It reviews, implements and compares them in terms of energy, throughput and area, aiming to understand the optimal architectures and analyze the impact of scalability to reduce computation costs in neural-network processing.
- The appendix shows a stand-alone work on characterization of approximate circuits using inferential statistics, and an early investigation on quantization of neural networks using low-precision logarithmic data encoding.

1.2 State of the art

Approximate computing has been investigated at all levels of abstraction, such as voltage-frequency-precision scaling at circuit level [4] or significance-driven computation at algorithmic level [5]. Table 1.1 [6] highlights the different approximate computing techniques, classifying them according to the type of errors, the property traded for errors, and the altered resources.

Table 1.1 – Taxonomy of approximate computing techniques from circuit to OS level [6].

Layer	**Technique**	**Error type**		**Property traded for**			**Affected resource**		
		Deterministic	*Non-deterministic*	*Energy*	*Runtime*	*Data density*	*Computation*	*Data storage/movement*	*Physical world I/O*
Circuit	Sensor value approximation		•	•					•
	Probabilistic sensor comms.	•		•					•
	Probabilistic computing		•	•			•		
	Stochastic computing	•			•		•		
	Voltage overscaling		•	•			•		
	Logic pruning	•		•	•	•	•		
	Approximate addition	•		•	•	•	•		
	Approximate multiplication	•		•	•	•	•		
	RTL approximations	•		•	•	•	•		
	Approx. high-level synthesis	•		•	•	•	•		
	Voltage overscaled SRAM		•	•				•	
Architecture	Deterministic lossy I/O	•		•				•	
	Voltage overscaling		•	•			•	•	
	Analog neural acceleration		•	•	•		•		
	Digital neural acceleration	•		•	•		•		
	Anytime computation	•			•		•		
	Approximate reads		•	•	•			•	
	Approximate writes		•	•	•			•	
	Reuse of failed data blocks		•			•		•	
	Variable redundancy		•			•		•	
	Approx. cache de-duplication	•				•		•	
	Load-value approximation	•		•	•			•	
	Low-refresh DRAM		•	•				•	
	In-network lossy compression	•				•		•	
Program	Floating-point optimization	•			•		•		
	Neural approximation	•		•	•		•		
	Relaxed parallelization		•		•		•		
	Accuracy verification		•	•			•		
	Data-parallel kernel approx.	•			•		•		
	Isolation of approx. data	•	•	•			•	•	
	Algorithm approximation	•			•		•		
	Loop perforation	•			•		•		
OS	Display color approximation	•		•					•
	Drivers for approx. sensors		•	•					•
	Dynamic accuracy adaptation	•		•			•		
	Task-level approximation		•	•			•		

The following sections describe a few examples of approximate computing techniques exploiting technological knobs, circuit and design techniques, and program-level approaches.

1.2.1 Approximate circuits using technological knobs

Voltage overscaling

In order to ensure correct functionality, digital circuits are traditionally designed with large timing margins to accommodate for process variation, voltage variation, temperature variation, and aging effects. Added at design time, for instance with increased design sizing, or through reduction of clock frequency, these margins result in losses in performance, energy efficiency, silicon area, and cost.

But power consumption decreases quadratically with supply voltage, which makes it a method for energy reduction. The maximal clock frequency, determined by the critical path, is normally scaled proportional to avoid timing violation, leading to performance degradation. *Voltage overscaling* [7, 8], i.e. scaling voltage and frequency below the critical point allowing acceptable timing error start to occur. This strategy can also be applied non uniformly over the circuits, for instance to scale voltage differently depending on the most or least significant bits of an adder to achieve lower overall error magnitude [9].

Timing-error detection

To better control the level of errors, in-situ error detection and correction (EDAC) techniques have been explored, estimating circuit delay with on-chip structures and allowing to dynamically adjust voltage or frequency to make the circuit operate without margins on the edge of failure. A common approach is to introduce specialized registers on critical and near-critical paths of data processing pipeline stages to monitor the timing violations and then acting to recover functionality of the design if a violation occurs. This strategy eliminates fast (e.g., voltage drops, local hot spots), slow (e.g., ambient temperature changes), and static (e.g., process variations) timing margins.

The Razor flip-flop [10], shown in Fig. 1.3, is probably the most famous technique used to get rid of the safety margins in digital circuits. It double samples pipeline stages values, once with a fast clock and again with a time-borrowing delayed clock. If timing errors start to occur the two latches have different outputs, hence the timing error can be detected and the correct value can be recovered with an extra clock cycle, ensuring error free operation. However, this error detection and correction comes at the cost of an hardware overhead. Other examples of EDAC techniques include RazorII [11] or Bubble Razor [12].

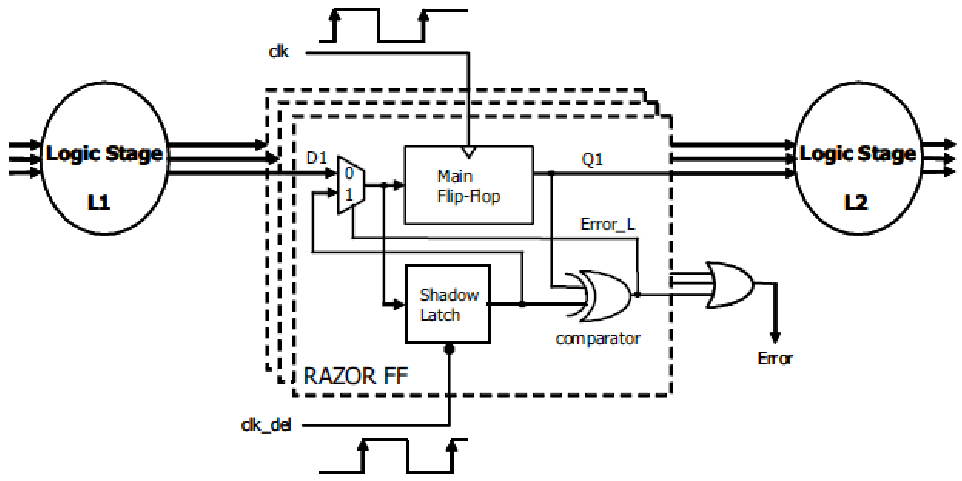
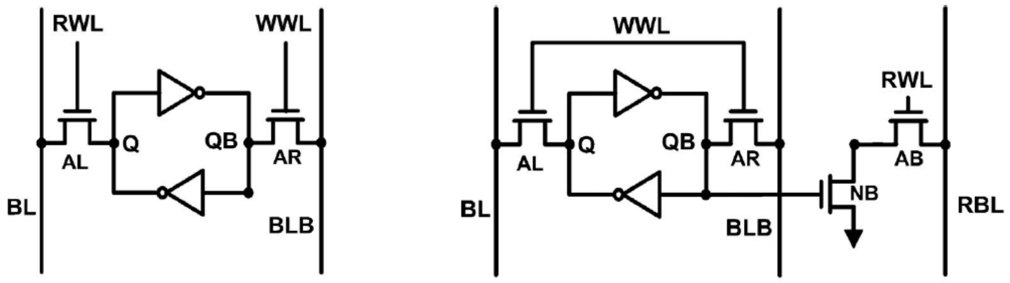


Figure 1.3 – Pipeline augmented with Razor flip-flops [10].

Voltage overscaled memories

Memory is ubiquitous in today's circuits, and its power consumption is always a major part of the entire system. Power consumption can also be lowered by voltage scaling, unfortunately, conventional 6T (6-transistor) SRAM bit cells, as displayed in Fig. 1.4a, are very sensitive to voltage scaling, leading to a significant number of read or write errors. To overcome these issues, hybrid SRAMs with robust 8T bit cells, as on Fig. 1.4b, have been proposed [13, 14] in order to reduce errors on high-significance data bits.



(a) 6T bit cell (b) 8T bit cell

Figure 1.4 – Schematics of 6T (a) and 8T (b) SRAM bit cells [13].

SRAM overscaling has also been proposed [15] for convolutional neural networks, and has demonstrated both leakage savings and memory-access power savings. Additionally, error rate in classification algorithms could be improved by training the neural network together with the overscaled memory.

1.2.2 Circuit-level design techniques

As timing errors are very difficult to predict, voltage overscaling is usually done at the cost of error correction or additional monitoring hardware. Architectural techniques presented in this section trade exactness or precision for power consumption, speed, and silicon area without further overhead. These technique are particularly well suited to arithmetic operators and hardware accelerators.

Circuit-level speculation

Speculative adders[16] exploit the fact that carry propagate sequences in additions are typically short, making it possible to estimate intermediate carries using a limited number of previous stages. They split the binary addition into several sub-paths executed concurrently for higher execution speed and energy efficiency, but at the risk of generating occasionally incorrect results. Thus, the critical path of the adder can be divided in two or more shorter paths, relaxing constraints over the entire design and improving the speed, area and power beyond the theoretical bounds of exact adders.

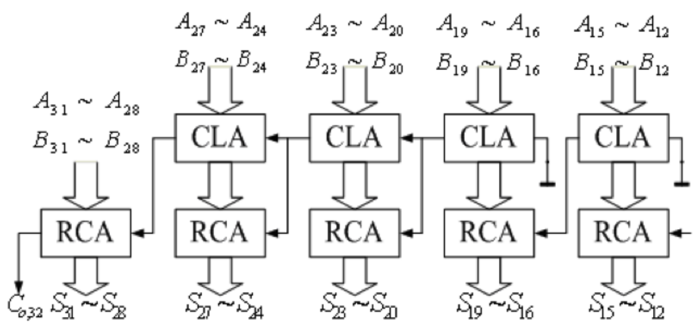
A number of speculative adders have been proposed in literature[17] with different approaches in order to reduce the error frequency or magnitude. The ETAII [18], shown in Fig. 1.5a, consists of regular sub-adder blocks with input carries speculated from Carry Look Ahead (CLA) blocks of the same length. The ETBA adder [19], displayed on Fig. 1.5b, adds sum balancing multiplexed blocks to mitigate relative errors. These technique show particularly large improvements in terms of area and delay.

Complexity reduction

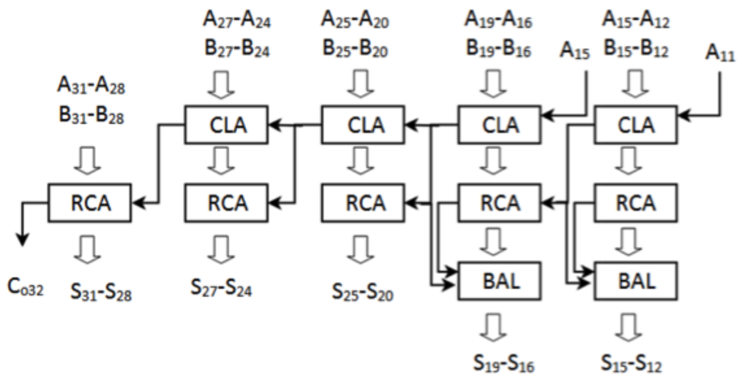
Many papers presented inexact arithmetic circuits based on complexity reduction. In the bio-inspired adder [20], the Least Significant Bits (LSBs) of the sum are simply approximated by OR gates instead of the full-adder cells, leading to direct area power and delay savings. Another approach to reduce circuit complexity consists in removing some transistors of the full-adder cell [21].

A more general method proposed called probabilistic logic minimization [22] introduces bit-flips in the Karnaugh maps of boolean functions to minimize its circuit implementation, as depicted in Fig. 1.6.

Probabilistic pruning [23], descendant of logic minimization, is a CAD technique consisting in removing circuit elements to trade exactness of computation against area and power.



(a) ETAII adder



(b) ETBA adder

Figure 1.5 – Block diagrams of the ETAII [18] (a) and the ETBA [19] (b) speculative adders.

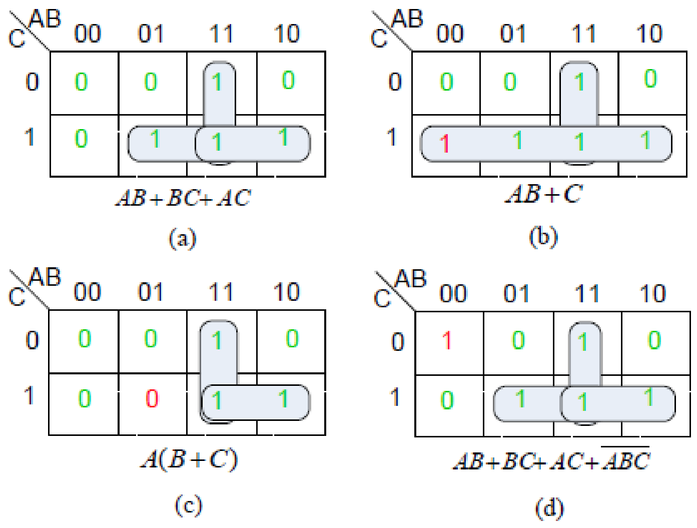


Figure 1.6 – Example of logic minimization. (a) represents the initial Karnaugh map of the function, (b) shows a favorable 0-to-1 bit-flip, (c) a favorable 1-to-0 bit-flip, and (d) a non-favorable 0-to-1 bit-flip [22].

Precision scaling

Deep-learning algorithms, required for robust audio-visual data inference, have proven superior recognition accuracies, but they inevitably come with large computational load, preventing low-power operation. Convolutional Neural Networks (CNNs) are the state-of-the-art deep-learning algorithm for image recognition and classification. They are now outperforming humans! They have recently been proven robust to basic approximations by scaling the bit-widths of Multiply-Accumulate (MAC) array.

The Origami processor [24], dedicated to CNN acceleration and running with 12-bit arithmetic, achieves a performance per watt of 803 GOPS/W while most conventional 16-bit CNN processors do not achieve more than 500. In the ConvNet processor [3], dynamic precision scalability have allowed improving energy efficiency up to 2.7 TOPS/W. Reduced bit-width has demonstrated energy gains up to 1.9x improvements in the overall CNN power consumption in layer 2 of the AlexNet [25] neural network. Combined with dynamic voltage-frequency scalability, it has proven up to 2.5x power-efficiency gains, as shown on Fig. 1.7. Further improvement [26] to reuse inactive arithmetic due to the scaled bit-width has improved efficiency even up to 10 TOPS/W.

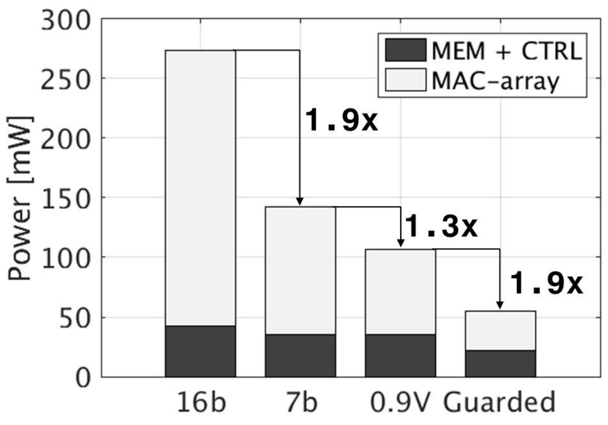


Figure 1.7 – Influence of different techniques on the CNN power consumption. Up to 5x gain is achieved through precision, voltage scaling, and sparse operator guarding [3].

1.2.3 Software-level approximations

Several techniques of approximate computing have been developed at algorithm, compiler and software levels.

An example of approximate computing at algorithm level is computation skipping [27]. This method dynamically skips block functions of computation to minimize bit-rate while maximizing video quality for the High Efficiency Video Coding algorithm (HEVC, also known as the H.265 video compression standard).

Another example is loop perforation [28], which can be seen as a special case of task skipping. Loop perforation trades accuracy for performance by skipping some unimportant iterations of a loop to reduce computational overhead.

Other efforts have been made on compilers and programming languages. The EnerJ [29] and FlexJava [30] languages allow programmers to annotate variables or functions that can be executed with errors or approximations.

1.2.4 Discussion

All the different circuit-level techniques presented above are exploited in some form in this work. Chapters 2 and 3 build upon speculative arithmetic circuits. Complexity reduction and voltage-frequency overscaling are developed in chapter 2. Finally, chapter 5 focuses on the design and implementation of dynamically precision-scalable hardware.

2 Speculative and pruned arithmetic circuits

This chapter introduces two architectural techniques for designing approximate arithmetic circuits, speculation and pruning, as well as their combination and extension possibilities.

Section 2.1 introduces the Inexact Speculative Adder (ISA), a novel speculative adder with dual-direction error compensation mechanism. Section 2.2 describes Gate-Level Pruning (GLP), a CAD tool that removes the least-significant logic gates from a circuit in order to reduce energy consumption and silicon area. In section 2.3, ISA and GLP are combined to maximize circuit efficiency. Finally, section 2.4 complements the ISA with a timing-error prediction model, showing its robustness to overclocking.

2.1 Inexact speculative adder (ISA)

Speculative adders [16] exploit the fact that the typical carry propagation chain of an addition does not span the whole length of the adder, making it possible to estimate an intermediate carry using a limited number of previous stages. Thus, the carry propagation chain, which is the critical path of the adder, can be split in two or more shorter paths, relaxing constraints over the entire design, reducing spurious glitching power, and improving the Energy-Delay-Area Product (EDAP) beyond the theoretical bounds of exact adders.

A number of speculative adders have been proposed in literature with different approaches in order to reduce the error rate or magnitude. The ETAII adder [18] consists of regular sub-adder blocks with input carries speculated from Carry Look Ahead (CLA) blocks of the same length. In the ETAIIM version [18], several of the most significant CLA blocks are chained in order to increase accuracy. The ETBA adder [19], direct descendent of the ETAIIM, adds variable speculation signs and sub-adder sum balancing multiplexer blocks to mitigate relative errors. N. Zhu *et al.* [31] and Y. Kim *et al.* [32] have recently demonstrated adders with improved accuracy by considering two prior carry speculation blocks instead of one, coupled with a carry select (ETAIV) [31] or a carry skip [32] technique, with the latter also using sum balancing over several sub-adder blocks.

One room for improvement in the existing speculative adders is that the circuit hardware is not utilized efficiently enough. Indeed, the overhead due to carry speculation is significant and lies entirely in the critical path. On the other hand, the sum balancing blocks, containing parallel multiplexers, take also a part in the critical path, require large fan-outs, but are very weakly exploited due to the low probability of incorrect speculations.

2.1.1 Proposed architecture

The structural diagram of the Inexact Speculative Adder (ISA) [33] adder is depicted in Fig. 2.1. The ISA splits the carry propagation chain in multiple paths executed concurrently. Each path consists of a carry speculator block (SPEC), a sub-adder block (ADD) and an error compensation block (COMP). For each SPEC-ADD-COMP path, the SPEC block generates a partial carry signal that the sub-adder ADD uses as a carry-in to generate a local sum. The COMP block compensates faulty sums either by correcting the local sum or by reducing error magnitude as in [19]. The first speculative path, operating on the LSBs of the adder, does not have SPEC nor COMP blocks since it uses directly the ISA carry-in.

The carry speculation of the SPEC block is generated from a few bits in a carry look-ahead approach sourced by either a static or a dynamic input. The latter, used in ETBA, can evenly distribute the errors at the cost of hardware and delay overhead. When a propagate chain covers the full SPEC block, the exact carry cannot be speculated from the partial product and the output carry is guessed at the input carry value. Long propagate sequences are uncommon in the case of uniform input distribution [18], thus the probability of speculation

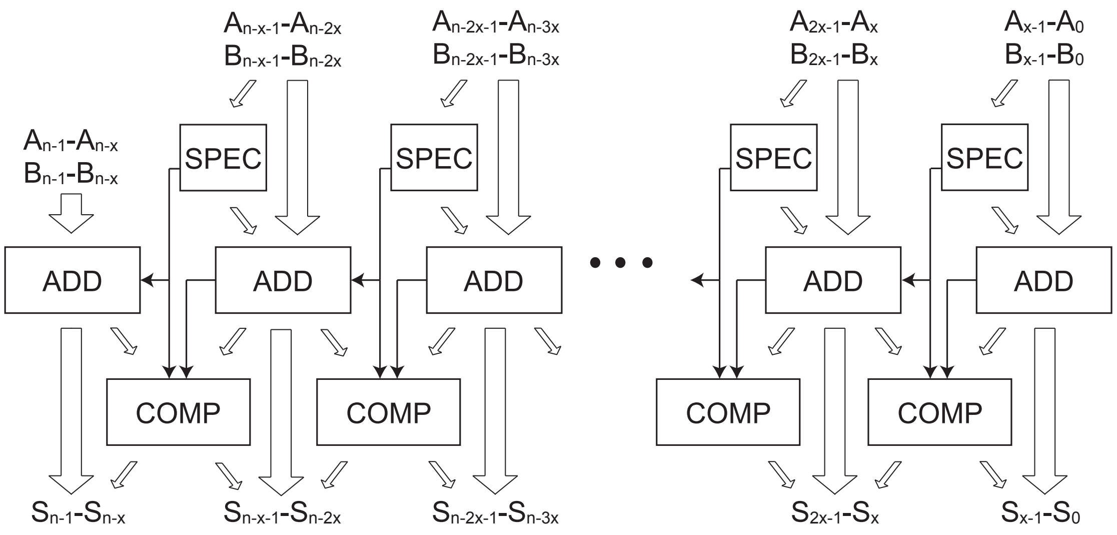


Figure 2.1 – General block diagram of the Inexact Speculative Adder (ISA).

fault decreases by increasing the size of the SPEC block.

The sub-adder ADD calculates block sums locally from the speculated carry of the SPEC block. Without compensation, an internal overflow caused by an inconsistent carry could lead to a massive error. Therefore, the COMP block detects those speculation faults by comparing the carry generated from the SPEC with the carry-out coming from the prior ADD block. The COMP block is implemented in between two local ADD blocks. In case of faulty speculation, it attempts to fully correct a group of LSBs of the local sum. The full correction consists in incrementing or decrementing the group of LSBs, and is only possible when it does not lead to another internal overflow. If correction is not possible, the COMP block can flip a group of the Most Significant Bits (MSBs) of the previous sub-adder sum to minimize the error magnitude.

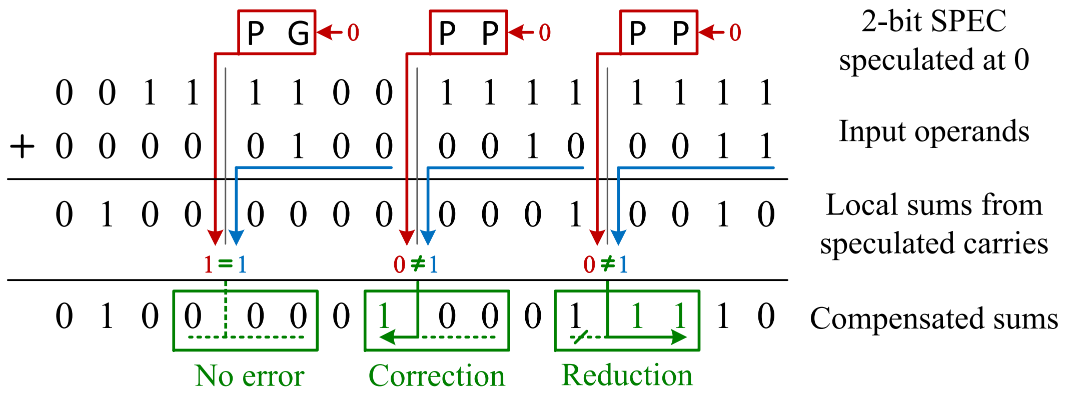


Figure 2.2 – Example of ISA addition arithmetic with 2-bit speculation, 1-bit correction and 2-bit error reduction. Faults only occur in the two right-hand paths. The 1st LSB of the central path can be corrected. The 1st LSB of the right path cannot be corrected, so the MSBs of the preceding sum are flipped to reduce the error value.

The achieved addition arithmetic, illustrated in Fig. 2.2, is a 5-step process:

1. A carry-in is speculated from a very short carry propagation chain for each sub-adder block.

2. The sub-adder calculates the local sum based on this speculated carry-in.

3. Comparison of the speculated carry-in and the prior sub-adder carry-out allows detection of faulty speculation.

4. In case of wrong speculation, correction of the local sum is attempted.

5. If correction is not possible, error magnitude is reduced by balancing the preceding sum bits.

The general block diagram of an Inexact Speculative Adder (ISA) adder is depicted in Fig. 2.1. An ISA splits the carry propagation chain in multiple paths executed concurrently. Each path consists of a carry speculator block (SPEC), a sub-adder block (ADD) and an error compensation block (COMP). For each of these SPEC-ADD-COMP paths, the different blocks have the following functions:

- *SPEC* – The speculator block generates a partial carry signal from a limited number of operand bits in a carry look-ahead approach and sourced by either a static or a dynamic input. When a propagate chain covers the full SPEC block, the exact carry cannot be speculated from the partial product and the output carry is guessed at the input value. As long propagate sequences are uncommon in uniform input distribution [18], the probability of fault decreases when increasing the size of this block.

- *ADD* – The sub-adder block calculates local sums from the speculated carry of the SPEC block.

- *COMP* – Without compensation, an internal overflow caused by an inconsistent carry could lead to a massive error. Therefore, the COMP block detects those speculation faults by comparing the carry generated from the SPEC with the carry-out coming from the prior ADD block. It then compensates faulty sums either by attempting to correct a few bits of the local sum or by reducing relative error over a few bits of the preceding sum.

The first speculative path, operating on the LSBs of the adder, does not have SPEC nor COMP blocks since it uses directly the adder carry-in. The achieved addition arithmetic is illustrated in Fig. 2.2.

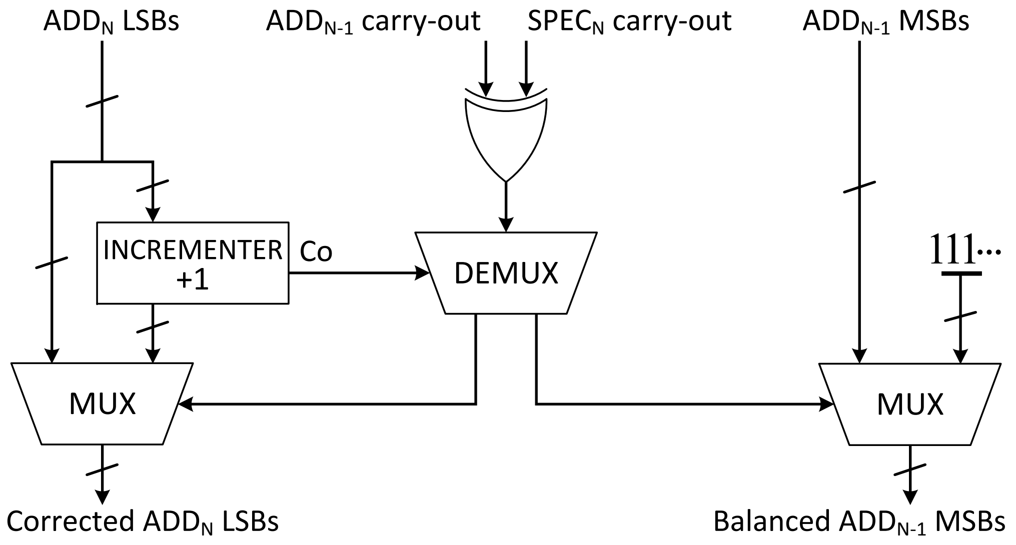


Figure 2.3 – COMP block implementation in the example of a fixed direction of errors leading to positive correction and balancing.

2.1.2 COMP block implementation

Fig. 2.3 presents a possible implementation of the COMP block. The COMP block detects inconsistencies between speculated carry and expected carry from the previous sub-adder with an XOR gate. This creates an error flag that triggers the activation of one of the two compensation techniques, namely error correction and error reduction.

The potential error always remains of the same nature as the input carry of the SPEC block. For example, speculations at 0 instead of 1 can only induce *too low* sums compared to expected ones, while speculations at 1 instead of 0 leads to *too high* sums. Therefore, the sign of the corrective compensation is always recognized. The error correction part of the COMP intercepts a group of LSBs of the local sum, at the position of the error, and performs an unsigned increment or decrement in the direction of this potential error (i.e. *too high* error is solved by a -1 and *too low* by a $+1$). The intercepted bus has a fixed number of bits, thus this operation is only possible if it does not cause an overflow. For instance, if the COMP operates error correction on 3 bits, incrementing 111_2 and decrementing 000_2 is irrelevant and leads to an overflow. The detection of this overflow allows to demultiplex the choice of compensation technique to apply.

When a speculation fault cannot be fixed by the error correction, the COMP balances in the opposite direction of the error a group of MSBs of the preceding sub-adder. This technique allows to attenuate an error at a given bit position by manipulating bits that are less significant with respect to this bit.

Assuming a small correction bus, the corrective increment or decrement operation allows to detect potential overflow and switch to the right compensation technique before the ADD block finish computing. Thus, a significant feature of this adder is that neither the pre-computing of error correction nor the compensation choice lie in the critical path of the ISA adder. The multiplexers are the only components of the COMP block lying in the critical path.

2.1.3 Analysis of error compensation

The objective of this section is to detail the conditions in which an error can be corrected or reduced by the COMP block.

Let a carry error C_{err} occur at the i^{th} bit of an adder. S_i, C_i and P_i denote respectively the sum, carry-in and propagate signals of the i^{th} stage addition. Hence, the sum is defined by:

$$S_i \triangleq P_i \oplus C_i = P_i \oplus C_{err}. \tag{2.1}$$

At the i^{th} stage, the condition under which correction is not possible is that S_i is already opposed to the error and a bit-flip would not compensate the error, meaning that:

$$S_i = \overline{C_{err}} \iff P_i = 1. \tag{2.2}$$

This results in propagating C_{err} to the $(i+1)^{\text{th}}$ stage where the same formulae apply again. As a result, the correction is infeasible only if all the bits of the COMP's correction bus are in propagation mode.

Even though similar to [19] and [32], the effect of the balancing error reduction is more complex in the present work due to the independent sizing between SPEC and COMP blocks. For instance, let an adder leave a non-correctable carry error at the i^{th} bit of the sum, speculated from a s-bit SPEC and followed by a r-bit error reduction scheme. Occurrence of the error presumes that the carry propagation chain prior to the error is longer than s bits, i.e. stages $i-1$ to $i-s$ are all in propagation mode. In addition, those stages are calculated in the previous sub-adder carry chain, which is non-erroneous. Hence, following (2.1), this results in:

$$P_{i-k} = 1 \iff S_{i-k} = \overline{C_{i-k}} = C_{err} \text{ for } k \leq s. \tag{2.3}$$

This means that the sum bits $i-1$ to $i-s$ tend to follow the inverse of the real carry, that being the faulty carry C_{err}. Thus, they can always be flipped to compensate the error. However, (2.3) is only valid for stages known in propagation states. If the COMP's error reduction is larger than the SPEC size ($r>s$), the extra sum bits ($i-s-1$ to $i-r$) do not satisfy this condition, thus, do not follow (2.3). Those bits may not be able to flip the error. This supplemental balancing does not further reduce error magnitude in worst case operations, although it does impact on the overall ISA accuracy in typical situations with shorter carry propagate chains.

2.1.4 The ISA design strategy

The ISA is a general topology of speculative compensated addition wherein the state-of-the-art adders are particular cases. It allows notable improvements concurrently in the circuit performance and accuracy control. This section describes the architectural advantages of the ISA.

Optimized block sizing

The ISA architecture can reduce speculative hardware overhead and improve speed. The state-of-the-art speculative adders always use blocks of the same size in the speculative paths. In the ISA architecture, with the flexibility provided by the advanced compensation scheme, the SPEC block lengths can be traded for longer ADD blocks to fit the same delay requirement. It is then possible to use fewer speculative paths and limit the in-critical path speculation-compensation overhead to a few stages of each path.

Speculation and correction trade-off

The COMP's correction technique resolves carry errors if its intercepted bits are not all propagate signals. In other words, the combined SPEC and COMP's correction techniques prevent errors on carry propagation chains of their cumulated bit lengths. Thus, an easy trade-off can be realized in order to fit and optimize both delay and error rate.

Error reduction and failed correction

All the speculative adders in the literature so far employ only LSBs balancing technique as introduced in [19] to reduce speculative errors. To be efficient in worst-case operation, such technique requires a SPEC block of the same size lying in the critical path. A significant feature of the novel COMP block is that even when the correction technique cannot compensate for the error, the uncorrected bits appear in the same state as they would be for error reduction. Thus, as illustrated in Fig. 2.4, the COMP's correction has the same effect as a shifted balancing scheme and is efficient even in worst-case operation.

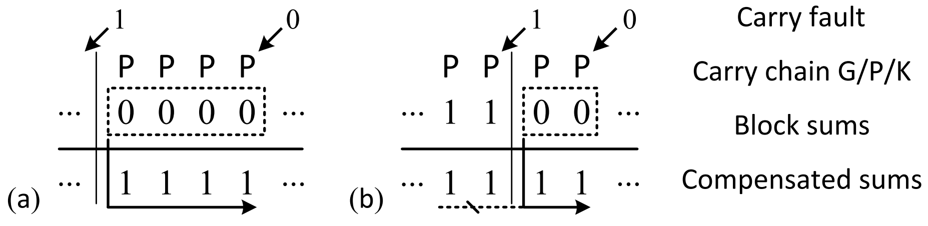


Figure 2.4 – Relative error equivalence between compensation with balancing only (a) and combined with correction (b) in case of non-correctable error.

Design strategy

The ISA offers a general topology of speculative compensated addition inclusive of the state-of-the-art and that allows an optimal balance between circuit and accuracy specifications.

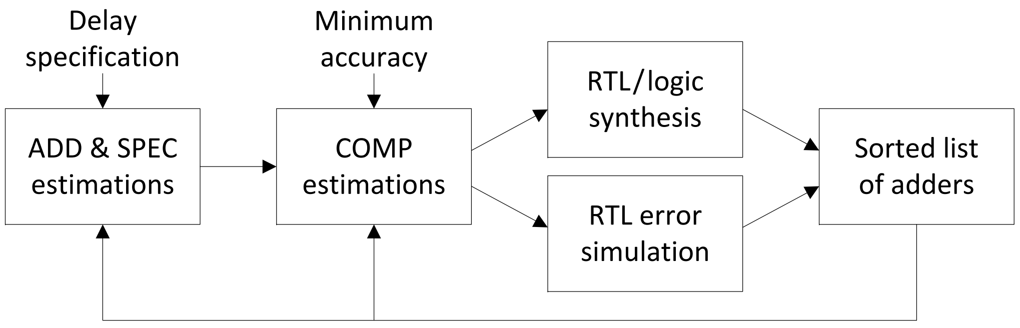


Figure 2.5 – CAD framework for ISA design.

A design methodology through a delay-accuracy approach is presented in Fig. 2.5. The adequate delay trade-off is mainly obtained by sizing SPEC and ADD blocks, principal slack elements of the ISA. Then, the COMP's error correction and error reduction techniques enable to tune and fit the accuracy requirements at the cost of hardware overhead and with a minimum delay penalty for multiplexing the result on a few compensated bits.

Adders in literature describe particular cases of implementation excessively considering either performances or errors. In the ISA architecture, the speculation overhead can be traded for longer sub-adders while fitting the same delay requirement. It is then possible to use fewer speculative paths and limit the in-critical path speculation-compensation overhead to a few bits of each path while fitting the accuracy requirement. This approach allows notable improvements in circuit performances [33].

2.1.5 Results and comparison

Methodology

In this work, 32-bit speculative adders with uniformly-sized blocks, i.e. parallel paths of 2×16, 4×8, 8×4 and 16×2 bits, have been considered. All circuits are synthesized in a 65 nm commercial technology library, and delay, area and power are estimated using Synopsys Design Compiler.

The metrics used to characterize approximate adders in this work are based on the relative error (*RE*), defined as:

$$RE = \left| \frac{S_{\text{exact}} - S_{\text{approx}}}{S_{\text{exact}}} \right| \tag{2.4}$$

where S_{approx} and S_{exact} are the approximate and correct sums of an addition, respectively. Interesting for many applications, particularly in media processing, the main metric considered is the RMS of the relative error (RE_{RMS}) that should be minimized. The worst-case accuracy (RE_{MAX}), i.e. the largest relative error of an adder, is also taken into account.

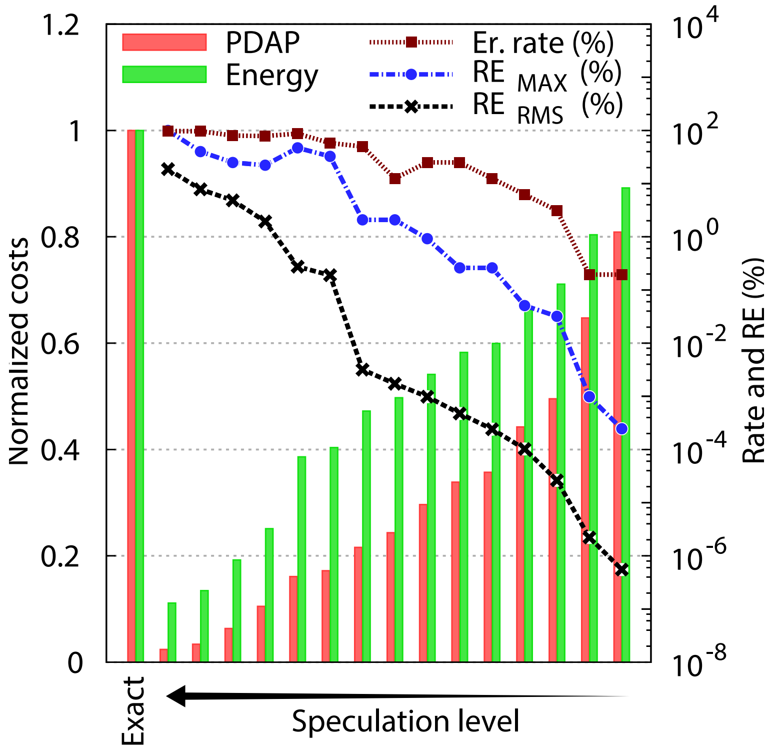
Most of the inexact adders are validated and characterized through the simulation of random sets of inputs since exhaustive simulation of large bit-width arithmetic blocks are too time-consuming. As a matter of fact, the metrics used are statistical estimators, d-ependent on the random input distribution and the chosen sample itself (occurrence of specific patterns initiates errors in specific adders).

In this work, 32-bit adders are compared using two samples of five million unsigned random inputs. First, a formal uniform distribution is used to estimate RE_{RMS}. Then, a logarithmic distribution exhibiting a very large dynamic range of scattered values is used to detect the worst-case accuracy RE_{MAX}.

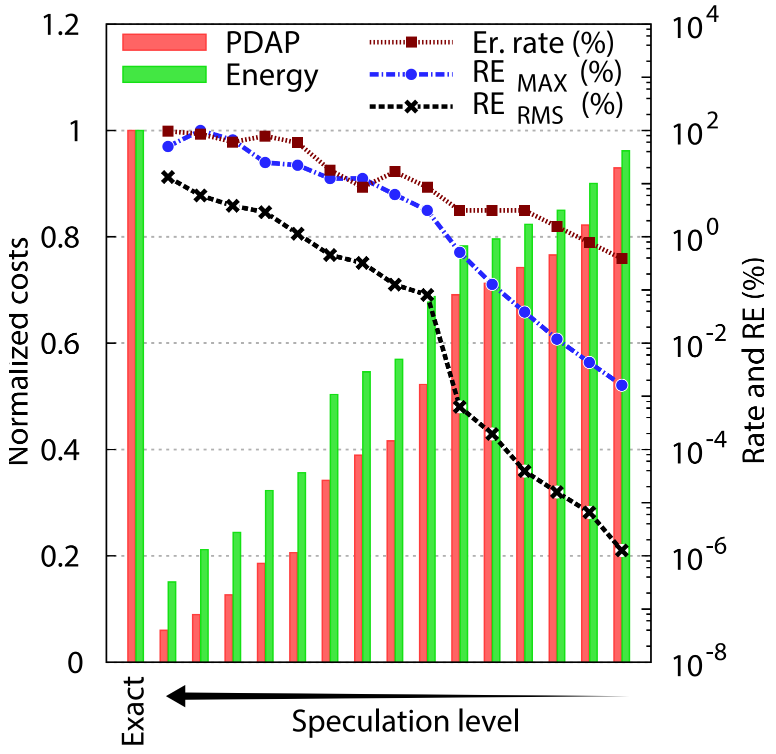
Results

Fig. 2.6a and 2.6b [34] show the range of circuit costs and error characteristics achievable by ISA adders constrained at 3.3 and 1.6 GHz, respectively. Each couple of bars shows the Power-Delay-Area Product (PDAP) and energy consumption at constrained speed, normalized to the exact adder represented on the left. Error characteristic lineplots for RE_{RMS} and RE_{MAX} are shown in percentage on the right logarithmic scale.

It is of interest to highlight the diversity of error engineering possibilities permitted by the ISA topology. For instance, for 3.3 GHz designs, tolerating 1 % of relative error RMS in the calculations allows 65 % power reduction and 88 % PDAP savings compared to an exact adder of same speed. If the desired application is more sensitive to error, tolerating only 0.001 % of error RMS could still allows 50 % power reduction and 73 % PDAP savings.



(a) Speculative adders at 3.3 GHz



(b) Speculative adders at 1.6 GHz

Figure 2.6 – Error characteristics and normalized cost of 32-bit ISA adders.

Timing constraint has a significant influence on the result obtained with the speculation. As visible on the two figures, gains tend to reduce, particularly for high-accuracy adders (on the right of the figures). This is due to the fact that at lower speed, the exact adder has a more sequential architecture, extremely energy and area efficient. The overhead required for high-accuracy ISA quickly outweighs the benefits of the shorter critical path.

On low-accuracy adders, RE_{RMS} and RE_{MAX} have similar trends. But this does not follow for higher accuracies as RE_{MAX} becomes complex and expensive to control. Thus, a gap appears between RE_{MAX} and RE_{RMS} when the constraints on the circuit become too high, at 1 % for 3.3 GHz and 10^{-3} % at 1.6 GHz.

Comparative study

Fig. 2.7a and 2.7b [33] show the power, EDAP requirements and error characteristics of exact, ETAII and ETBA adder implementations synthesized under 3.3 and 5 GHz constraints. For fair comparison, as the ISA offers the largest design space, this latter has been fitted to typical ETAII and ETBA circuits.

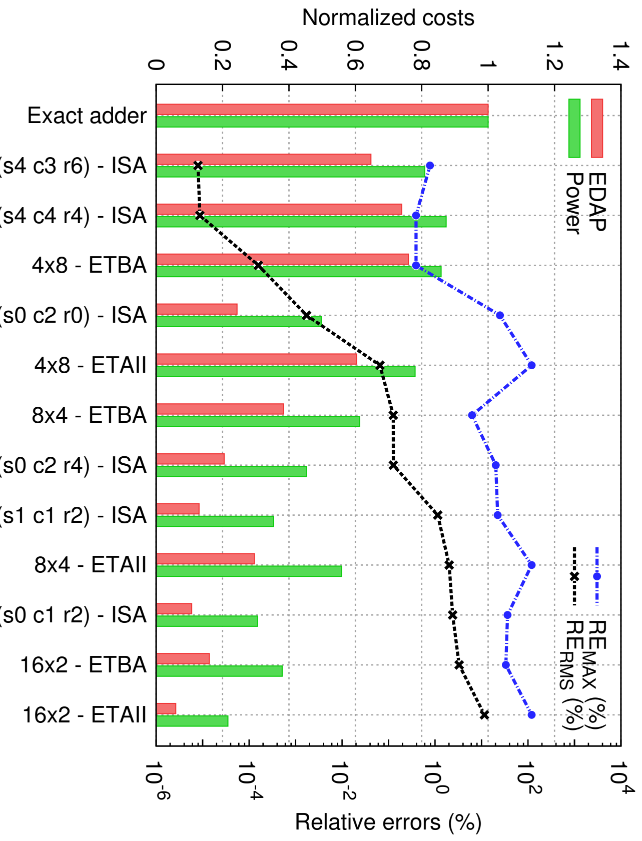
It is particularly clear that some ISA configurations achieve significant reduction of power consumption and EDAP at equivalent accuracy than the ETBA and ETAII. At identical RE_{RMS} and nearby RE_{MAX}, ISA circuits achieve between 11 % and 26 % of power reduction and between 24 % and 60 % of EDAP reduction upon ETBA.

The ISA architecture allows to tune and match any error specification. For instance, increasing SPEC and error reduction lengths minimizes progressively RE_{RMS} as depicted by the 8×4 adders (ISA and ETBA) in the middle of Fig. 2.7b. However, the increasing in-critical path overhead leads to a breakdown in circuit efficiency to fit in the delay constraint.

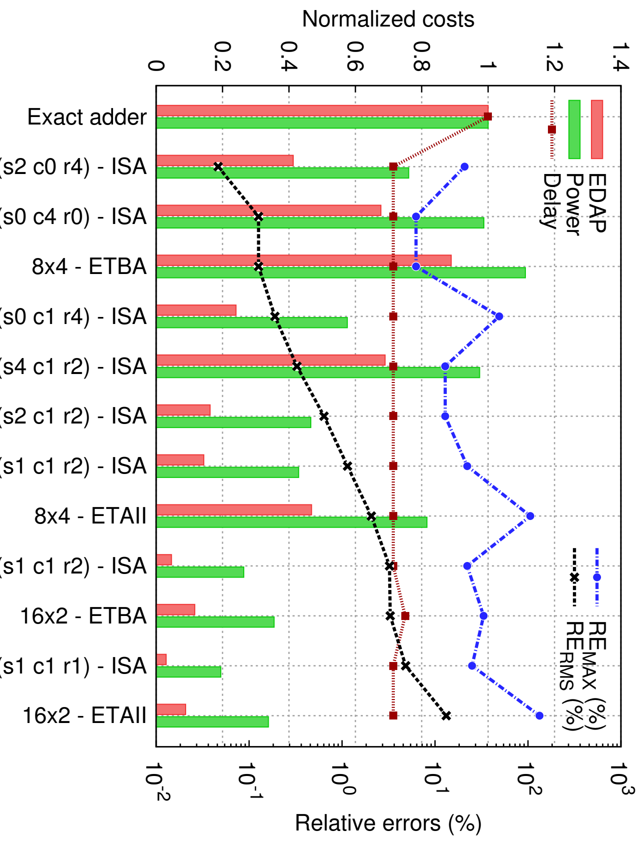
Shifting the speculative hardware towards error correction can limit the resource cost while continuing to increase accuracy (as with the 8×4 [s0 c4 r0] ISA on the left of Fig. 2.7b). The ETAII, one of the most EDAP and energy efficient existing adders [32], is generally outperformed by ISA adders with short SPEC lengths.

On the other hand, reducing the in-critical path speculative hardware relaxes the design and the adder structure can collapse into less speculative paths, offering good accuracy with significant gains in power and EDAP, as some blocks are removed and energy or delay budget can be reallocated to increase adder performance. In Fig. 2.7b, 4×8 ISA are sometimes twice as EDAP efficient for higher accuracy as the above-mentioned 8×4 ISA. Such structural contractions lead to large reductions of RE_{RMS} at relatively low costs as observable in Fig. 2.7a.

Controlling RE_{MAX} requires an expensive combination of SPEC and COMP blocks. Relaxing a bit this requirement leads to significant savings in all aspects as observable on the left-hand sides of Fig. 2.7a and 2.7b. However, weakly compensated implementations such as ETAII



(a) Speculative adders at 3.3 GHz.



(b) Speculative adders at 5 GHz.

Figure 2.7 – Comparison of ISA with state-of-the-art adders. Specific ISA implementations are denoted by number and size of sub-adders, SPEC size *s* and COMP's error correction-reduction lengths *c* and *r*. Delays are shown in (b) as the fastest exact adder cannot fit the 5 GHz constraint and 16x2 ETBA suffers from a drop of efficiency exactly at synthesized speed (q.v. section 5.4), it is replaced with a lower speed for fair comparison.

and 4×8 [s0 c1 r4] ISA may be limited in practical use due to the occurrence of high relative errors. The variety of results and large variations of the chosen error characteristics point to the importance of also developing robust error metrics to match applications and circuits.

Thanks to its new speculative path and compensation scheme, the ISA architecture proposed herein greatly improves hardware efficiency upon the state-of-the-art and introduces a new way to control errors.

2.2 Gate-level pruning (GLP)

The concept of *pruning* has been introduced with *Probabilistic Logic Minimization* [23], a CAD technique where bit-flips are introduced in Karnaugh maps to simplify logic functions, and *Probabilistic Pruning* where full-adder cells are pruned out of adders. An important limitation of all these techniques is that the amount of inaccuracy is set at design time and cannot be changed.

2.2.1 Proposed framework

Gate-Level Pruning [35, 36] is a CAD technique to automatically generate inexact circuits starting from a conventional design by adding only one small step in the digital design flow. The CAD framework is presented in Fig. 2.8.

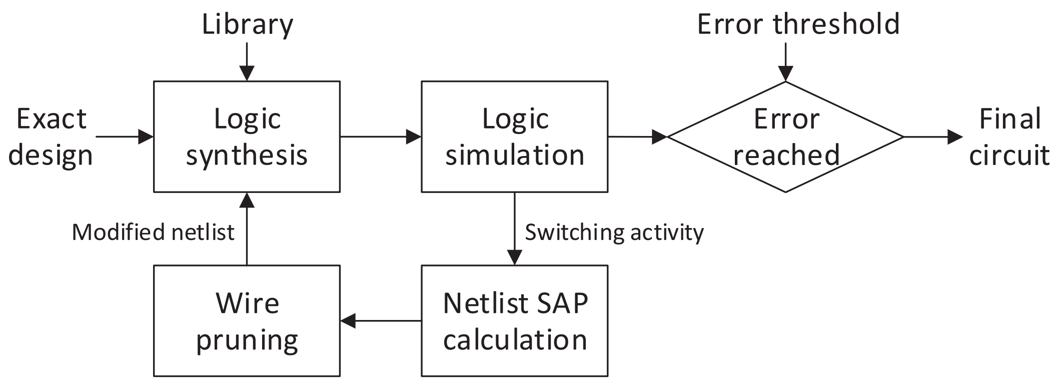


Figure 2.8 – CAD framework for Gate-Level Pruning.

Any exact circuit can be represented by a directed acyclic graph as depicted in Fig. 2.9, where the nodes are components such as gates, and whose edges are wires. The decision to prune a node is based on two criteria: the significance, which is a structural parameter, and the activity or toggle count. The nodes with the lowest Significance-Activity Product (SAP) are pruned first. By doing so, the error magnitude grows with the amount of pruning. Alternatively, depending on the application's requirement, the designer may choose to prune nodes according to the activity only, in order to minimize the error rate.

The activity of each wire is extracted from the SAIF file (Switching Activity Interchange Format) obtained through gate-level hardware simulations. This file contains the toggle count of each wire, as well as the time spent at the logic levels 0 and 1 respectively. In order to get an accurate activity estimation, the system should be simulated with an input stimulus representative of the *real operation* of the circuit. The more the simulation is realistic, the more the toggle count is accurate and leads to an efficient pruning.

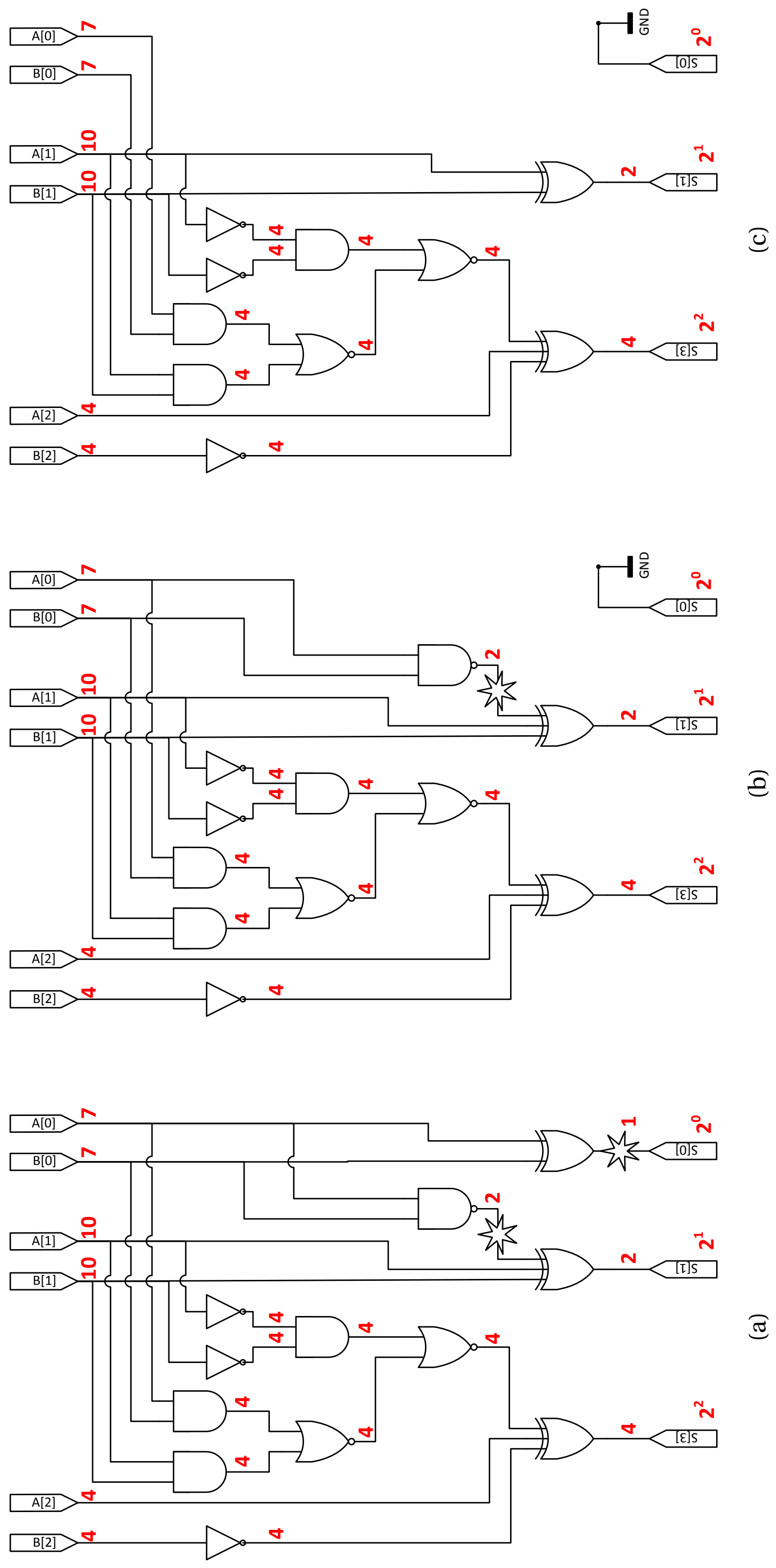


Figure 2.9 – Example of GLP flow with a 3-bit adder. The net significance is written in red and the stars indicate the nets that are pruned first. The original netlist (a) is pruned of one node (b) and then two nodes (c).

The significance of each primary output is set by the designer depending on the application's requirement. Pruning is applied on several arithmetic circuits where each primary output is weighted by a power of two. It is therefore worth applying a weighted significance attribution, where each bit position has a significance two times higher than the previous when moving from the LSB to the MSB. Reverse topological graph traversal is then performed to compute each nodes' significances as follows:

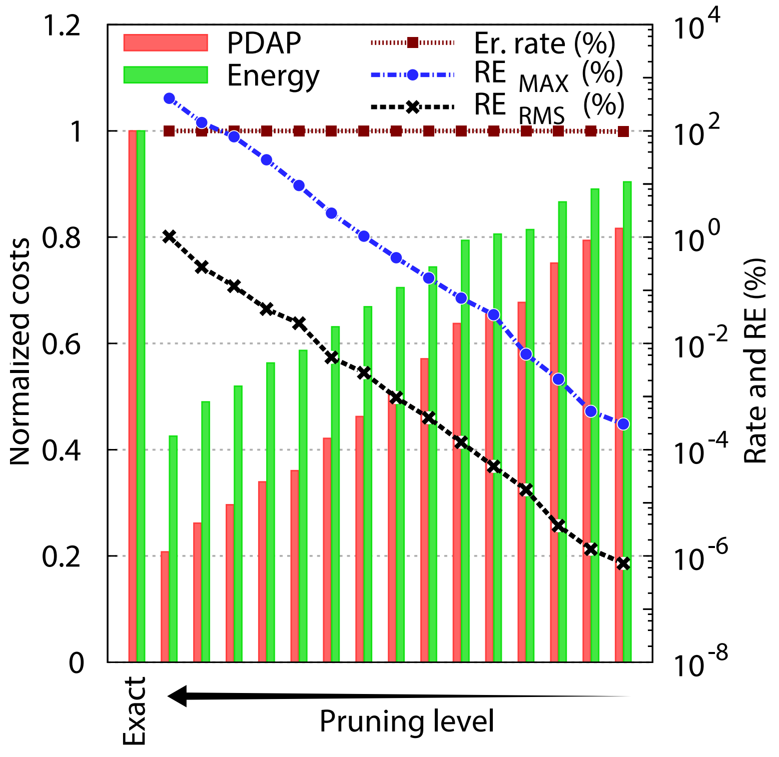
$$\sigma_i = \sum \sigma_{desc(i)} \tag{2.5}$$

where σ_i is the significance of the node i and $\sigma_{desc(i)}$ is the significance of the direct descendants of node i. An example of weighted significance attribution is shown in Fig. 2.9.

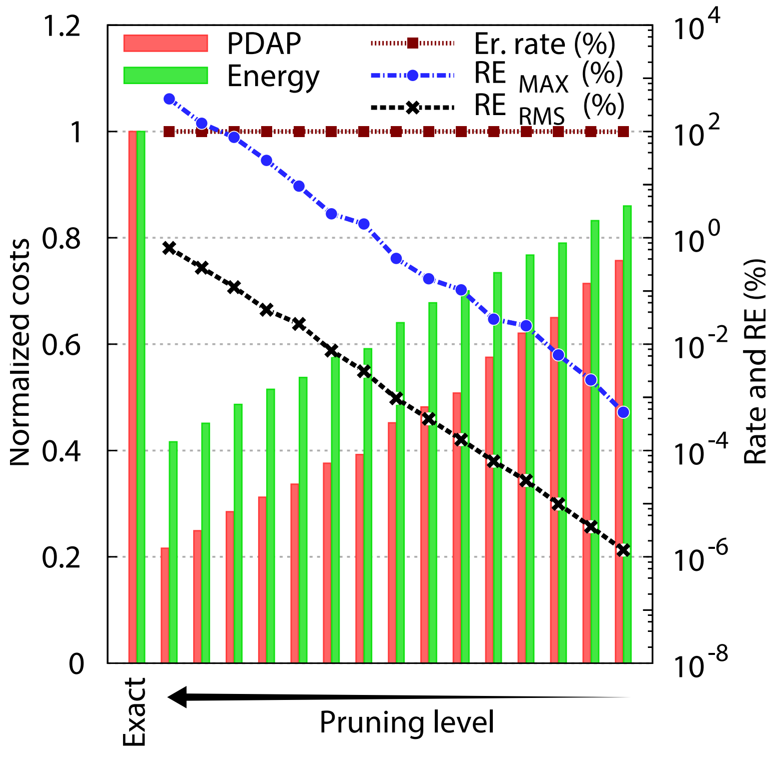
Once the significance and activity is determined, the nodes, i.e. gates and their corresponding wires, are ranked according to their SAP. The ones with the lowest SAP are disconnected from the verilog netlist, and an incremental re-synthesis is performed in order to remove or replace the unconnected gates.

2.2.2 Results

Fig. 2.10a and 2.10b [34] show the range of circuit costs and error characteristics achievable by pruned adders constrained at 3.3 and 1.6 GHz, respectively. Pruning and speculation bring the same order of magnitude of savings. For both speculation and pruning, the RE_{RMS} and the RE_{MAX} grow with an exponentially trend versus circuit savings. But the two techniques clearly have a different impact on the output quality: the error rate of pruned adders rapidly reaches 100 %, the reason being that in the first steps of the pruning process, some of the least significant outputs are removed. On the other hand, in speculative adders, a small speculation-correction overhead leads to a decrease of the error rate despite lower circuit efficiency.



(a) Pruned adders at 3.3 GHz



(b) Pruned adders at 1.6 GHz

Figure 2.10 – Error characteristics and normalized cost of 32-bit pruned adders.

2.3 Combination of speculation and pruning

2.3.1 Proposed method

Pruning and speculation produce errors of different nature. While ISAs tend to produce rare errors with a low relative magnitude which can appear on significant bits if the carry chain is cut at this position, GLP leads to high error rates with errors generally occurring on the least significant bit positions. For these reasons, it is worth trying to combine ISA and GLP to get even more circuit savings for similar error levels [37].

The general methodology to combine speculation and pruning, making *pruned speculative* circuits, is:

1. Synthesize speculative circuits at the desired speed constraint starting from the HDL description.
2. Apply the pruning methodology to the gate-level netlist of the speculative circuits.

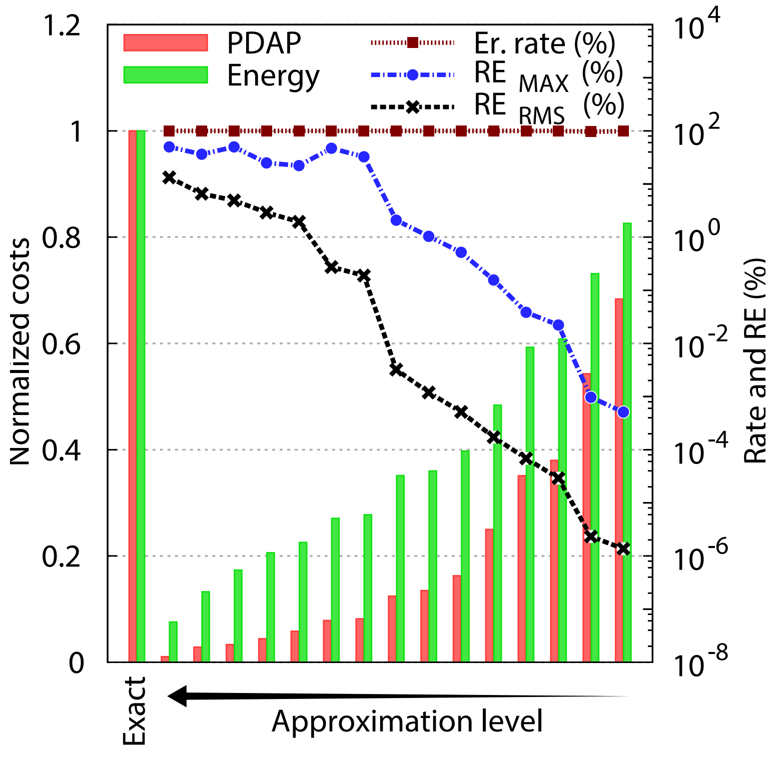
2.3.2 Results and comparison

Results

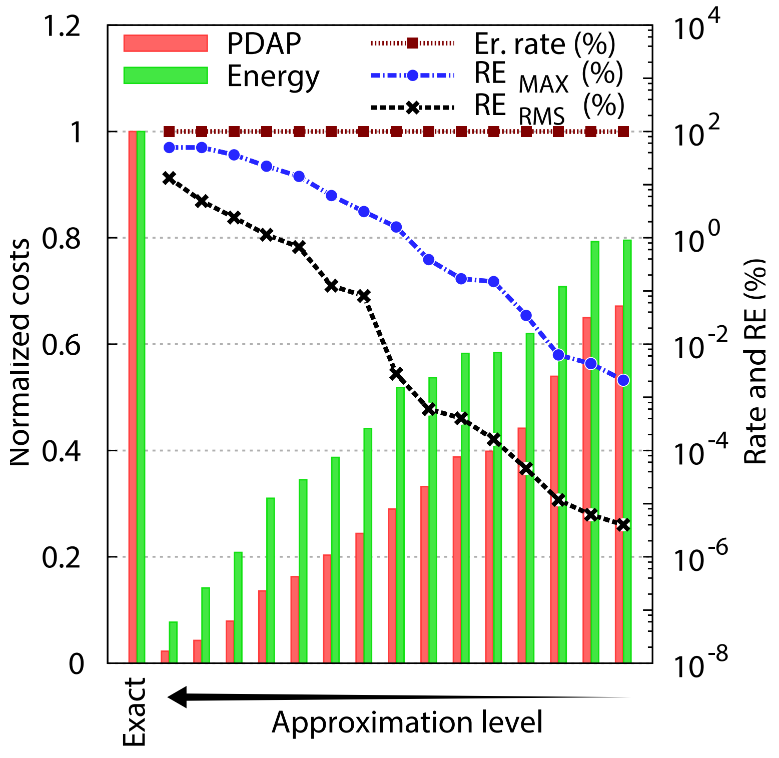
Fig. 2.11a and Fig. 2.11b show the normalized costs as well as the error characteristics of mixed 32-bit adders synthesized at 3.3 GHz and 1.6 GHz, respectively.

It is very interesting to see that the RE_{RMS} and the RE_{MAX} follow almost the same trend for the mixed adders than for the speculative adders. This is due to the fact that the two techniques produce different types of errors: in speculative adders which are cut into sub-blocks, errors are rare but can occur as often on LSBs than on MSBs, the latter having a significant impact on the error magnitude.

In opposition, the weighted significance attribution of the pruning methodology leads to a large number of errors, and thus an error rate 100 % for most of the pruned adders, but those are limited to the LSBs and have a small impact on the error magnitude. Hence, the assumption can be made that the errors resulting from pruning and speculation are uncorrelated, and thus, equalizing the error levels resulting from each technique enables additive area, power and delay savings. This is verified by simulation, all the *pruned speculative* adders depicted Fig. 2.11a and 2.11b have a lower PDAP cost and consume less energy than the pruned or the speculative adders.



(a) Pruned adders at 3.3 GHz



(b) Pruned adders at 1.6 GHz

Figure 2.11 – Error characteristics and normalized cost of 32-bit mixed adders.

Comparative study

Table 2.1 summarizes the PDAP reduction obtained through the use of the three techniques, namely pruning, speculation and the mixing of both, taking comparable designs from Figs. 2.6, 2.10 and 2.11. It is shown that a *pruned speculative* adder with only 2 % relative error magnitude has a normalized PDAP of 0.05, which is a factor 20 improvement. Moreover, energy consumption is reduced by a factor 4 compared to the exact 32 bit adder synthesized at the same speed.

Table 2.1 – Comparison of the three inexact design techniques.

Specifications		PDAP normalized to exact adder		
Frequency	RE_{RMS}	Pruning	Mixed	Speculative
3.3 GHz	$3 \, . \, 10^{-6}$ %	0.76	**0.54**	0.64
	10^{-3} %	0.52	**0.14**	0.3
	2 %	0.2	**0.05**	0.1
1.6 GHz	10^{-5} %	0.64	**0.54**	0.78
	10^{-1} %	0.3	**0.2**	0.43
	1 %	0.2	**0.14**	0.2

2.4 Overclocking of speculative adders

Cloud computing requires to process a massive amount of data, speeding up this process by overclocking the circuits would lead to significant energy gains. On the other side, variability is enormous for IoT devices, and designing for worst case is terrible for energy efficiency. In both context, timing errors could be catastrophic. In a desperate attempt to protect circuits from these errors, designers typically apply ultra-conservative—thus ultra-expensive—guardbands to avoid timing errors. In this chapter, we combine speculative circuits designed for critical-path optimization and overclocking to reduce conservative timing guardbands and enable operation under overclocking.

A supervised learning model is applied to overclocked speculative adders to predict timing errors at bit level [8]. Errors from speculative architectures and overclocking are combined in order to minimize the overall error compared to what an exact and properly-clocked circuit would output. This combined approach targets at optimizing both structural errors and timing errors to insure good-enough arithmetic computations. Those two approaches targeting different abstraction levels and introducing different types of errors in digital circuits could intuitively be the perfect combination to maximize circuit efficiency. As in analog circuits, in which contributions from different noise sources are fine-tuned to maximize SNR while minimizing power consumption

2.4.1 Proposed method for timing-error prediction

As proposed in [38], the bit-level timing-error prediction model for circuit overclocking uses a binary classification method to predict timing errors for a given clock reduction and input load. It captures the dynamic circuit path sensitization behaviors by learning the mapping relationship between input workload and bit-level timing errors. For each bit position, a binary classifier is trained to predict if it is timing-erroneous. The overall model construction flow, containing two parts, *Data Collection* and *Model Training*, is illustrated in Fig. 2.12.

During the *data collection* phase, the circuit RTL code is first synthesized into a gate-level netlist and its corresponding Standard Delay Format (SDF) information. Second, using random stimuli, a gate-level SDF-annotated simulation is done at a desired clock speed. For each output bit, we determine the timing class as either *correct* or *erroneous.*

In a *model training* phase, the useful feature vectors from input stimuli are extracted. Both current and preceding input stimuli are used. Besides, the current and preceding output bit values are also considered as input features since bit flips can only occur when these two are different [39]. If these two bits hold the same value, the latched value is correct to users even if the clock period does not meet the sensitized path delay.

A binary classifier is trained for each bit position using a Random Forest tree Classification (RFC). RFC is a method composed of a number of decision trees, which learn a set of decision

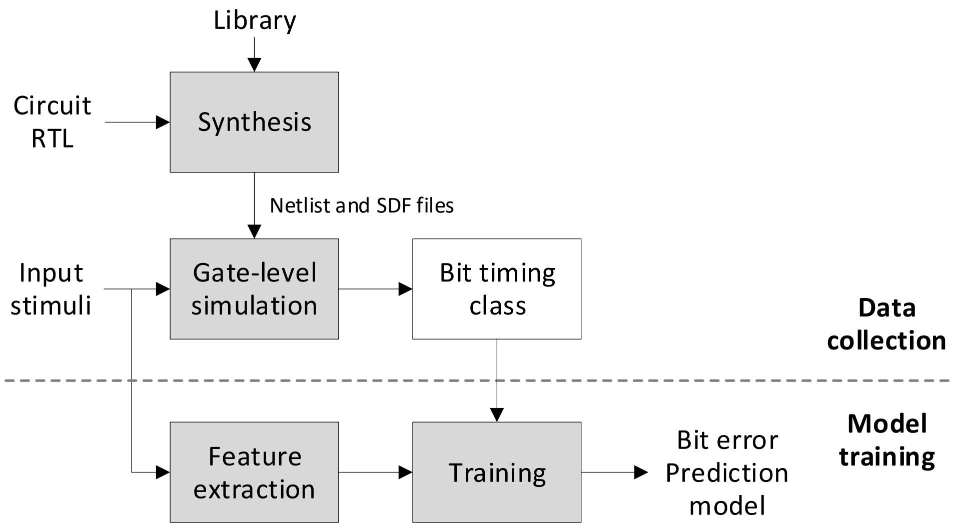


Figure 2.12 – Bit-level timing error prediction model construction flow.

rules based on the pattern of input and their possible outcomes. It considers the joints effects of different bit positions but could incur overfitting problem. RFC alleviates overfitting issue by developing more than one decision tree and use their average result as final prediction. It may lose the opportunity to learn some “irregular” patterns, overall it reduces the overfitting and boosts performance.

2.4.2 Combining structural and timing errors

All previous works have discussed individual use of either approximate circuit design, such as speculative compensated architectures, or guardband-reduction (overclocking) timing-error prediction. But those two approaches targeting different abstraction levels could intuitively be the perfect and complementary combination to maximize circuit performances and robustness. Indeed, timing errors occur on the critical paths, which would be split into multiple shorter paths in a speculative circuit. The timing errors would thus be distributed among all outputs—instead of only degrading MSBs in conventional circuits—but at the cost of structural errors due to speculation. Parameters of the speculative structure and levels of guardband reduction can be adjusted together in order to find an optimum between timing and structural errors.

A new error model needs to be developed to distinguish and combine correctly the error contributions from both abstraction layers. First, at behavioral level, *structural errors* are caused by the design of the ISA architecture. Those deterministic errors vary with the selection of design parameters such as the selection of speculation, error-correction and error-reduction mechanisms. Structural errors are obtained by comparing the outputs from the designed circuit from exact addition results. Then, at gate level or below, *timing errors* occur when overclocking the approximate circuit, thus are obtained by comparing the over-clocked circuit to the same inexact but properly-clocked circuit. Those errors vary with different clock periods and are less predictable as they also depend from the previous circuit state or inputs.

To simplify the three type of output values used to compute those errors, we define the following types of output values:

- y_{silver}, the *silver output* obtained from the over-clocked approximate circuit, containing both structural and timing errors.
- y_{gold}, the *golden output* the expected value from the implemented circuit, containing the structural errors only.
- y_{diamond}, the *diamond output* ideal output value from an exact addition or conventional adder circuit.

Thus, we compute the *arithmetic error* (E) from each abstraction level as:

$$E_{\text{struct}} = y_{\text{gold}} - y_{\text{diamond}} \quad E_{\text{timing}} = y_{\text{silver}} - y_{\text{gold}}\,, \tag{2.6}$$

whereas the *relative error* (RE), both contributions being calculated with respect to the exact result, is defined as:

$$RE_{\text{struct}} = \frac{y_{\text{gold}} - y_{\text{diamond}}}{y_{\text{diamond}}} \quad RE_{\text{timing}} = \frac{y_{\text{silver}} - y_{\text{gold}}}{y_{\text{diamond}}}\,. \tag{2.7}$$

Despite this study only considers unsigned computations, it is important for arithmetic and relative errors to be kept signed. Indeed, if both error contributions are in the same directions, they would add to each other to increase the overall error, such as in Fig. 2.13:

output values			error contributions	
y_{diamond}	1000	8	RE_{struct}	$\frac{6-8}{8} = -\frac{2}{8}$
y_{gold}	0110	6	RE_{timing}	$\frac{4-6}{8} = -\frac{2}{8}$
y_{silver}	0010	4	RE_{joint}	$-\frac{2}{8} - \frac{2}{8} = -\frac{4}{8}$

Figure 2.13 – Example of additive errors.

But if two errors happening simultaneously are in opposite directions, they could compensate each other and reduce the overall error, such as in Fig. 2.14:

output values			error contributions	
y_{diamond}	1000	8	RE_{struct}	$\frac{6-8}{8} = -\frac{2}{8}$
y_{gold}	0110	6	RE_{timing}	$\frac{7-6}{8} = +\frac{1}{8}$
y_{silver}	0111	7	RE_{joint}	$-\frac{2}{8} + \frac{1}{8} = -\frac{1}{8}$

Figure 2.14 – Example of compensating errors.

2.4.3 Experimental study

This study focuses on the case of binary addition based on the use of ISA adders synthesized for 3.3 GHz in a 65 nm technology. Several ISA adders have been selected and implemented with design parameters optimizing error and circuit costs. Timing-error prediction has been adapted to predict timing errors on these circuits for different overclocking levels. Fig. 2.15 depicts the flow used to combine ISA errors with timing errors.

```
inputs: set of ISA architectures, input set, clock periods
outputs: mean arithmetic errors
foreach $ISA \in$ ISA architectures do
    foreach $x \in$ input vectors do
        compute $y_{\text{diamond}}[x]$
        compute $y_{\text{gold}}[x, ISA]$
        compute $E_{\text{struct}}[x, ISA] = y_{\text{gold}}[x, ISA] - y_{\text{diamond}}[x]$
        foreach $clk \in$ clock periods do
            compute $y_{\text{silver}}[x, ISA, clk]$
            compute $E_{\text{timing}}[x, ISA, clk] = y_{\text{silver}}[x, ISA, clk] - y_{\text{gold}}[x, ISA]$
            compute $E_{\text{joint}}[x, ISA, clk] = E_{\text{timing}}[x, ISA, clk] + E_{\text{struct}}[x, ISA]$
    compute means of $|E_{\text{joint}}[x, ISA, clk]|$ over inputs
```

Figure 2.15 – Pseudo-code computing the mean arithmetic error of over-clocked ISAs with structural and timing errors.

Twelve different ISA designs have been selected from [34], they are the best implementations fitting the 0.3 ns timing constraints. All ISA have regular structures with uniformly sized blocks (i.e. parallel paths of 2x16, 4x8, 8x4 bits only) and are denoted by quadruples of bit-widths: (block size, SPEC size, correction, reduction). They have been confronted to an exact adder, also constrained at 0.3 ns.

Approximate circuits are commonly characterized and validated through the simulation of random sets of inputs. As a matter of fact, the presented results are statistical estimations depending on the random sample distribution (occurrence of specific patterns initiates errors in specific adders). In this work, adders are characterized using a sample of ten million unsigned random inputs. The main metric considered is the Root Mean Square (RMS) of the relative error *RE* which is independent of the adder bit-width and proportional to the SNR, which it interesting for many applications, particularly in multimedia processing.

Circuits have been synthesized with Synopsys Design Compiler in an industrial 65 nm technology from high-level descriptions in order to benefit from the compiler's optimization libraries and most favorable architecture choices.

2.4.4 Results

Timing-error model evaluation

To evalutate the model prediction accuracy for a selected overcloking rate, the *Average Bit-level Prediction Error Rate* (ABPER) is used and defined as follows:

$$\text{ABPER}[clk] = \frac{\sum_{bit\ n}\left(\frac{\sum_{cycle\ t} |TC^{(pred)}_{clk,n,t} - TC^{(real)}_{clk,n,t}|}{||\#cycles||}\right)}{||\#bit\_positions||} \tag{2.8}$$

where $TC^{(pred)}_{clk,n,t}$ and $TC^{(real)}_{clk,n,t}$ are the predicted and real timing classes, 0 for timing-erroneous and 1 for timing-correct (for a given clock period clk, bit position n and cycle t).

Although the ABPER is a good metric for bit-level prediction accuracy, the metrics for approximate circuit characterization are usually based on the arithmetic error value of the output. To make ABPER compatible with arithmetic error, the *Average Value-level Predictive Error* (AVPE) is used and defined as:

$$\text{AVPE}[clk] = \frac{\sum_{cycle\ t} \frac{|\, y^{(pred)}_{\text{silver}}[clk,t] - y^{(real)}_{\text{silver}}[clk,t]\,|}{y^{(real)}_{\text{silver}}[clk,t]}}{||\#cycles||} \tag{2.9}$$

where $y^{(pred)}_{\text{silver}}$ and $y^{(real)}_{\text{silver}}$ are the predicted and real arithmetic output values at a given clock period clk and at cycle t.

Note that the model does not directly generate arithmetic values, it only generates timing-class vectors, which are arrays of bit-flip positions, and deduces the corresponding y_{silver} compared to the expected output y_{gold}.

Fig. 2.16 presents the ABPER for each ISA at three overclocking ratios: 5, 10 and 15 %. From this figure, it is first observable that almost all ABPER values are around or less than 1 %, demonstrating a high prediction accuracy of the model. Second, ABPER at higher overclocking is always larger than that at less overclocking. For example, the third ISA (8,0,0,4), has ABPER around 0.1 % at 0.285 ns (5 % overclocking), and has ABPER around 1 % for 10 % and 15 % overclocked frequencies. This is because more paths violating timing specification resulting in more timing errors, which makes model harder to track all path sensitization behaviors. Some ABPER can reach 0 if there is no timing error, such as ISA (8,0,0,0) at 0.285 ns and 0.27 ns (scale is limited to 10^{-6}, whether the ABPER is lower or zero).

Fig. 2.17 presents the AVPE for each ISA at three overclocking ratios. This figure shows that

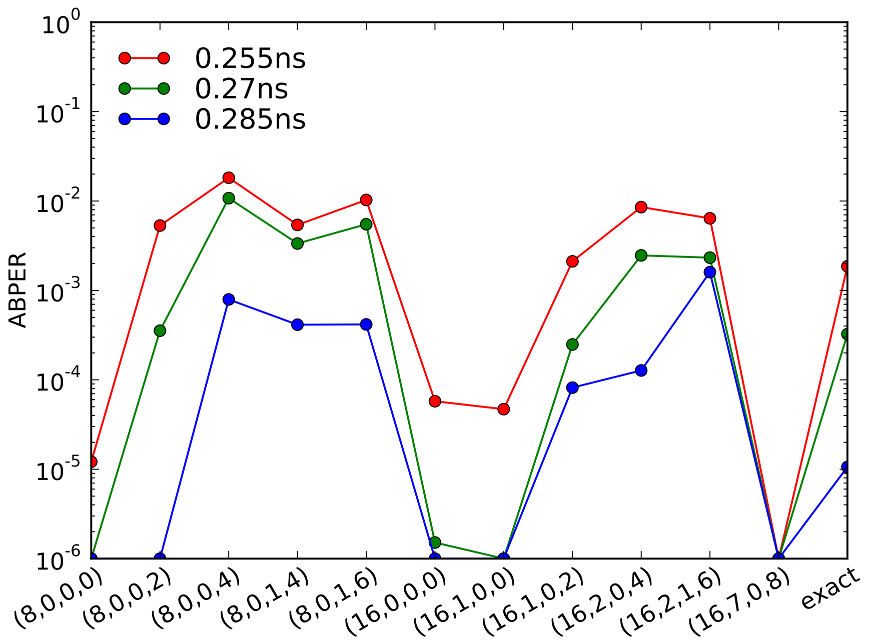


Figure 2.16 – Average bit-level prediction error rate (ABPER) under three overclocking ratios.

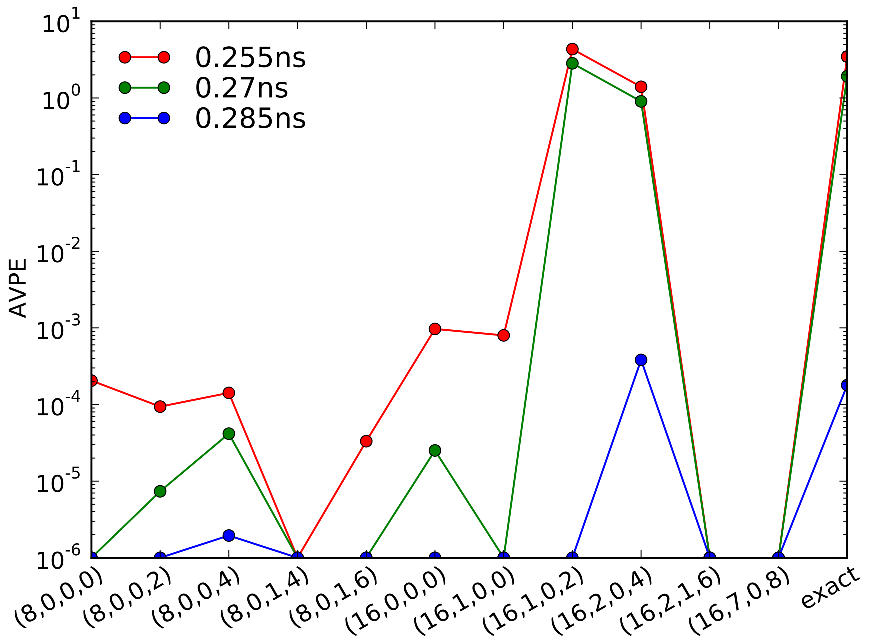


Figure 2.17 – Average value-level predictive error (AVPE) under three overclocking ratios.

although bit-level prediction accuracy are always good but the mispredicted bits could sometimes cause a large arithmetic error. For example, the eighth ISA (16,1,0,2) at 0.255 ns and 0.27 ns causes a AVPE around 5. This is because many mispredicted bits are among most significant bits that can cause a large deviation up to 2^{32} from original value. While for most

ISA, the third ISA (8,0,0,4) for example, has AVPE less than 0.1 % for all three clock frequencies, showing the misprediction effect on arithmetic value is negligible (scale is limited to 10^{-6}, whether the AVPE is lower or zero). Overall, most AVPE are lower than 10^{-2}, indicating that misprediction on arithmetic error is tolerable for most ISA designs.

Results of error combination

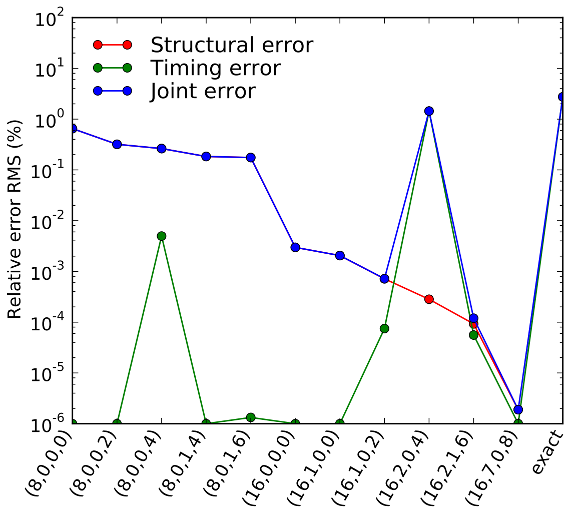
Fig. 2.18 shows the structural and timing relative error RMS as well as their resulting joint contribution for ISA designs under the three overclocked frequencies.

At the lowest overclocking rate of 5 % (Fig. 2.18a), the exact adder circuit (rightmost of the figure) shows to be subject to large timing errors which make it the worst adder of the group in terms of overall joint error RMS. For most ISA adders, the joint error is visibly dominated by the structural-error contribution coming from the speculative architecture. Low and medium-accuracy ISA circuits (on the left part of the figure) seem very robust to timing errors, having negligible timing errors compared to structural errors. Among the high-accuracy ISA designs, only ISA (16,2,0,4) has succumbed to a massive amount of timing errors. This is due to the specific gate-level and critical-path structure in this adder, which is not easily predictable in designs optimized by industry compilers. Though, if this specific ISA has a low sensitivity threshold to timing errors, it is still better than the exact adder in terms of joint error.

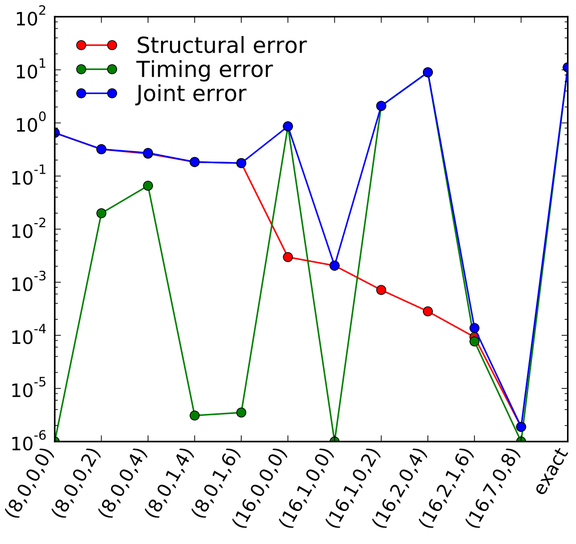
At 10 % overclocking (Fig. 2.18b), timing-error contributions strongly increase, but stay lower than structural-error contributions for low-accuracy ISA adders. Two additional high-accuracy ISA circuits have fallen to timing errors: ISA (16,0,0,0) and (16,1,0,2) ISA circuits. Yet, they are still operating slightly better than the exact adder, whose average error, entirely due to timing, has been multiplied by 3 compared to circuit overclocked by 5 %.

At the highest overclocking ratio of 15 % (Fig. 2.18c), all the selected high-accuracy ISA designs have fallen to timing errors. Yet, some of these designs still exhibit decent overall accuracy such as ISA (16,2,1,6). This latter relegates to the second place ISA (16,7,0,8), which has a more accurate architecture but is found less resilient to aggressive overclocking. Understanding this variability in timing-error robustness as well as the difference of threshold between structural and timing errors could be highly beneficial to low-power and time-constrained circuit design. This would require a deeper analysis combining more speculative designs to better cover the design space offered by inexact speculative circuits.

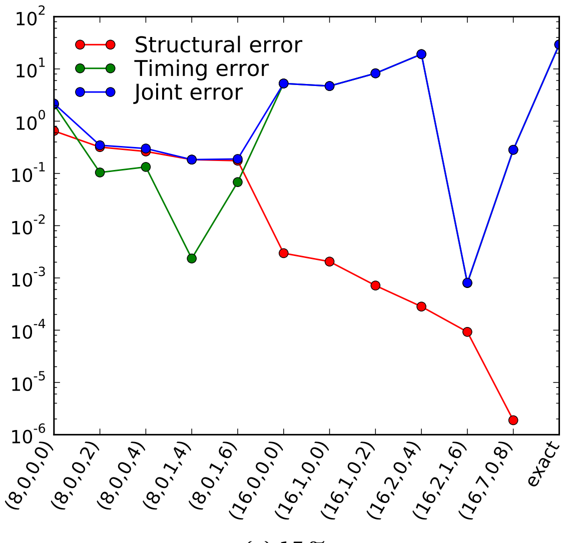
For low-accuracy ISA overclocked by 15 % (left part of Fig. 2.18c), it is particularly interesting to note the high balance between timing and structural errors. This compromise between the two error contributions gives generally a better overall accuracy than adders designed with high structural accuracy.



(a) 5 %



(b) 10 %



(c) 15 %

Figure 2.18 – Relative error RMS of ISA under 5, 10 and 15 % overclocking.

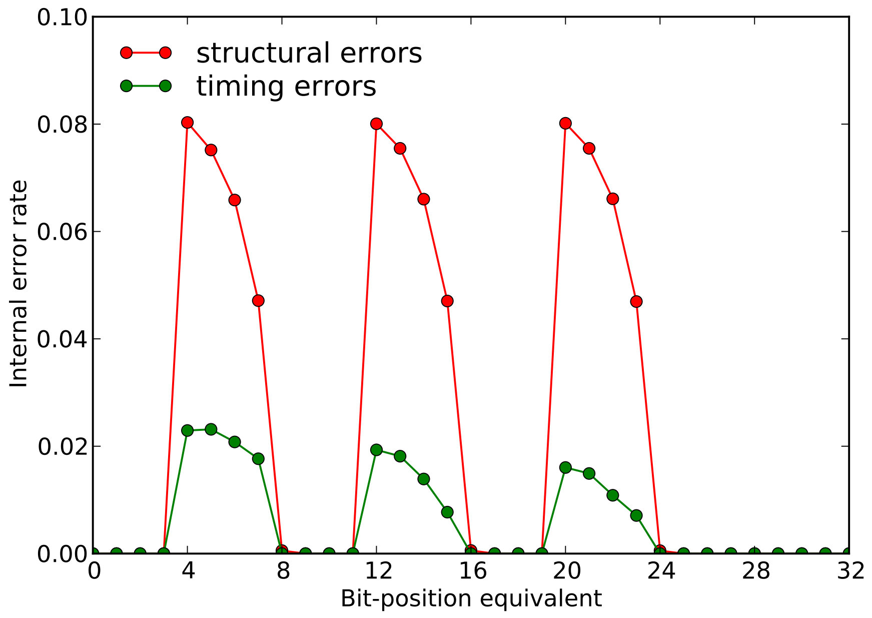


Figure 2.19 – Bit-level-equivalent error distribution in ISA (8,0,0,4) under 15 % overclocking.

Structural and timing error balance

In order to better understand how the two types of errors interplay with each other, Fig. 2.19 displays the internal distribution of structural and timing errors within the example of 15 %-overclocked ISA (8,0,0,4) since this configuration shows the best balance between errors (c.f. Fig. 2.18c). Arithmetic structural errors have been translated into their equivalent bit-level positions. Note that the timing errors distribution is not as regular as the structural errors distribution. While it is easy to distinguish when several arithmetic speculative errors occur simultaneously on different speculative paths and translate independent errors into bit positions, timing errors might span over various outputs.

Structural errors are immediately recognizable on three speculative paths (the first speculative path, operating from the LSB, uses directly the adder carry-in so doesn't have errors). As this ISA only has 4-bit error reduction (no error correction), it only introduces errors on the preceding sub-adder sums, that is why structural-error peaks are slightly shifted on the left of the figure.

In a conventional adder, overclocking would dangerously degrade MSBs. In this ISA, despite causing structural errors, the 4-path speculative structure leads to a split of critical path, distributing the timing errors over those paths instead of the MSBs. Those errors mainly occur on the 4-bit error reduction block, last logic element in the critical path. This trade-off between structural and timing errors demonstrates the good resilience of ISA architectures compared to conventional circuits.

2.5 Conclusion

This chapter has introduced two techniques for designing approximate arithmetic circuits, ISA and GLP. Based on two existing concepts, these techniques have strongly improved the state of the art. Maximizing the circuit efficiency and error controllability, the ISA has perfected and generalized the speculative adder architecture. GLP has fully automatized the pruning concept into a CAD tool that removes the least-significant gates to reduce energy consumption and silicon area.

As these approximate circuit techniques lead to different types of errors, it has been shown that their combination maximizes their benefits without degrading accuracy. Mixed pruned-speculative adders can achieve up to 85 % PDAP reduction for a RMS relative error of 1 %. Finally, a timing-error prediction model has been developed, together with a methodology to combine structural and timing errors, and has proven the strong robustness of ISA against overclocking.

3 Approximate circuits by fabrication of false timing paths

This chapter introduces a novel concept to optimize arithmetic circuits by artificially inserting and exploiting false paths, and co-designing circuit implementation together with circuit functionality. This technique is demonstrated for the design of an approximate adder, called the Carry Cut-Back Adder (CCBA), that trades off arithmetic precision in a floating-point manner.

An exhaustive and industry-oriented comparison of 32-bit CCBAs against 10 state-of-the-art approximate adders, including truncated exact adders, is performed for two target frequencies. All circuits are described at behavioral level and synthesized in an industrial design manner for a 65 nm commercial CMOS technology. Mean and maximal relative errors are used to assess the accuracy. Results show that the CCBA offers by far the best performance for worst-case relative errors. For mean relative errors, the CCBA is proven to be comparable to truncated adders among high-accuracy circuits, and to outperform truncation in low-speed configurations.

The organization of this chapter is as follows: Section 3.1 gives an overview of the concept of circuit optimization by fabrication of false timing paths. Section 3.2 details the architecture, arithmetic principle and design strategy of the CCBA. Section 3.3 finally presents the CCBA results and its comparison against truncated adders and state-of-the-art approximate adders.

The source code of the CCBA adder is available online at:

- https://github.com/vincent-camus/carry-cut-back-adder

This GitHub project contains the VHDL source code of the CCBA approximate adder, VHDL testbenches for behavioral and gate-level verification, synthesis scripts for both Synopsys Design Compiler and Cadence Genus, as well as simulation scripts for MentorGraphics Modelsim or Questa. It is provided with a part of the Nangate 45 nm open-cell library, a free predictive library developed by Nangate.

3.1 Approximate circuit design and optimization by fabrication of false paths

Digital circuits need to be designed to be functional in the worst-case scenario, i.e. when their *critical path* is activated. This is achieved by finding all potential critical paths and adopting on them conservative—thus expensive—timing margins. But sometimes, it happens that a critical path *can never be logically activated*, it is then called a *false path*, as it is unnecessary to apply conservative constraints over it.

False paths are traditionally unexpected byproducts of circuit design. Finding them and obtaining their information, known as delay constraints or timing exceptions, makes it possible to relax timing constraints on signal paths during the Static Timing Analysis (STA). It can enable the synthesis tool to achieve the desired design performance (e.g. power, area, or speed) or timing closure by focusing efforts on real paths instead of false paths. Thus, many scientific articles [40, 41] and patents [42, 43, 44] have described techniques to identify them in the circuit netlist by analytical or numerical ways.

3.1.1 False-path fabrication

The novelty and main interest of this new technique is to artificially introduce and exploit false paths to optimize the implementation of digital circuits [45, 46, 47]. Preventing full activation of a delay-critical signal in a circuit by inducing a false path allows more relaxed timing constraints, resulting in lower circuit implementation cost, higher yield, or earlier signal arrival times. In some cases, if a signal path originally fails to fit the delay constraint, this technique can make it possible to fit the constraint without the need to redefine design specifications, or without costly methods such as upsizing cells and transistors, buffering, parallelizing or increasing pipelining.

In order to create a false path or transform a signal path into a false path, we introduce two required logic elements:

- A *cutting* element multiplexing the signal path in its center, either to maintain the path itself, or to substitute it for an alternate path that is faster.
- A *monitoring* element triggering the cutting element to select the faster alternate path when it detects a possibility of full signal-path activation based on the monitoring of a few related nets.

Fig. 3.1 illustrates an example of fabricated false path. The monitoring block tracks a portion of the circuit related to the original signal path in order to detect potential propagation across it. If such risk is detected, it generates a signal that triggers the cutting element into selecting a shorter signal path. Independently of all the other (non-monitored) parts of the signal path, a

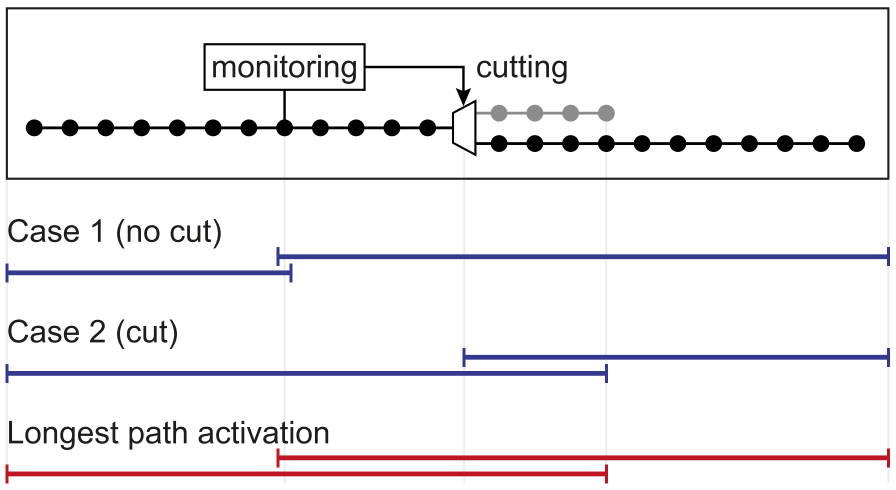


Figure 3.1 – Diagram illustrating a fabricated false path. The monitoring block tracks signals related to a stage of the original black-dotted signal path. When propagation is possible through this stage, it triggers the cutting multiplexer to select the gray path, resulting in a shorter effective propagation.

case analysis of the longest possible logic propagation along the signal path results in shorter path activation, thus lower path delay.

An important step is to manually exclude the generated false paths from STA. This is the case for the CCBA circuit described in the following section, for which signaling timing exceptions is crucial to get the correct implementation. Indeed, identifying false paths in a circuit is a non-trivial and computer-demanding task. Thus, omitting to signal them to the synthesis tool is likely to lead to a miss. In that case, the tool would unnecessarily attempt to meet delay constraints on them, losing all the benefits of the technique. Different types of timing exceptions are possible, depending on the synthesis tool, for instance *set max delay*, *set false path* or *set disable arc*.

3.1.2 Significance-driven cuts

It is important to note that inducing or fabricating a false path requires a careful co-design of circuit timing together with circuit behavior. Indeed, the non-complete path activation will prevent its full and (assumed) original behavior to establish, thus altering the overall circuit behavior. For this reason, this technique is best suited for the design of approximate circuits, for example by transforming an exact design into an approximated version lacking full functionality.

The greatest challenge is undeniably to find the right signal path and to implement a cut that guarantees minimal and controlled changes on the functionality, such as reduced precision or controllable arithmetic errors. An evident application of this technique is thus on arithmetic operators and computing datapaths. Indeed, their intrinsic bit and net significance can help to guide the selection of monitoring and cutting elements. Furthermore, in arithmetic circuits, the critical path generally spans from low to high significance signals. Thus, the monitoring can be configured to track one or more high-significance signals, while the cutting would occur at a lower-significance position.

The cuts can be designed within a circuit architecture, as proposed in the next section, or integrated within a CAD tool. Such tool could automatically fabricate false paths within combinational or arithmetic circuits to relax costly or unreachable timing requirements.

Fig. 3.2 shows an example of cuts introduced in a circuit to reduce delay while ensuring the desired accuracy or functionality. In the initial circuit of Fig. 3.2a, three long signal paths fails to fit the 10 ns delay constraint. Different cuts, pictured as arrows, are introduced on these paths in Fig. 3.2b to trade accuracy for better delay. But as the cuts introduce errors, the functionality or accuracy is too deteriorated. Thus, Fig. 3.2c presents a trade-off with less invasive cuts, in which the timing is only partially improved, but the accuracy is sufficient.

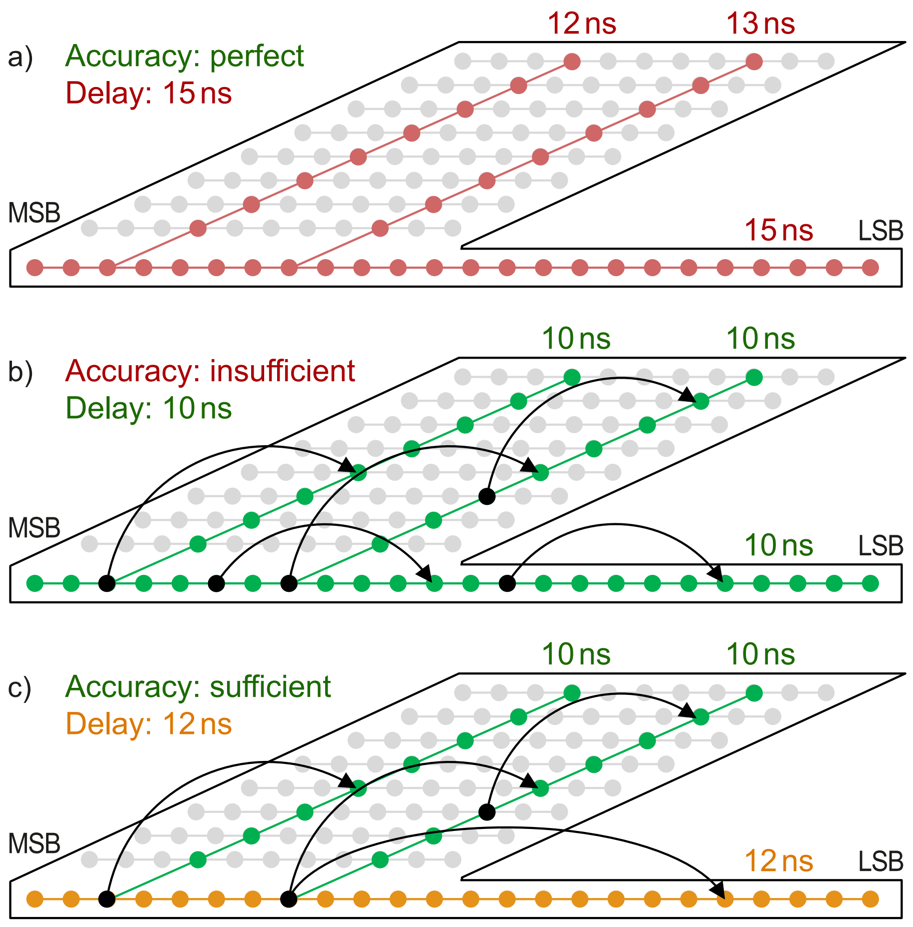


Figure 3.2 – Example of false-path fabrication steps to optimize delay and accuracy. The initial arithmetic circuit (a) fails to fit a 10 ns delay. Different cuts, pictured as arrows, are introduced in (b) and (c) to trade accuracy for better delay.

3.2 Carry cut back adder (CCBA)

3.2.1 State-of-the-art approximate adders

Additions are the most common arithmetic units in digital systems. With the demand for higher speed and power efficiency, many attempts have been made to build them in an approximate manner. This section catalogs and reviews in details existing approximate adders.

The most common approach to build approximate adders is using the concept of *carry speculation* [16]. As carry propagation typically does not cover the entire length of the adder, it is feasible to guess relatively accurately an internal carry based on a small number of preceding stages. As a result, the carry propagation chain can be reduced or sliced into multiple shorter paths executed in parallel, enabling performance beyond theoretical bounds of exact adders.

Different speculative schemes have been explored in the literature, among which *segmented* [48, 18], *compensated* [19, 49, 33] and *timing-starved* adders [50, 51]. As they contain the full addition stages (simply unconnected to be executed in parallel), all these adders can easily and inexpensively be made dynamically configurable to allow exact computation or variable-latency correction [48, 49, 50, 51]. This work solely considers their core architecture without such features. Other architectural techniques are based on simplifying LSBs of the addition [20, 35], either by replacing the low-significance full-adder cells by an approximate counterpart, or by pruning low-significance gates after circuit synthesis.

Speculative segmented adders

Early speculative adder works are based on the Equal Segmentation Adder (ESA) [48]. The ESA simply slices the addition into multiple sub-adder blocks executed in parallel, without carry propagation amongst themselves. This segmented carry chain without any circuit overhead offers a high energy efficiency at the cost of numerous and uncontrolled errors.

To reduce the error rate, the Error-Tolerant Adder type II (ETAII) [18] complements the sub-adders with equally-sized carry generator sub-blocks to speculate more accurately the input carry of each sub-adder.

Speculative compensated adders

In order to reduce the error impact and limit the worst case, segmented adders have been coupled with multiplexer-based error compensation. The Error-Tolerant Balancing Adder (ETBA) [19], direct descendant of the ETAII, uses an error balancing technique based on multiplexers to mitigate the relative error on the preceding sub-adder block in case of incorrect carry speculation.

The Generate-Signals-Exploited Carry Speculation Adder (GCSA) [49] has a functionally similar carry scheme as the ETBA. It differs by introducing, in case of incorrect speculation, the error reduction on the current sub-adder block rather than on the preceding one.

Introduced in section 2.1, the ISA is a generalized and optimal architecture of speculative adder. It minimizes the carry-generator overhead, reducing its large critical delay. It also optimizes error reduction with a dual-direction compensation on both preceding and current sub-adder blocks. The ISA improves accuracy while strongly outperforming other speculative adders in terms of speed and energy efficiency. It is worth noting that the ISA encompasses the state-of-the-art segmented and compensated adders, those being boundary cases.

Speculative timing-starved adders

The Almost Correct Adder (ACA) [50] is the best-known timing-starved adder. It is composed of an array of overlap-ping and translated sub-adder blocks, so that each sum bit of the ACA is constructed using exactly the same amount of preceding carry stages (except the first ones, which require less). The critical-path delay is thus limited, but the circuit cost is fairly high.

As for the ACA, the Accuracy-Configurable Approximate Adder (ACAA) [51] is also composed of overlapping sub-adder blocks. But those sub-adders are translated by half of their bit-width. Thus, fewer blocks are required and the circuit complexity is reduced.

Adders with simplified LSB

The Lower-part-OR Adder (LOA) [20] divides the addition into two parts. The upper part computes precise addition of the MSBs, while some OR gates approximate the lower-part addition instead of conventional FA cells. An extra AND gate is used to generate the carry-in of the upper part addition from the preceding stage. Despite a high error rate, error values remain small, while the critical path is reduced to the carry chain of the upper-part adder only.

Gate-Level Pruning (GLP) [35] belongs to the class of CAD tools to automate the design of approximate circuits. It removes low-significance gates, trading accuracy in exchange for area and power savings. GLP has been successfully applied to adders, for which it retains the gates required for accurate carry propagation while discarding those used for generating low-significance outputs.

3.2.2 Proposed architecture

The general structure of the proposed Carry Cut-Back Adder (CCBA) is depicted in Fig. 3.3. As presented in [45], the CCBA is built on an ordinary fixed-point adder composed of the chain of sub-adder blocks (ADD), with insertion of multiplexers that can cut the carry chain to shorten the effective critical path. The cut multiplexes the real carry with a carry speculated from a much shorter chain.

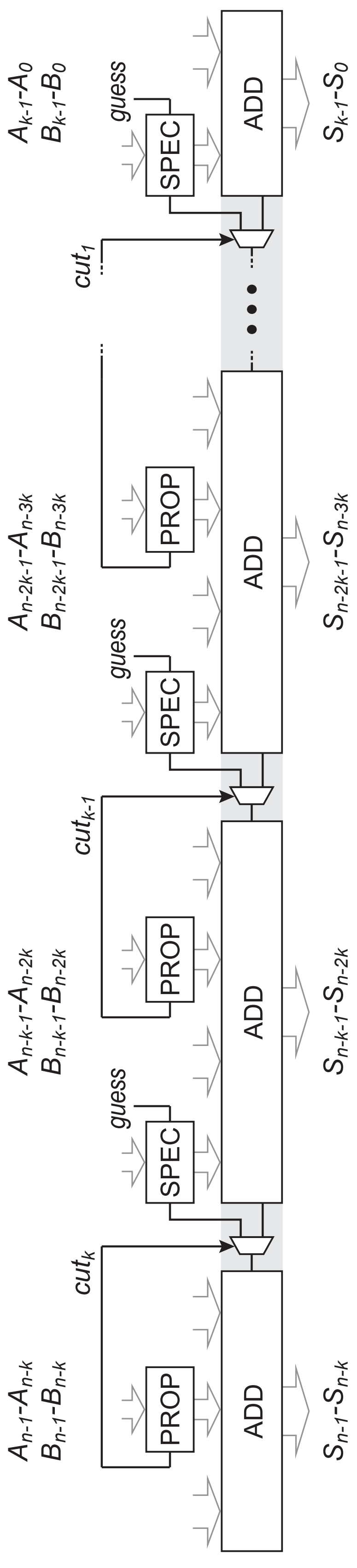


Figure 3.3 – General block diagram of the proposed Carry Cut-Back Adder (CCBA).

The decision to cut the chain is taken in the carry propagate block (PROP) that monitors a group of carry stages and generates the cut signal if those are all in propagate states. The cut always occurs at a lower-significance position than the PROP in the carry chain, guaranteeing low relative errors.

Note that this *cut-back* mechanism appears as a feedback between two carry-chain positions. But the PROP only uses local propagate and generate signals, which are computed from the input operands and not from the carry chain. It is therefore not a recursive loop and cannot influence the circuit stability.

The speculated carry is generated in the optional carry speculator block (SPEC) with a *guess* value, identical if there are multiple cut-backs. Shorter than the exact carry path, this alternate path speculates the carry from a few preceding stages and propagates the *guess* if those stages are all in propagate mode. The *guess* is usually a hardwired ‘0’ or ‘1’, but it can be a dynamic value, such as a preceding-stage input operand to avoid a bias in the error distribution [19].

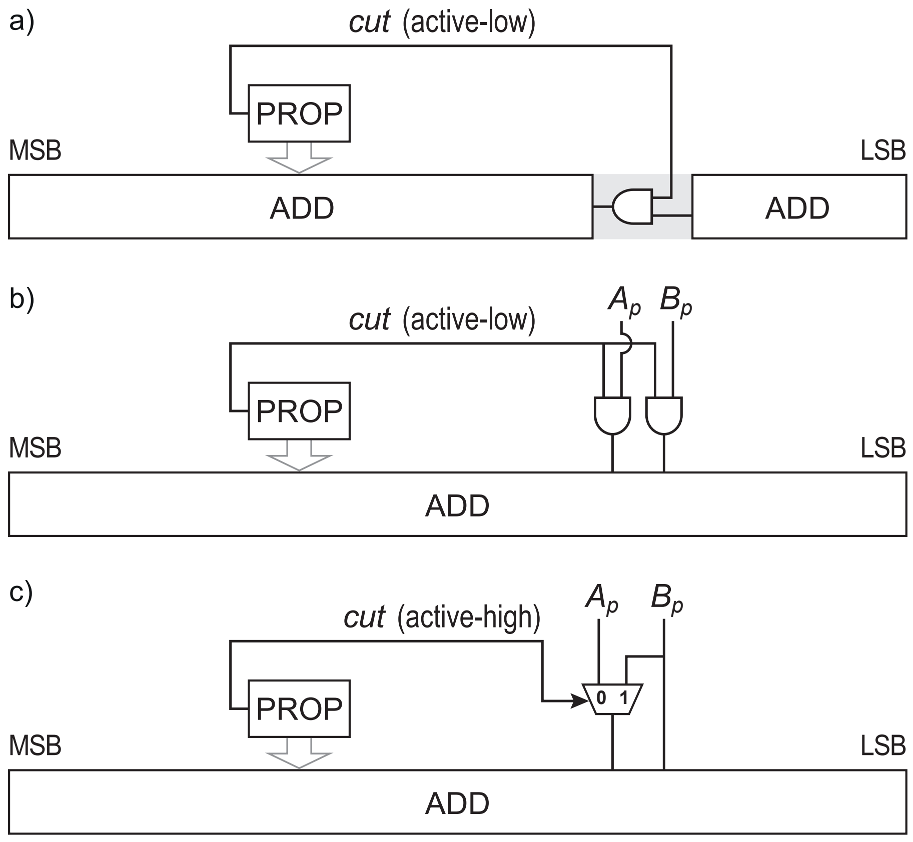


Figure 3.4 – Examples of CCBA without SPEC. (a) shows an AND-cut, equivalent to 0-bit SPEC with *guess* = 1. (b) and (c) show input-induced cuts. (b) sets the p^{th} inputs to ‘0’, forcing a kill state. (c) sets both inputs to the same value, allowing kill or generate states only.

If there is no SPEC block (equivalent to a 0-bit SPEC), the multiplexer can be simplified to a monotonic gate, as in the *AND-cut* of Fig. 3.4a where $cut = 0$ dictates the AND output regardless of its second input. Another solution, called the *input-induced* cut, is to induce the cut from the input operands themselves, as in Figs. 3.4b-c where both stage inputs are zeroed or given the same value to allow kill or generate states only. A different HDL description is required for those latter implementations.

3.2.3 Circuit timing

The main advantage of this approach remains in its timing characteristic. In a regular adder, the critical path is only activated if all the stages are in propagate mode. This occurs with a low probability as the carry propagation is naturally broken by the distribution of input bits. The adder within the CCBA physically contains the entire carry chain (through ADD blocks and multiplexers), but *this path can never be activated.* By monitoring a few stages of the adder, the PROP detects such risk and calls the SPEC as shorter path to be used instead, ensuring that the design meets tighter timing constraints.

Fig. 3.5 shows a case study of the longest propagation chains that can flow through a CCBA built with two OR-cuts enabled by active-high *cut* signals (example without SPEC for simplicity, but same reasoning in the general case). Each cut-back module splits the carry chain with two possibilities:

- $cut = 0$: The OR gate output follows the input, propagating the carry from one ADD block to the other. In this typical case, no intentional cut happens at the cut position, but the carry chain is naturally broken within the PROP among the ADD_2 stages. The critical path is therefore limited, as it cannot entirely cross over the PROP.
- $cut = 1$: All the stages within the PROP are in propagate mode. The carry necessarily propagates through the PROP and there is a risk of long critical-path activation if the other non-monitored stages are also in propagate mode. The active-high *cut* signal deliberately forces the OR output at '1' no matter the real carry coming from the preceding ADD block. The carry propagation is therefore interrupted at this position and its maximum length remains limited.

As explanation of Fig. 3.5, no intentional cuts occur in case 1, the carry chain being naturally broken within the two PROP blocks. Two deliberate cuts occur in case 2, artificially breaking the carry chain at the OR gate positions. Cases 3 and 4 both contain one naturally broken chain and one deliberately cut, shortening carry chains to various lengths.

Despite the fact that the full carry chain physically exists in the design, no input combination can activate it from the start to the end. It is therefore a *false path*, and it can be excluded from the timing optimization. The *effective* critical paths, in red in Fig. 3.5, sum up the longest

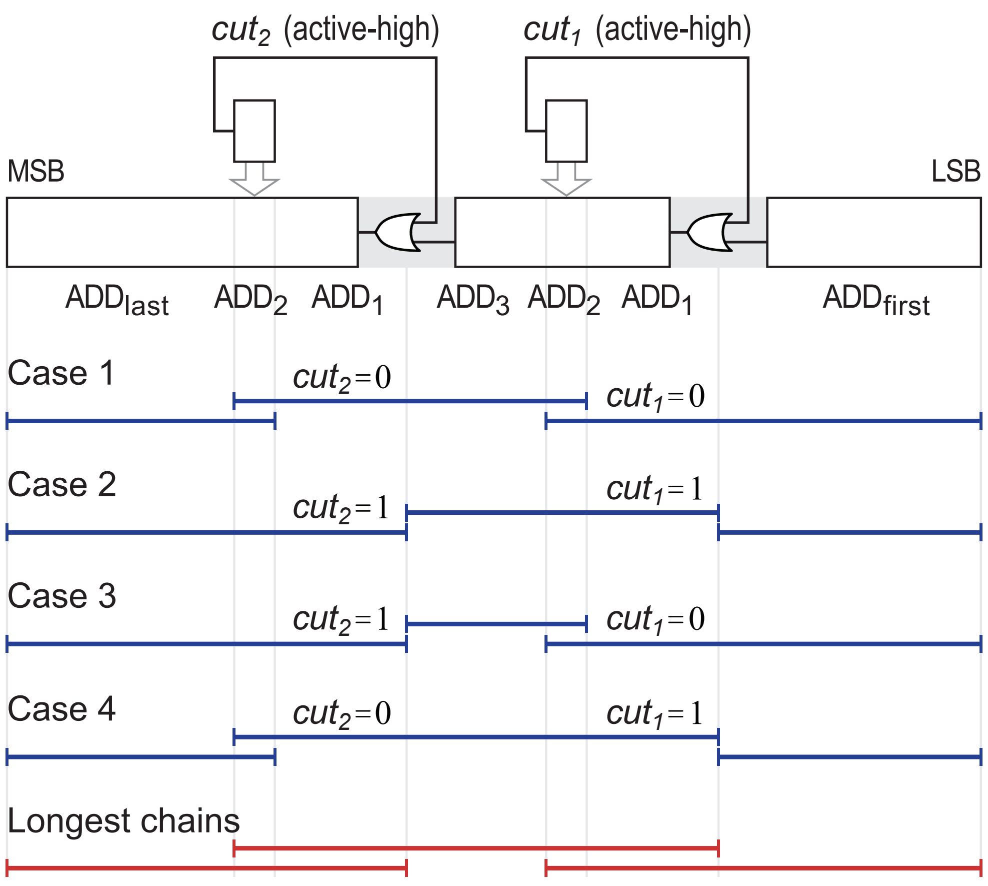


Figure 3.5 – Diagram of the longest carry chains and resulting effective critical paths in the example of an implementation of CCBA with two OR-cuts.

propagate chains that can occur in the circuit among the different cases. Insertion of more carry cut-back modules, possibly overlapping each other, would lead to shorter effective critical paths.

3.2.4 Arithmetic and errors

The CCBA addition arithmetic is illustrated in Fig. 3.6. Stages within PROP and SPEC blocks are indicated with their carry state: P, G and K, representing propagate, generate and kill states, respectively. Cuts and signals are drawn in dotted lines when cuts are inactive.

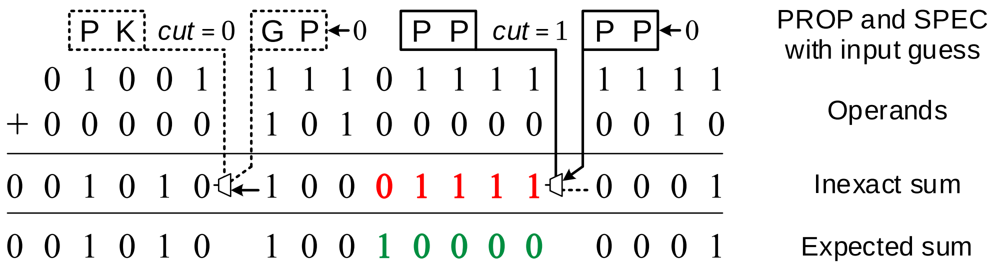
An error only occurs with the concurrence of three factors:

- A sequence of propagate signals spans the entire PROP bit-width, triggering the cut.
- A sequence of propagate signals spans the entire SPEC bit-width, making the exact carry prediction impossible with the SPEC stages only.

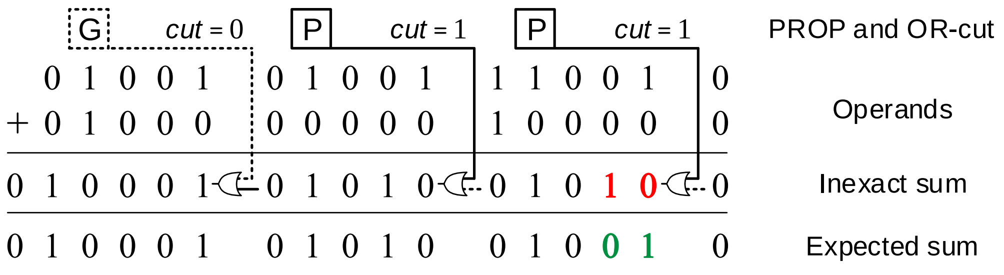
- A wrong guess of the carry that inputs the SPEC (Fig. 3.6a) or that directly substitutes for the real carry (Fig. 3.6b).

Because of the simultaneous occurrence of the three aforementioned properties, an error occurs in the right-hand path of Fig. 3.6a. In the OR-cut implementation of Fig. 3.6b, the active-high *cut* signal is also the guess value due to the use of the OR gates. The first error condition is met for the two right-hand paths, directly triggering the cut since there is no SPEC. The guess value at '1' unintentionally follows the real carry in the central path and leads to a correct sum. But it happens to be wrong in the right-hand path and leads to a faulty sum.

Occurrence of an error implies that one or both operands have non-zero bits at the PROP position to drive those stages into propagate mode. As the error occurs at the cut position, at a lower-significance position, the expected sum is necessarily much larger than the introduced error. In the example of Fig. 3.6a, the absolute error is 16 while the expected sum is 43,265 so the relative error is 0.04 %. In the computation of Fig. 3.6b, the relative error is only 0.006 %. Such low relative errors are typical in speculative adders for calculations involving large value operands. However, it is the worst case that gives the upper-bound relative error and defines the minimum precision of an adder.



(a) CCBA with two multiplexed cuts with 2-bit PROP and SPEC blocks and guess at '0'



(b) CCBA with three OR-cuts with 1-bit PROP blocks

Figure 3.6 – Example of CCBA addition arithmetic for two circuit architectures. Dotted lines depict inactive cuts.

3.2.5 Worst-case error and floating-point precision

Error propagation

It is interesting to note from Fig. 3.6 that the error caused by the cut can propagate on many bits, but seems to keep the magnitude of the carry cut-back position, i.e. the first wrong bit. However, a series of erroneous bits can result in very different arithmetic errors due to compensation mechanisms. Thus, a careful demonstration has to be provided.

Let S_i, C_i and P_i denote the sum, carry-in and propagate signals of the i^{th} stage addition, respectively. The sum and carry propagation are defined by:

$$S_i = P_i \oplus C_i \tag{3.1}$$

$$P_i = 1 \implies C_{i+1} = C_i. \tag{3.2}$$

Let assume a carry error at the i^{th} bit of the adder, with an erroneous carry of value C_{err}. The sum bit and the carry-out depend on the value of P_i:

- If $P_i = 1$, (3.1) gives $S_i = \overline{C_{err}}$ instead of C_{err} for the expected sum bit, while (3.2) propagates the wrong carry C_{err} to the next stage, where the same formulae apply again.
- If $P_i = 0$, (3.1) gives $S_i = C_{err}$ instead of $\overline{C_{err}}$ for the expected sum bit, but the wrong carry is not propagated by (3.2), so the next stage addition is correct.

Assuming that the erroneous sum spreads from the m^{th} to the p^{th} stage, the error pattern appears as shown in Fig. 3.7:

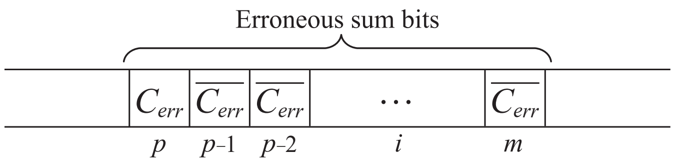


Figure 3.7 – Balanced error pattern.

Just as in Fig. 3.6, where the error patterns are shown in red, the last faulty bit counterbalances the first ones, the absolute error has the significance of the first erroneous bit only:

$$2^p - 2^{p-1} - 2^{p-2} - \cdots - 2^m = 2^m. \tag{3.3}$$

This result is valid if the carry propagates normally. But some circuits can have more than one cut-back multiplexer. If the stages between two cut-backs are in propagate modes, the normal propagation driven by (3.2) could be disrupted. Thus, the previous result needs to be recomputed for this case.

Let assume the same carry error ($C_i = C_{err}$) in a propagating stage ($P_i = 1$, else there would be no carry-chain perturbation). If another cut-back happens to guess the same faulty carry C_{err}, it does not disrupt the normal propagation and the previous result holds (3.2). But if the carry cut happens in the opposite direction $\overline{C_{err}}$, it overrides (3.2) and reverses the carry error: the carry, that was false until now, comes back to the value of the expected addition. Thus, despite the cut, the current sum bit, determined by (3.1) with the exact carry, is correct, as well as the next stages.

The last erroneous sum bit being the preceding stage at value $\overline{C_{err}}$, the error pattern appears this time as in Fig. 3.8:

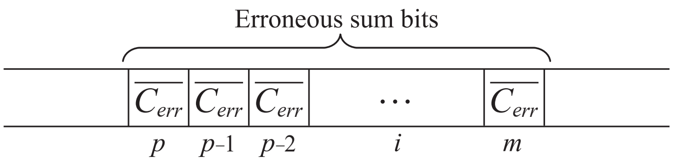


Figure 3.8 – Unbalanced error pattern.

All the erroneous bits are in the same direction and the absolute error is simply their sum:

$$2^p + 2^{p-1} + 2^{p-2} + \cdots + 2^m = 2^{p+1} - 2^m. \tag{3.4}$$

The magnitude of this error is much higher than in the first case, but it can only occur if several carry-cuts happen in opposite directions. Thus, to avoid such dramatic errors, the SPEC guess or the straight carry-cut must be chosen in the same direction for all the cut modules of the CCBA.

Worst-case relative error

Having validated the fact that any error only has the magnitude of the bit position of the cut that caused it, the low impact of this error on the expected sum should be demonstrated.

The worst case happens when the error magnitude is the highest on the lowest expected calculated sum. Occurrence of an error implies that the three factors mentioned in section 3.2.4 are realized, thus the PROP and SPEC blocks intercept only propagate signals. All the non-zero operand bits producing those propagates add up to the expected sum:

- The PROP non-zero bits, which significantly contribute to maximizing the expected result and thus to minimizing the worst-case relative error.

- The SPEC non-zero bits, which contribute to a lower extent in increasing the sum by attenuating a portion of the magnitude of the error.
- If the SPEC guess or straight carry-cut is '0' (speculating a low carry), an error replaces a real carry necessarily at state '1' coming from a generate stage. Added to the SPEC non-zero bits, this stage further increases its sum to 2^m.

Thus, whenever an error occurs, while it keeps the magnitude of the cut bit significance, i.e. an arithmetic error of value 2^m, the sum is always expected to be greater than:

$$2^m + \sum_{k \in \text{PROP}} 2^k \quad \text{and} \quad \sum_{k \in \text{SPEC}} 2^k + \sum_{k \in \text{PROP}} 2^k, \tag{3.5}$$

leading to a relative error lower than:

$$\frac{2^m}{2^m + \sum_{k \in \text{PROP}} 2^k} \quad \text{and} \quad \frac{2^m}{\sum_{k \in \text{SPEC}} 2^k + \sum_{k \in \text{PROP}} 2^k}, \tag{3.6}$$

in the cases where the carry guess is at '0' and '1', respectively. This result holds also if multiple errors occur in different carry-cut modules, since the ratio of error over sum is preserved.

As a result, a floating-point precision can be achieved at design time through a proper sizing and positioning of PROP and SPEC blocks and selection of the right carry guess. It is easy to verify that the worst-case relative error is 7.7 % for the example in Fig. 3.6a and 12.5 % in Fig. 3.6b, corresponding to minimum precisions between 4 and 5 bits.

3.2.6 Design and implementation strategy

The CCBA allows considerable improvements concurrently in circuit performance and error control. This section describes how to exploit its architectural advantages.

Circuit implementation

As the carry cut-back technique introduces hardware overhead to the adder, mainly for implementing the PROP and SPEC blocks, their circuitry must be minimized. The easiest way is to limit their bit-widths to a few stages in order to reduce their area and delay, as those need to be computed at first to determine the activation and value of the cut. Executed in parallel, the delay overhead is limited by the slowest between PROP and SPEC.

Usually implemented with a fast and efficient carry-lookahead architecture, the area overhead of these functional blocks can fortunately be balanced. Indeed, PROP and SPEC can be built with a similar architecture than the adder segments that they overlay. They could then share most of their circuitry, for instance with a carry lookahead adder, that uses small sum generators onto a carry-lookahead network.

It is worth mentioning that alike speculative adders, the CCBA can easily be made dynamically configurable to embed an exact computation mode or a variable-latency error correction mechanism. The cut-backs simply have to be disabled, the carry will propagate normally throughout the entire adder chain (with the original critical-path delay) and directly lead to the exact addition result.

Timing constraints

To effectively benefit from the cut-back mechanism, it is necessary to manually exclude the generated false paths from STA. As mentioned in 3.1.1, this ensures that the synthesis tool does not unnecessarily spend resources to meet delay constraints on them.

This can be achieved by providing the tool with a timing exception script based on the CCBA design information. An easy way to do so for a single arithmetic module is to set multiple delay constraints for each effective path instead of a single constraint for the entire design. They would correspond to the longest paths shown in red in Fig. 3.5 (not forgetting the SPEC if there is). For instance, on the CCBA implementation of Fig. 3.6a, delay constraints should be set on the three longest paths: from input to output bits [0..7], [2..15] and [10..17]. For the CCBA of Fig. 3.6b, there are this time four longest paths: [0..4], [1..9], [6..14] and [11..16].

Error trade-offs

The CCBA offers a large design space to co-design circuit timing together with functional errors. This allows maximizing circuit savings while minimizing the application quality loss. It also enables to dissociate arithmetic precision from dynamic range, usually fixed by the bit-width.

Firstly, the CCBA design parameters can independently set different error characteristics:

- The error rate depends on the number of cut-back modules and of the PROP and SPEC bit-widths.
- The maximum error can be adjusted mainly by sizing the cut length and the PROP bit-width, and to a lesser extent by modifying the SPEC bit-width and input guess.
- Adjusting other error metrics, such as SNR, Root Mean Square (RMS) error or any other statistical metric can be achieved using the similar models than those built for speculative adders [52, 53].

Secondly, the same design parameters affect circuit timing and costs in terms of area and power, intuitively:

- More cuts leads to a faster circuit, but at the cost of more and higher errors, as well as circuit overhead required for SPEC, PROP and multiplexing.
- Larger SPEC or PROP reduces the errors at the cost of delay, area and power overheads.
- Increasing the cut length lowers the error impact without hardware overhead. However, a longer cut induces a lower timing relaxation, leading to lower circuit savings.
- To optimize the timing budget, it can be interesting to equalize the delay of all the timing paths, by equally spacing cut-back modules and by adusting the bit-widths of the first and last ADD blocks.

3.2.7 Design-space minimization

Due to its flexibility, the choice of the right set of parameters remains cumbersome. Fortunately, the dependency between errors and circuit characteristics can help limiting the design-space exploration to a low number of possible candidates.

In order to find the minimum design requirements to fit a given maximum relative error, a simple methodology can be derived by reversing equation (3.6). First, neglecting the terms associated to the SPEC gives the minimum *cut length*, composed of the bit-width of the PROP (l_{PROP}) and of the ADD$_1$ (l_{ADD_1}) blocks, as on Fig. 3.5:

$$l_{\text{cut}} = l_{\text{PROP}} + l_{\text{ADD}_1} \geq \lfloor 1 - \log_2(\text{RE}_{\text{max}}) \rfloor \tag{3.7}$$

where RE_{max} is the maximum relative error requirement. For instance, RE_{max} constraints of 50 %, 25 % and 12.5 % would impose minimum overall cut lengths of 2, 3 and 4 bits. The rounding down is due to the fact that the PROP and SPEC can be used for error compensation as well. But the choice of a greater cut length would ensure the required accuracy without constraint on those blocks.

Fixing this integer term while reversing (3.6) allows expressing the minimum required PROP bit-width to further limit the error and fit the desired RE_{max}:

$$l_{\text{PROP}} \geq \left\lceil l_{\text{cut}} - \log_2\left(2^{l_{\text{cut}}} + 1 - \frac{1}{\text{RE}_{\text{max}}}\right)\right\rceil \tag{3.8}$$

for a carry guess fixed at ‘0’. Note that the SPEC does not influence the maximum relative error in case of carry guessed at ‘0’, as suggested by (3.6). Therefore, no other design parameter can help limiting the error, this is why the rounding up is necessary to ensure the required accuracy.

In the case where the carry guess is dynamic or fixed at level '1', the expression is changed to:

$$l_{\mathrm{PROP}} \geq \left\lfloor l_{\mathrm{cut}} - \log_2\left(2^{l_{\mathrm{cut}}} - \frac{1}{\mathrm{RE}_{\mathrm{max}}}\right)\right\rfloor. \tag{3.9}$$

Choosing a higher integer would guarantee fitting the error constraint. However, for that case, the SPEC can finally complement the cut length and PROP for error compensation. The minimum required SPEC bit-width is expressed as:

$$l_{\mathrm{SPEC}} \geq \left\lceil -\log_2\left(1 + 2^{l_{\mathrm{cut}}} - 2^{l_{\mathrm{cut}} - l_{\mathrm{PROP}}} - \frac{1}{\mathrm{RE}_{\mathrm{max}}}\right)\right\rceil. \tag{3.10}$$

For example, let assume a 7.7 % relative error constraint. Equation (3.7) directly implies a minimum cut length of $l_{\mathrm{cut}} = 4$. For the case of a fixed guess at '0', (3.8) simply gives a minimum PROP bit-width of $l_{\mathrm{PROP}} = 2$ (no requirement for SPEC). For the case of a dynamic or fixed guess at '1', (3.9) and (3.10) give requirements of $l_{\mathrm{PROP}} = 2$ and $l_{\mathrm{SPEC}} = 7$. A 7-bit SPEC seems unreasonable for circuit efficiency. Choosing a larger PROP (e.g. $l_{\mathrm{PROP}} = 3$) directly guarantees to fit the chosen $\mathrm{RE}_{\mathrm{max}}$ without such unrealistic SPEC requirement. For both guess values, circuits with larger cut length (e.g. $l_{\mathrm{cut}} = 5$) fit the error straight away no matter PROP and SPEC bit-widths.

The number of cut-backs and their absolute positions do not influence the $\mathrm{RE}_{\mathrm{max}}$. Thus, many implementations remain to be investigated to find the optimal circuit, but the design space would be reduced to a few dozen candidates.

3.3 Results and comparison

3.3.1 Methodology

Scope of the exploration

In this work, 32-bit unsigned adders have been investigated in the frame of controlling and lowering the relative errors, particularly in its worst case, which delimits the floating-point error and thus the minimum precision of the operator.

To further narrow the design space, all the architectures of approximate adders have been kept uniform and regular for fair comparison, i.e. segmented and compensated adders using uniformly-sized blocks with identical properties and CCBA using uniformly-spaced and identical cut-back modules. They have all been considered with multiple types of carry generator guess: either fixed at ‘0’ or ‘1’, or dynamic using the preceding-stage input operand as introduced by [19].

More than 50,000 implementations of approximate adders with diverse error characteristics have been investigated to provide an overall picture of their performance.

Circuit implementation and characterization

All the approximate adders described in section 3.2.1 have been implemented in VHDL from the highest-level behavioral description. Therefore, the internal architecture of ADD, PROP and SPEC blocks is left to the compiler’s optimization, in order to benefit from the most favorable architecture to fit the timing constraint. The exact adder used as reference has been instantiated from the Synopsys DesignWare IP library.

All designs have been synthesized using Synopsys Design Compiler in the UMC 65 nm process for two target frequencies: 800 MHz (low-power) and 3.3 GHz (high-performance). Post-synthesis delay, area, leakage and dynamic power have been extracted by Synopsys Design Compiler. Only implementations meeting the timing constraints have been considered.

Circuits have been assessed in terms of silicon area, Power-Delay-Area Product (PDAP), and energy per operation at the targeted frequency. This latter has been chosen not to disadvantage slower circuits yet fitting the timing constraint.

Error characterization

The metrics used to characterize approximate adders are based on the relative error (RE), which has the advantage of being independent of the size of the adder, as defined in 2.1.5. The main metric used to characterize the circuits for this study is the maximum of the relative error (RE_{max}), which states the worst case or minimum precision of the circuit. It is also essential for targeting commercial products. The mean relative error (RE_{mean}) is also taken into consideration as it is widely spread in the analysis of approximate circuits [54].

The approximate adders have been characterized through the simulation of two samples of five million unsigned random inputs. First, a logarithmic distribution, exhibiting a very large dynamic range, has been used to detect the worst-case error RE_{max}. Then, a uniform distribution has been utilized to estimate RE_{mean}.

However, five million test vectors is a very small subset of the exhaustive simulation, which contains 1.8×10^{19} combinations of two 32-bit operands. In order to assess the quality of each estimate, three additional samples of five million inputs have been used to measure the deviation of the statistical estimate, a basic use of inferential statistics. Note that other methods have recently been proposed to improve the estimation accuracy by dynamically adjusting the number of simulation vectors [55] (see annex A) or by using a formal approach to analyze errors [56] (not used in this work).

As the error characteristics of adders spread over multiple orders of magnitude and are plotted on logarithmic scale, the Relative Standard Deviation (RSD) has been used (rather than the variance or the standard deviation) for measuring the dispersion of the results over the four random samples.

3.3.2 CCBA results

Error characteristics and circuit costs are shown for a selection of CCBAs at 3.3 GHz in Fig. 3.9 and at 800 MHz in Fig. 3.10. Circuit costs are normalized to the exact adder represented on the left of the figures. Sorted by RE_{max}, those designs have been selected as they represent optimal and well-balanced circuits.

CCBAs are tagged by architectures, 'm' for multiplexed and 'ii' for input-induced, and denoted by quintuples with their most important design parameters: (*number of cuts*, *cut length*, PROP, SPEC, *guess*). Fixed guesses are indicated by '0' or '1', and dynamic ones by 'A' or 'B'. Input-induced cuts are expressed the same way although they override two input signals instead of the intermediate carry.

To first assess the quality of the error characterization, RE estimations are shown with Relative Standard Deviation (RSD) error bars in logarithmic scale, magnified by a factor 500. The largest variation is observed in the middle of Fig. 3.9, but only represents a RSD of 0.4 % (i.e. the standard deviation on the estimation accounts for 0.4 % of the estimated value). The typical variations observed for RE_{mean} are in the order of 0.1 %, while around 10^{-3} % for RE_{max}, which confirms that five million vectors are sufficient for a correct estimation of RE_{mean} and RE_{max}.

Figs. 3.9 and 3.10 highlight the large design flexibility and accuracy range enabled by the proposed architecture. The CCBA design parameters allow tuning the precision on almost seven orders of magnitude of RE_{max} while exhibiting PDAP reductions up to 70 %. This

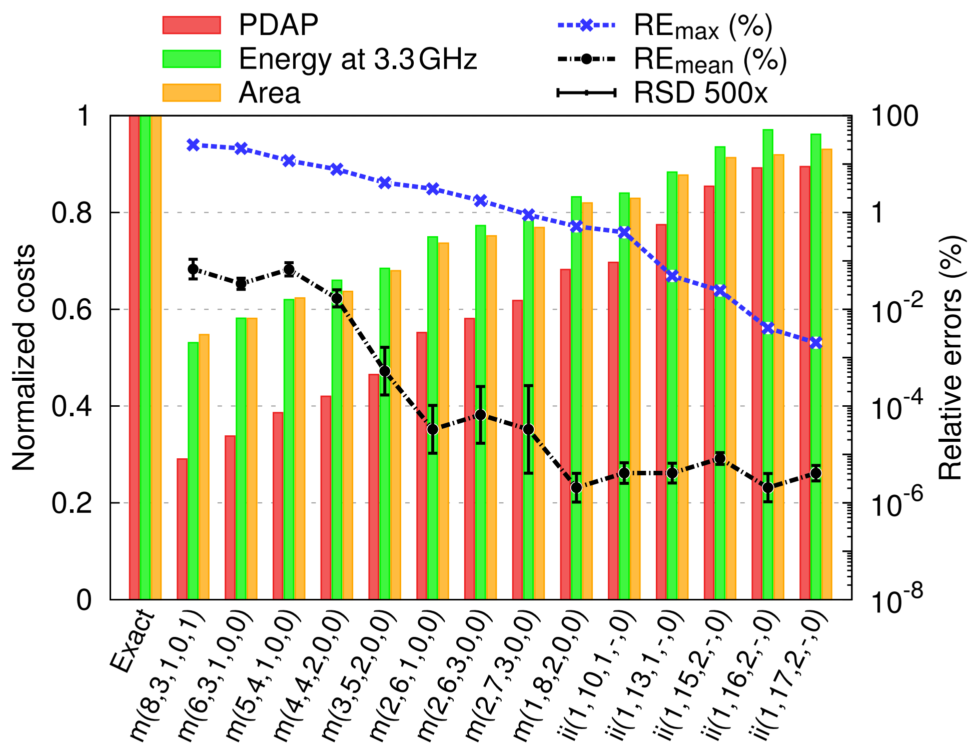


Figure 3.9 – Relative errors and normalized circuit costs of various 32-bit CCBAs (on the horizontal axis) synthesized at 3.3 GHz in a 65 nm technology.

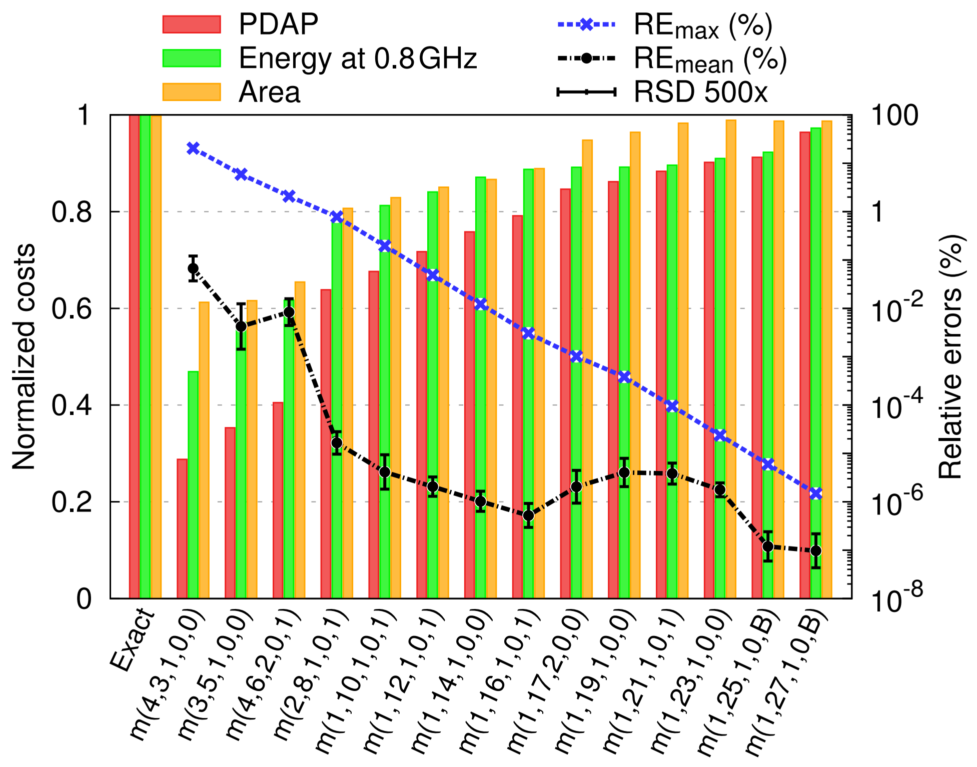


Figure 3.10 – Relative errors and normalized circuit costs of various 32-bit CCBAs (on the horizontal axis) synthesized at 800 MHz in a 65 nm technology.

precision tunability is three to four orders of magnitude higher than previously reported [45]. Those stunning results are enabled by the improvement of the HDL code and by the new input-induced cuts which mainly benefit to high-performance and high-accuracy adders (right of Fig. 3.9).

As expected from 3.2.6, the *cut length* (addition of ADD_1 and PROP bit-widths, as on Fig. 3.5) is the main parameter regulating the relative error. It ranges from 3 to 17 bits in Fig. 3.9, corresponding to RE_{max} from 25 % down to 2.0×10^{-3} %, and reaches 27 bits in Fig. 3.10 for a RE_{max} of only 1.5×10^{-6} %.

RE_{mean} follows the same trend as RE_{max} for implementations with multiple cuts (left of Figs. 3.9 and 3.10), for which a small change in their structure, repeated over many cuts, strongly impacts the overall error rate and mean. It scales in a lower degree among adders having a single cut and with higher accuracy (right of Figs. 3.9 and 3.10).

Target delay has a significant influence on the results. Low-speed implementations show better normalized savings than high-performance ones at equivalent precision. At 2 % RE_{max}, 3.3 GHz CCBAs achieve up to 25 % energy savings and 40 % PDAP reductions against around 40 % and 60 % for 800 MHz circuits.

Fig. 3.10 presents a sharp drop in circuit efficiency around 1 % RE_{max}. This corresponds to the precision from which the 800 MHz design becomes delay-constrained. Indeed, higher precision demands wider PROP, SPEC or greater cut length, which all lie in the effective critical path of the CCBA. This does not appear for 3.3 GHz adders, which are always tightly constrained.

Another irregularity occurs in the right-hand side of Fig. 3.10, where savings in high-accuracy adders appear to come exclusively from energy reduction. As for speculative adders, CCBA circuit savings usually come from the use of slower but more efficient topologies for the adder sub-blocks, enabled by the shortening of the critical path (e.g. Ripple-Carry instead of Han-Carlson, or Han-Carlson rather than Kogge-Stone). However, for very high accuracy, the large size of the sub-adders does not allow the compiler to select more area-efficient architectures, but it is still possible to relax timing and reduce dynamic power consumption compared to the exact adder.

Note that all the shown implementations have either a 0-bit SPEC or no SPEC at all, i.e. the penultimate design parameter being 0 or undefined, respectively. This is indeed appearing as the best circuit trade-off for lowering RE_{max}, sole criteria by which those designs have been selected. This is also true for the PROP bit-width, ranging from 1 to 3 bits only. Out of the scope of this thesis but interesting for future studies, the use and focus on other metrics, such as error rate or arithmetic errors, would lead to optimization strategies employing bigger SPEC and PROP blocks.

3.3.3 Comparative study

This section makes a comprehensive and exhaustive comparison of CCBAs and state-of-the-art approximate adders. All the adders described in 3.2.1 are represented: ESA [48], ETAII [18], ETBA [19], GCSA [49], ISA [33], ACA [50], ACAA [51], LOA [20] and pruned adders [35]. Exact truncated adders are also given for reference.

Figs. 3.11 and 3.12 (on pages 66-67) show dot plots of the accuracy-efficiency Pareto frontiers achievable by approximate adders at 3.3 GHz and 800 MHz, respectively. Error characteristics are measured on horizontal axes, compared by RE_{mean} on left subfigures and by RE_{max} on right ones. Circuit costs are measured on vertical axes, regarding energy consumption for top subfigures and normalized PDAP for bottom ones (PDAP is normalized to the exact 32-bit adder). The best designs are towards the bottom-left corners of subfigures.

Before going into the detailed results, the quality of the statistical estimations has been assessed by computing their variation over four samples of input vectors. The median RSD observed is in the order of 0.1 % for RE_{mean} and 10^{-4} % for RE_{max}. Only a handful of designs exhibits a high variation, over 5 %, mainly among high-accuracy ACA and ACAA adders. This is due to their large overlapping sub-adders inducing extremely low error rates [55]. The most critical case shows 28 % RSD for RE_{mean} or 4 % RSD for RE_{max}. In regard to the wide distribution of adder errors, covering many decades of the logarithmic scale, such variability does not challenge the viability of the study.

Mean relative error

Figs. 3.11a-b and 3.12a-b compare approximate adders in regard to the RE_{mean}. Note that those circuits do *not* correspond to the ones in Figs. 3.9 and 3.10, they show a different Pareto-optimal set solely considering RE_{mean}.

At high speed (Figs. 3.11a-b), truncated adder and LOA show the best circuit efficiency at a given RE_{mean}. On the contrary, ACA, ETBA and GCSA display the highest energy and PDAP costs. These are also unable to achieve high accuracy due to their direct dependency between sub-adder bit-width and error compensation, which makes it difficult to find an implementation optimizing both circuit and functionality. Outperformed by truncated adder and LOA at low accuracies, ISA and CCBA tend to catch up for RE_{mean} below 10^{-6} %, in terms of energy consumption as well as in PDAP.

At lower speed (Figs. 3.12a-b), the pruned adder displays moderate savings, with a closer trend to speculative circuits, particularly with the ISA. The latter shows a slightly better efficiency for RE_{mean} above 10^{-6} %, while pruning takes over below 10^{-6} %, validating the results obtained in [34]. LOA and truncated adder still outperform the state of the art, particularly for low RE_{mean}. However, at high accuracy, the CCBA surpasses both of them in terms of energy efficiency. This confirms the trend identified in section 3.3.2 that high-accuracy CCBA circuits exhibit energy savings rather than area reductions.

Maximum relative error

Figs. 3.11c-d and 3.12c-d compare approximate adders regarding their ability to limit the worst-case precision, i.e. $\mathrm{RE}_{\mathrm{max}}$. As this study focuses on lowering the relative errors, only adders able to limit $\mathrm{RE}_{\mathrm{max}}$ below 100 % have been represented: truncated adder, LOA, ETBA, GCSA, ISA and CCBA.

For both high-performance and low-power scenarios, the overall trend is similar. With its OR gates on the lower-part addition, the LOA cannot compensate errors, though it preserves the overall magnitude of the computed sum, leading to a 50 % worst-case error. Disregarding low-significance operand bits and underestimating the sum result, truncated adders have exactly a $\mathrm{RE}_{\mathrm{max}}$ of 100 %.

ETBA and GCSA show moderate savings, but their error compensation schemes are limited by their dependency between block sizes and error compensation, as for mean errors. The ISA, which has an optimized architecture dissociating critical path and error control, offers a larger range of $\mathrm{RE}_{\mathrm{max}}$ with more decent energy and PDAP reductions. For both 3.3 GHz and 0.8 GHz, the ISA offers the best trade-off for worst-case errors in the 7-50 % range.

The CCBA outperforms all other adders for $\mathrm{RE}_{\mathrm{max}}$ below 7 % for energy and PDAP in both low-power and high-speed cases. It is also the only adder capable of high accuracy, with $\mathrm{RE}_{\mathrm{max}}$ below 0.1 %, while still displaying considerable savings compared to exact adders. This is particularly emphasized for low-power circuits: at 10^{-3} % $\mathrm{RE}_{\mathrm{max}}$, the CCBA enables PDAP reductions up to 22 % or energy savings up to 36 %.

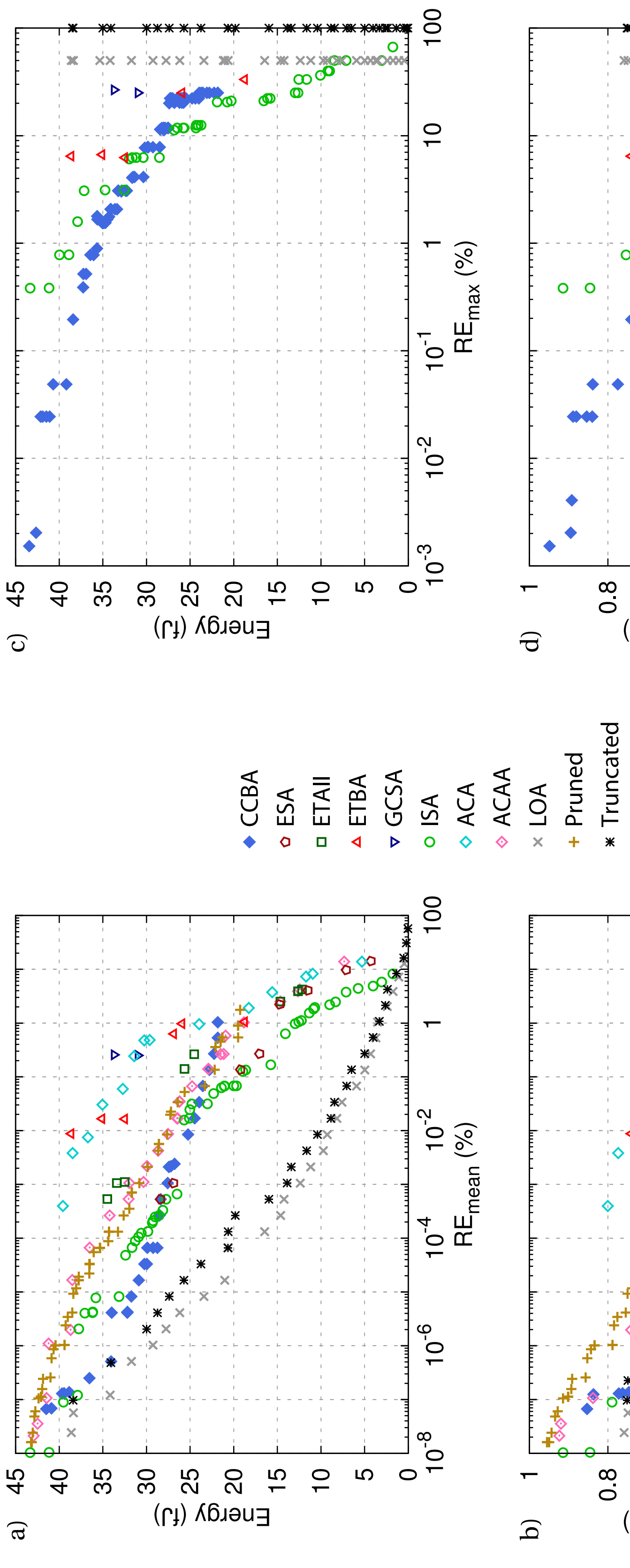


Figure 3.11 – Comparison of relative errors and circuit costs of approximate adders synthesized at 3.3 GHz in a 65 nm technology.

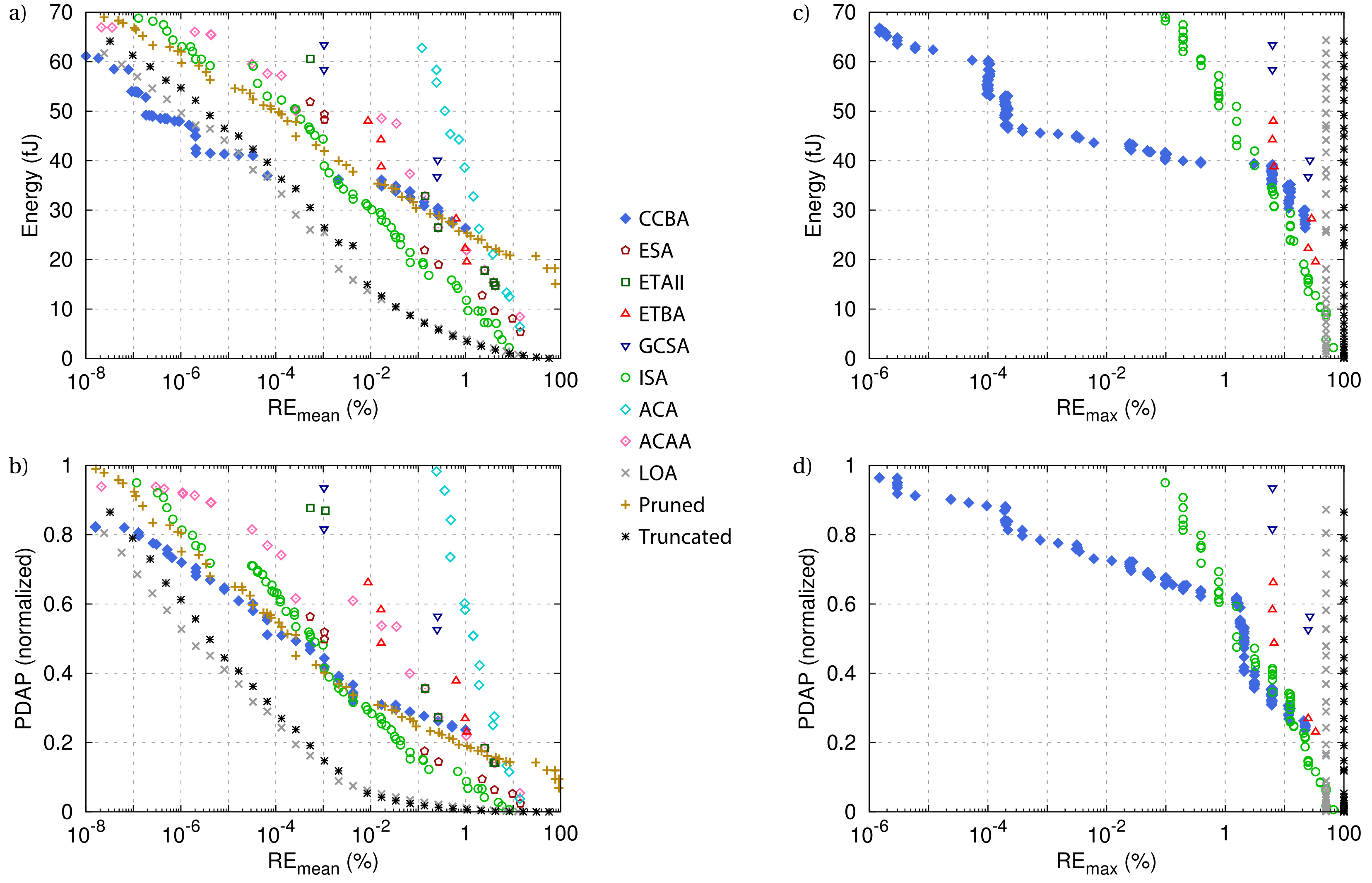


Figure 3.12 – Comparison of relative errors and circuit costs of approximate adders synthesized at 800 MHz in a 65 nm technology.

3.4 Conclusion

This chapter has introduced a novel concept to optimize approximate circuits by fabricating false timing paths, i.e. critical paths that can never be logically activated. Co-designing circuit timing together with functionality, this method proposes to monitor and cut critical paths to transform them into false paths. This allows to relax timing constraints, resulting in lower circuit costs or higher performance. Implementing the cuts on low-significance nets of arithmetic circuits can guarantee minimal behavioral changes, such as reduced precision or controllable arithmetic errors.

This technique has been applied to an approximate adder circuit, called the Carry Cut-Back Adder (CCBA), in which high-significance stages can cut the carry propagation chain at lower-significance positions, guaranteeing a high accuracy. This lightweight approach prevents the carry-chain activation, therefore relaxing timing constraints and strongly improving circuit efficiency. A design methodology has been presented in order to tune the accuracy, to optimize and correctly implement the timing constraints, and to reduce the design-space exploration.

An industry-oriented comparison for a 65 nm commercial CMOS technology has been carried out against 10 state-of-the-art approximate adders, including truncated exact adders. The CCBA architecture allows tuning the accuracy over almost seven orders of magnitude while exhibiting power-delay-area reductions up to 70 % for low-speed implementations. It greatly outperforms all other approximate adders for worst-case errors, and most of them for mean errors. Finally, it even improves upon exact truncated adders in terms of energy in the case of high-accuracy low-power designs. For a worst-case accuracy of 99.999 %, energy savings up to 36 % or power-delay-area reductions up to 22 % are demonstrated compared to low-power conventional designs.

This work has proven the considerable advantage of exploiting false timing paths on adder circuits. This novel approach could benefit larger arithmetic circuits, such as multipliers [57], as well as bigger datapaths, like CORDIC [58] and FPU [59]. Its extremely lightweight circuit implementation could help building highly-efficient configurable or precision-scalable hardware accelerators, with a better predictable and controllable impact on their functionality.

4 Approximate accelerators and applications

Chapters 2 and 3 have demonstrated the efficiency of approximate circuit techniques on arithmetic adders, enabling large power and area savings. However, an adder only represents a small fraction of the system it is placed in. Additionally, arbitrary thresholds of mean or maximum arithmetic errors do not give assurance on the real-world application quality. For these reason, it is necessary to assess these techniques on bigger datapaths and applications.

This chapter investigates the integration of the aforementioned ISA and GLP techniques within a Floating-Point Unit (FPU) accelerator utilized for an image-processing application.

In section 4.1, a novel approximate multiplier based on the ISA, the Inexact Speculative Multiplier (ISM), is presented. Then, ISA, ISM and GLP techniques are used to construct three approximate FPUs, integrated and taped-out in a 65 nm quad-core processor. Section 4.2 validates the use of these FPUs through a High-Dynamic Range (HDR) image tone-mapping application to run on-chip.

4.1 Approximate floating-point units

The Floating-Point Unit (FPU) is one of the most common building block integrated in computing systems. Used in many applications of Digital Signal Processing (DSP), graphics or high-performance workloads, FPUs are integrated in each commercial computer's CPU and in many microcontrollers. Each GPU is also traditionally containing thousands of them working in parallel.

FPUs feature a mathematically superior alternative to fixed-point computing with higher computational ability and flexibility, but with a much higher complexity, power consumption and circuit cost. This is due to their different number representation, based on the fact that any number can be written as:

$$(-1)^S \cdot M \cdot 2^E \tag{4.1}$$

where S is the sign bit, E is the exponent, and M is the mantissa (or significand), a fraction between 1 and 2. In the IEEE 754 standard convention, also called *binary32* or simply *float*, numbers are represented on 32 bits as on Fig. 4.1 [60].

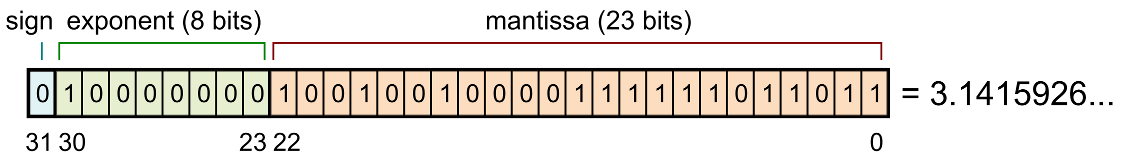


Figure 4.1 – Representation of floating-point numbers with the IEEE 754 standard [60].

Computing with such floating-point format is different than with conventional fixed-point circuits, and also differs depending of the desired operation.

Similarly to additions using scientific notation, to add or subtract two floating-point numbers, their exponents must first be aligned by shifting the mantissa of the smallest-exponent number. After the exponents are equalized, a conventional fixed-point addition or subtraction can be performed on the two mantissas. The final step is to shift the mantissa and adjust the exponent so that the number is in normalized form.

To multiply or divide two floating-point numbers, both exponents and mantissas are processed separately. Unlike additions and subtractions, there is no need to align the exponents. The mantissas are directly multiplied or divided, while the exponents are added or subtracted. The final step is also to normalize the number if needed.

4.1.1 Inexact speculative multiplier (ISM)

FPUs require both fixed-point adders and multipliers for the aforementioned mantissa operations. But multiplier circuits have a much higher area, power consumption and delay than their adder counterparts. Yet, few works in literature have addressed the case of speculative multiplication. This section briefly introduces the Inexact Speculative Multiplier (ISM) [59], a new approximate multiplier circuit derived from error-compensated speculative architectures, and that could be used in FPU circuits.

Conventional parallel multiplier architectures are based on computing a set of partial products and summing them together. To be integrated in high-performance blocks such as a FPU, this process is generally pipelined over multiple stages. As depicted in Fig. 4.2, the proposed ISM is based on a two-stage architecture. The first stage, a Partial Product Multiplier (PPM), generates and merges partial products with a compressor tree into two partial sums. The second stage, dedicated to the final accumulation, uses an ISA to add the two partial sums in a speculative way. While the compression tree mainly contributes to the multiplier area, the addition stage is the slowest element of the multiplier. Thus, even the slightest approximation by speculation can significantly increase the multiplier performance.

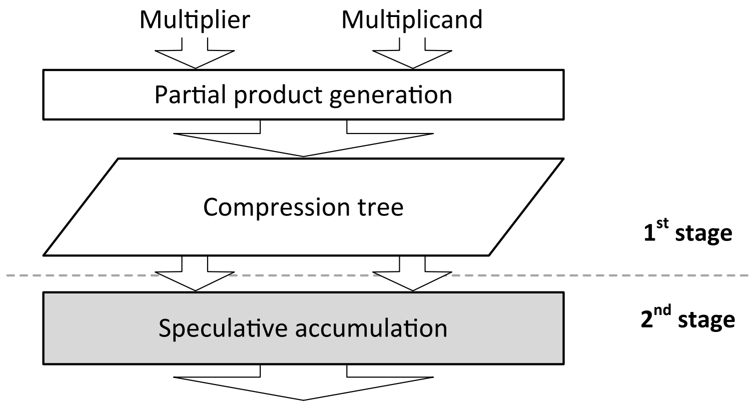


Figure 4.2 – Architecture of the Inexact Speculative Multiplier (ISM).

In the case of unsigned operation, sizing the different speculative and error-compensation parameters of the ISA in the accumulation stage directly allows to trade worst-case and average errors as in [33]. In the case of two's-complement signed multiplication, small numbers typically contain large series of ones or zeros depending of their sign. To ensure that the speculation does not break the sign of the expected result, a dynamic carry guess of the inverse of the expected sign is required on all speculative paths to avoid any sign error, i.e. a XNOR of the two operand's MSBs. Other ISA parameters are selected in the same approach as for unsigned operation.

Note that with the same architecture, future works could benefit from the CCBA, rather than the ISA, to build a far more effective and accurate *Carry Cut-Back Multiplier (CCBM)* [61].

4.1.2 Approximate FPU architecture

Floating-point computations are more complex than fixed-point ones, as mantissa and exponent must be computed separately with different hardware, and include handling of normalizations and exceptions. The architecture used for the approximate FPU is shown on Fig. 4.3 [59], it handles additions, subtractions, multiplications, as well as *float-to-int* and *int-to-float* conversions.

Only mantissa computations are approximated to avoid errors in the exponent, which would have a dramatic impact on the results. The two approximate circuit design techniques described in chapter 2, namely Gate-Level Pruning [35] and Inexact Speculative Adder [33], as well as their combination, have been adapted and fitted in the FPU mantissa computations.

As depicted in Fig. 4.4, a chip has been realized based on the PULP architecture [62] with 4 Or10n cores, 16 kB of L2 memory, 16 kB of Tightly Coupled Data Memory (TCDM) organized into 8 banks and 4 kB of instruction cache. Each core has a dedicated FPU capable of additions, subtractions and multiplications with 2 cycles of latency. One of them is compliant to the IEEE 754 single-precision standard while the three others are approximate variations of it. The chip has been fabricated with UMC 65 nm standard process technology and has been designed to run at a maximum frequency of 500 MHz with a power supply of 1.2 V.

All the 4 FPUs share the same architecture, the only difference is the replacement of the original mantissa adder and mantissa multiplier by approximate versions of them. In a *pruned* FPU, Gate-Level Pruning has been used to generate the approximate adder and multiplier. The Inexact Speculative Adder and the new Inexact Speculative Multiplier have been implemented in a *speculative* FPU. At last, in a *mixed* FPU, both speculation and pruning techniques have been combined to obtain even higher power and area savings. To ensure a minimal guaranteed

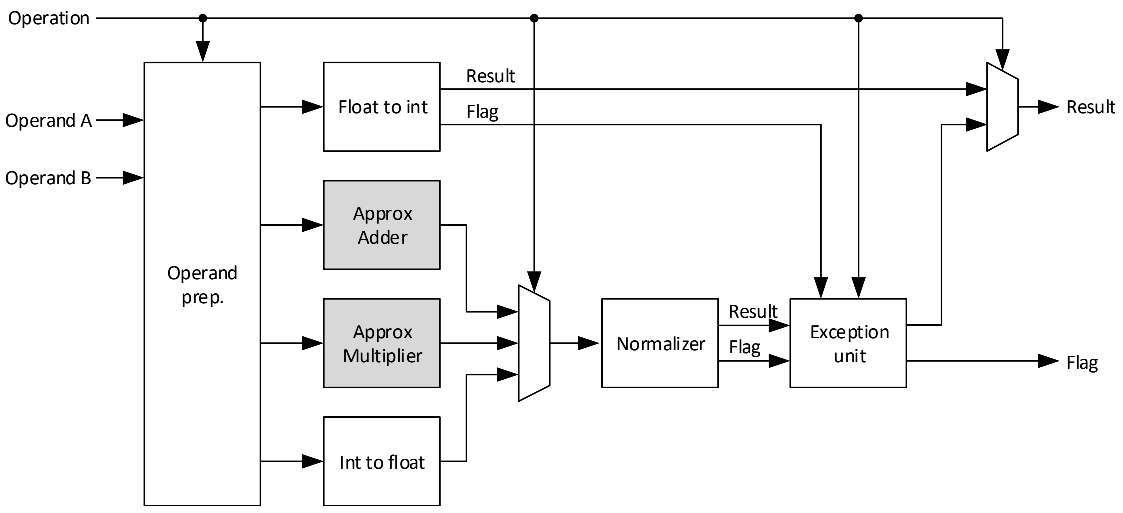


Figure 4.3 – Architecture of the approximate FPU.

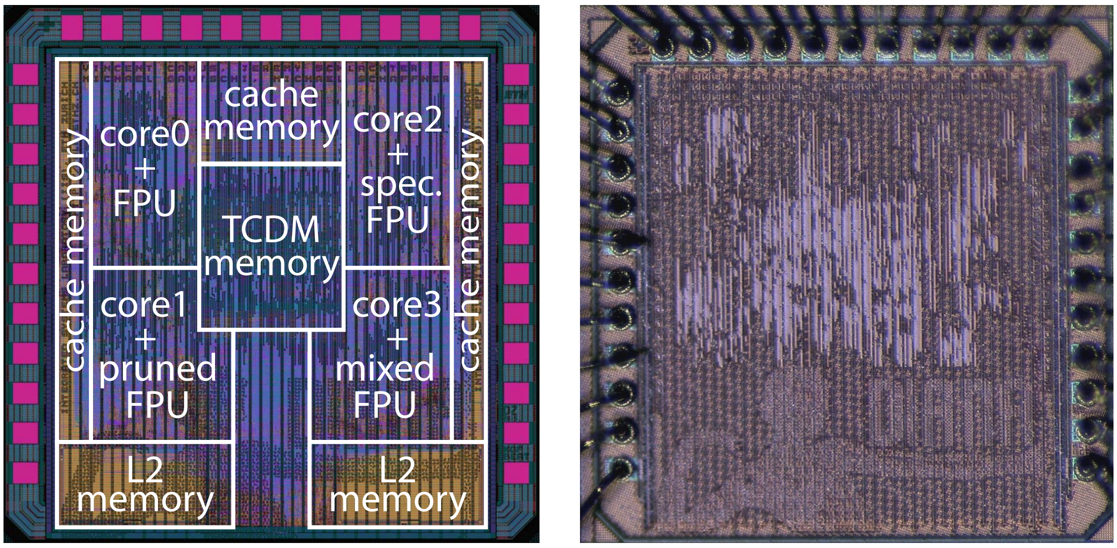


Figure 4.4 – Floorplan and die microphotograph of the chip. Die size is 1.56 mm^2.

precision and in order to better compare the three techniques, all the approximate FPUs have been chosen to maintain exactly 10 bits of exact arithmetic computation.

4.1.3 Measurement and results

Each of the approximate FPUs has been fed with a set of twenty million random uniformly distributed inputs to get a statistical estimation of the approximate behavior. The hardware used for floating-point additions and subtractions is different from the one used for multiplications and moreover, is implemented with different speculative circuits and pruning levels. For this reason, FP additions and multiplication have been characterized independently. The metrics used to characterize approximate FPUs are the same than the ones defined in chapter 2.

Table 4.1 summarizes all the error characterizations. It can be noticed that the RE_{RMS} remains low for all the floating-point operations. However, as soon as pruning is involved, the RE_{MAX} becomes 1 (100 %). Indeed, in the pruning process some LSBs are set to a fixed value. For instance, if the LSB of an adder is set to logic '0' the operation $1 + 1 = 0$, whereby RE_{MAX} is 1.

Table 4.1 – Error characteristics of the approximate FPUs.

FPU	Addition / subtration		Multiplication	
	RE_{RMS}	RE_{MAX}	RE_{RMS}	RE_{MAX}
Pruned	1.15 E-3	1	1.4 E-3	1
Speculative	2.36 E-6	5.69 E-3	2.6 E-5	1.17 E-1
Mixed	2.27 E-4	1	1.4 E-3	1

Speculative circuits show much lower errors than other approximate techniques, and are particularly good in ensuring a bounded error. Error compensation and FPU rounding unit both share the same philosophy that a few bits in one direction are equivalent to a single one at adjacent position. For instance, the FPU rounding would approximate the sequence '0.111' by '1.000', while the speculative error '0.000' instead of '1.000' would be compensated by '0.111'. Note that the speculative FPU, containing the full functional adders and multipliers (with disconnected carry chains), could easily and cost-effectively be implemented with an exact computation mode.

The total power consumption of the chip has been measured by running a vector multiplication and addition benchmark, one core at a time. In order to be able to extract the consumption of a single FPU, i.e. without the overhead of the cores and memories, a second set of power measurement has been performed by running the same benchmark with all the assembly floating-point add and multiply instructions replaced by *No Operations* (NOPs). Measurements were done at 1.2 V and at room temperature. This test has been performed over 9 chips and with frequencies ranging from 100 MHz to 300 MHz (the chips ran up to 500 MHz, but the power measurement tools were not accurate above 300 MHz).

Measurements shown in Table 4.2 and Fig. 4.5 show that the pruned FPU achieves 15 % power and 11 % area savings, whereas the speculative FPU enables 12 % power and 14 % area savings. Thanks to the switching activity criteria, pruning generally achieves better power reduction than speculation but with higher errors. Despite speculation requires extra hardware for carry generation and error compensation, it strongly relaxes the timing constraint (at synthesis, it could almost fit 1 clock cycle instead of 2), allowing to simplify the architecture. Combining pruning and speculation leads to 27 % power, 36 % area and 53 % Power-Area Product (PAP) savings thanks to their radically different circuit approaches, and with similar error levels to pruning.

Table 4.2 – Power, area and power-area product of the integrated FPUs.

FPU	Power (mW)	Area (μm^2)	Power-area product ($W \cdot \mu m^2$)
Exact	2.81	13 200	37.1
Pruned	2.40	11 850	28.4
Speculative	2.48	10 070	25.0
Mixed	2.07	8 550	17.7

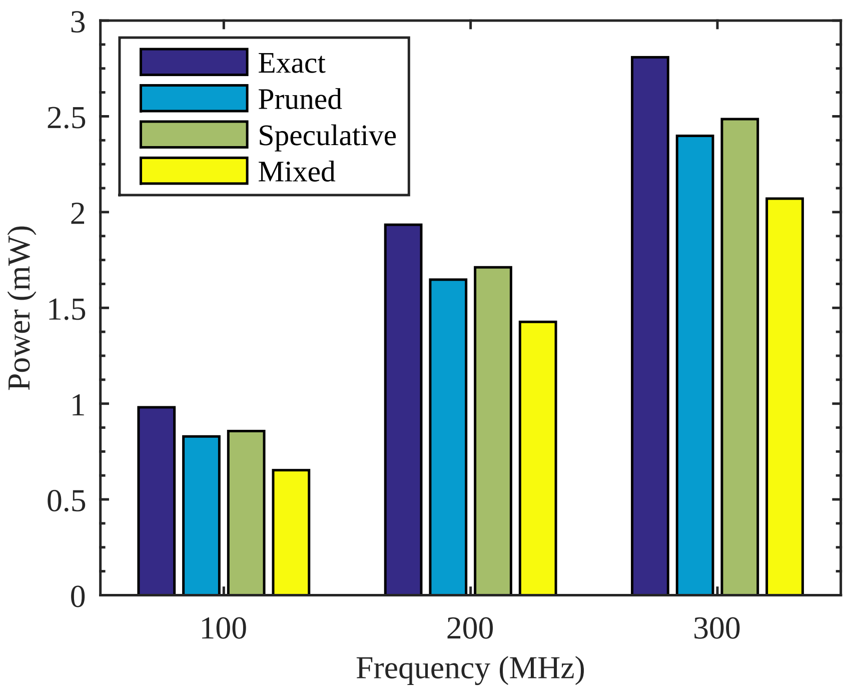


Figure 4.5 – Measured power consumptions of the 4 FPUs at different frequencies.

4.2 Application to HDR image tone-mapping

The use of those FPUs have been validated by programming a High Dynamic Range (HDR) image tone-mapping application to run on-chip [59]. Carried by the high demand for consumer digital cameras integrated in phones, tablets or Internet-of-Things (IoT) devices, HDR tone-mapping is an excellent application to evaluate and validate the use of approximate hardware in the FPU by comparing the end-user impact of the image quality loss.

Extensively using floating-point computations, tone mapping is one of the most common treatments used on HDR images. Those images are characterized by a very high ratio between the luminance of the brightest and the darkest pixel and are able to represent a wide range of luminance values. Tone mapping is widely used to match the dynamic range of the captured image with the one of the device where it is going to be displayed, which is usually limited. Tone mapping allows to optimize the contrast, which is directly related to the dynamic range, while still preserving the high resolution of the image and its color appearance. In this way, dark zones will become brighter while bright zones will become darker.

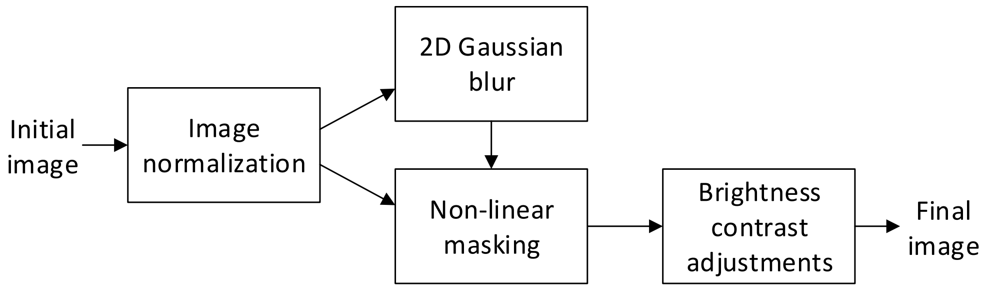


Figure 4.6 – Tone-mapping algorithm.

4.2.1 Tone-mapping algorithm

In this work, a tone-mapping application using non-linear masking algorithm [63] has been implemented in C and compiled to be executable on the realized chip. This method has been chosen as it is less computationally-intensive than other algorithms and particularly because it minimizes the use of floating-point divisions as those have not been implemented.

As depicted in Fig. 4.6, the implemented tone-mapping algorithm consists in multiple operations:

1. Image normalization: each pixel of the initial image is normalized with respect to the maximum pixel value. This step therefore multiplies each of the pixels by a constant value.

2. Gaussian blur: a blurred monochrome image is generated by convolution of a Gaussian

filter onto the normalized image. Pixels of this mask image serve as local brightness values. This step involves thousands of multiply and accumulate operations.

3. Non-linear masking: the main tone-mapping operation applies a pixel-by-pixel gamma correction using a power coefficient of the blurred-image pixel. The floating-point power function is built out of base-2 exponent and logarithm functions, themselves generated through an input-range reduction followed by a 6^{th}-to-8^{th} degree Taylor series, thus using a large number of floating point multiply-accumulate operations for each pixel.
4. Brightness and contrast adjustments: a fixed brightness and contrast adjustment step is added to further optimize the image.

The very high number of floating-point operations should be a good indicator of the robustness of the approximate FPU since the propagation and accumulation of errors could have a significant impact on the final image.

Note that with the low on-chip memory, the largest image that could be tone-mapped on the chip itself was around 40×40 pixels. In order to process larger images, a netlist-to-C conversion script was written to be able to easily and directly simulate applications and algorithms using approximate hardware.

4.2.2 Results

Fig. 4.7a shows an HDR image before tone-mapping, the landscape is not visible at all and the sun is too bright, hiding part of the clouds. Fig. 4.7b shows the same image after tone-mapping and brightness-contrast correction computed with the exact FPU, the entire scenery is now discernible. Fig. 4.7c show the tone-mapped images computed with the pruned, the speculative and the mixed FPU, respectively, with Peak Signal-to-Noise Ratio (PSNR) ranging from 76.4 dB using the pruned FPU to 127.3 dB using the speculative FPU, in line with the error characterizations. There is absolutely no difference discernible by the human eye between the picture processed by the exact FPU and the ones processed by the approximate FPUs.

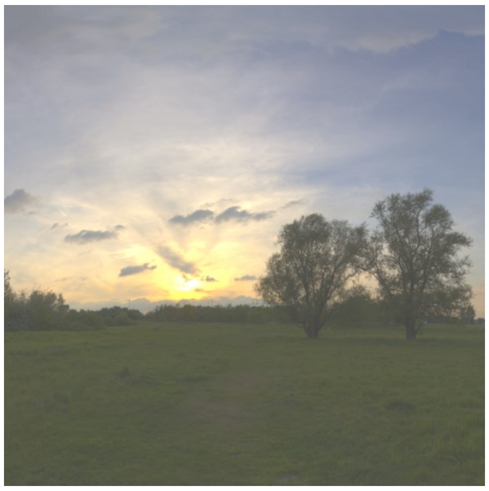
To further investigate the quality loss, the approximate tone-mapped images have been compared to the exact one, pixel by pixel and color by color. The *Pixel Value Difference* (*PVD*) has been used to show the error on each individual pixel and color component between the image processed with an approximate FPU and the one processed with the exact FPU. It is simply defined as the arithmetic difference between the approximate pixel value and the exact pixel value.

Fig. 4.8 plots the *PVD* distribution for each of the approximate tone-mapped images. The speculative FPU produces very small errors of specific magnitudes due to the specific positions of the cuts in the carry chains. On the other hand, errors produced by the pruned and mixed FPUs are spread by two orders of magnitude more than for the speculative FPU, but large

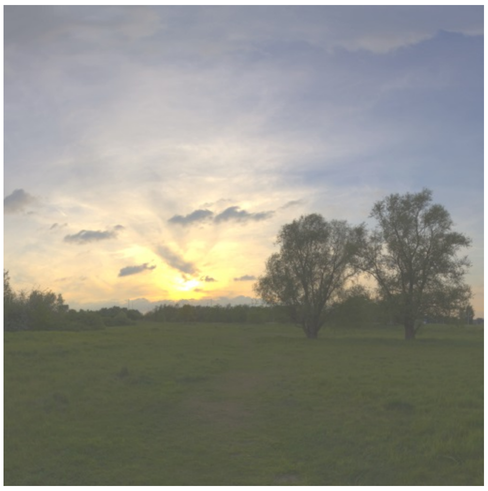
(a) Original image

(b) Exact

(c) Pruned (76.4 dB)



(d) Speculative (127.3 dB)



(e) Mixed (90.0 dB)

Figure 4.7 – Initial (a) and tone-mapped (b-e) images. (c-e) are obtained with approximate FPUs and indicated with their PSNR quality. Image size is 512×512 pixels.

errors remain rare. The error distribution of the tone-mapped image processed by the mixed FPU combines the continuous distribution as with the pruned FPU and high error-count around zero as with the speculative FPU.

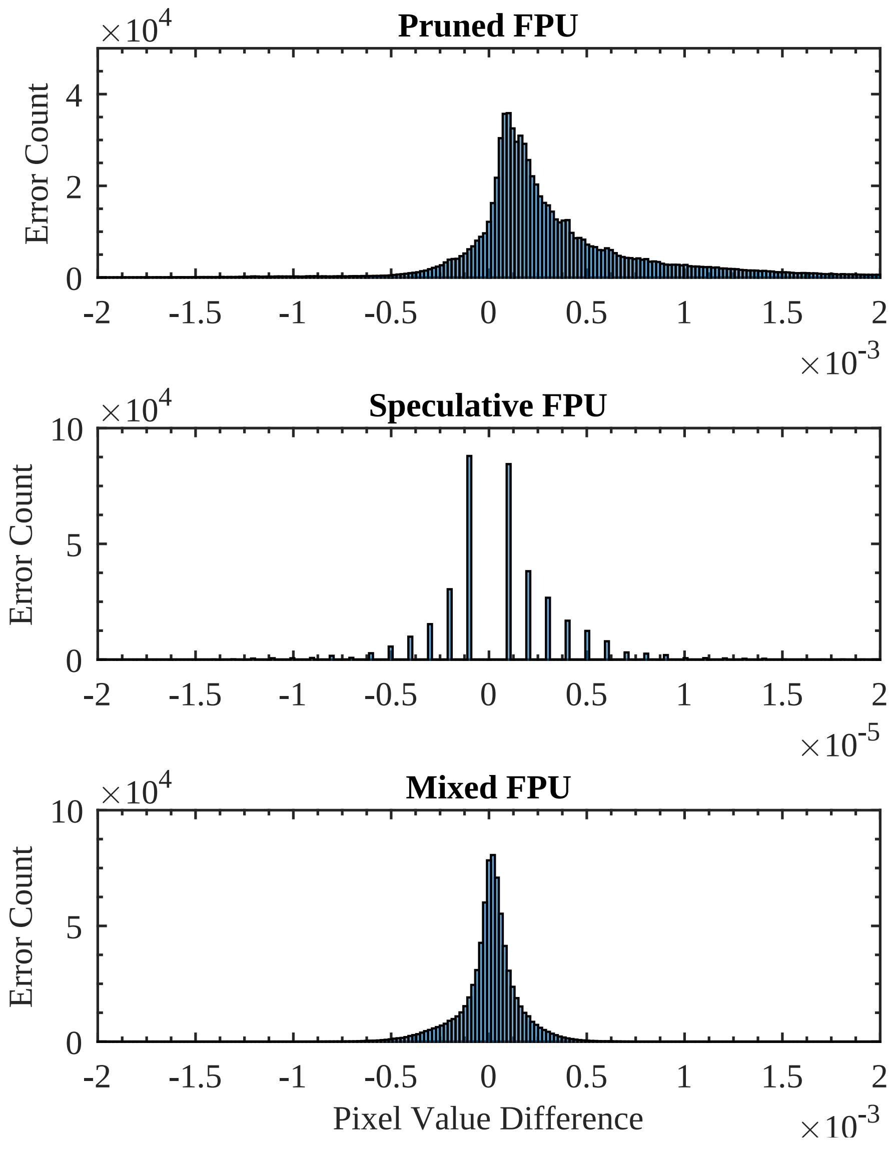


Figure 4.8 – Error distributions of the 3 images tone mapped with the approximate FPUs. X-axis of the speculative and Y-axis of the pruned FPU are scaled differently.

4.3 Conclusion and perspectives

The ISM, a novel speculative multiplier based on the ISA, has been presented. By combining GLP, ISA and ISM techniques, three approximate single-precision FPUs have been designed by approximating the mantissa adders and multipliers. These FPUs have been integrated in a 65 nm CMOS process within a quad-core PULP processor in order to demonstrate their functionality in a computing system. Measurements have shown 15 % power and 11 % area savings for the pruned FPU and 12 % power and 14 % area savings for the speculative FPU. Producing different types of errors, pruning and speculation can be combined to achieve 27 % power, 36 % area and 53 % power-area product reductions.

The use of those approximate FPUs have been validated by running a floating-point-intensive tone-mapping algorithm on HDR images. Results have shown no visible quality loss, with image PSNR ranging from 76.4 dB using the pruned FPU to 127.3 dB using the speculative FPU. Additional error measurements have confirmed that each technique produces a specific error distribution with errors remaining small and centered around zero.

This chapter has proven that approximate arithmetic techniques enable decent savings on the entire FPU accelerators with negligible quality degradation, particularly for techniques bounding the error values. However, as the study of chapter 3 highlights, approximate adders are outperformed by truncation at high error levels. It would therefore be interesting to first correctly size circuit bitwidths before applying such approximate techniques as fine-tuning. Such strategy would also alleviate memory storage and movement costs.

5 Precision-scalable MAC units for neural-network processing

The trend for deep learning has come with an enormous computational need for billions of Multiply-Accumulate (MAC) operations per inference. Fortunately, precision-scalable circuits with reduced precision, another dimension of approximate computing, have demonstrated significant benefits on it, paving the way towards processing in mobile devices and IoT nodes.

In this chapter, run-time configurable MAC units from ISSCC 2017 and 2018 are surveyed, implemented, and compared objectively under diverse precision scenarios. A new multi-bit MAC architecture is also presented and compared. This work analyzes the impact of scalability and compares the different MAC units aiming to understand the optimal architectures to reduce computation costs in neural-network processing.

Section 5.2 introduces general considerations about scaling precision and details the deducted methodology to compare the different designs with impartiality. Section 5.3 surveys the considered architectures. Finally, Section 5.4 analyzes the energy breakdown of scalability individually and presents a comparison across all approaches.

5.1 Introduction

The current trend for deep-learning applications, such as image classification and speech recognition, has come with an enormous computational need. Indeed, state-of-the-art Deep Neural Networks (DNN) require billions of Multiply-Accumulate (MAC) operations, the fundamental component of their convolution layers, as well as fetching of millions of network parameters (weights) and feature maps (activations). Many hardware improvements have recently been proposed, innovating at different abstraction levels to solve both memory and computational bottlenecks [64, 65].

Exploiting reduced precision has demonstrated huge benefits with no or negligible impact on the network accuracy, paving the way towards embedded DNN processing in mobile devices and IoT nodes. It has led to a new trend for precision-scalable neural processors to minimize energy at target performance without giving up flexibility. Recent papers have introduced run-time configurable MAC architectures optimized for deep learning, built either with high parallelization capabilities [26, 66] or bit-serial approaches [67, 68].

However, it is difficult to assess the efficiency of these architectures for two reasons:

1. They have been implemented using diverse process technologies, bitwidths and scalability levels, and sometimes specific features. They also and have been integrated within quite different systems, with various memory sizes and interfaces, or targeting different trade-offs. This makes it impossible to extract the precise cost of the MAC or its contribution to the system.

2. While nearly all these works detail the relative efficiency breakdown of scaling down precision in their design, few evaluate their absolute performance against a baseline design, without configuration nor parallelization overheads. This makes it hard to compare and select the right architecture for a chosen application.

In this chapter, the most prominent precision-scalable MAC accelerators are surveyed, implemented, and compared in a fair way under different precision scenarios. All circuits are synthesized in a 28 nm CMOS process with bitwidth precision ranging from 8 bits down to 2 bits using both hardware scalability features and data gating. This study, published in [69], analyzes the impact of precision scalability on energy efficiency, especially its unavoidable energy overhead due to the introduced configurability, and gives a global picture of energy and throughput-per-area capabilities of the different architectures.

Table 5.1 – Reviewed designs and their scalability features.

MAC architecture	**Weight scalability**	**Activation scalability**
Conventional [70]	Data gating	Data gating
DVAFS (Envision [26])	Symmetric subword parallel	
Divide-and-conquer (DNPU [66])	Subword parallel	Data gating
Bit serial (UNPU [67])	Serial	Data gating
Multi-bit serial (proposed)	Serial	Data gating

5.2 Considerations and methodology

5.2.1 Scalability by design and by data gating

Precision-scalable MACs have gained interest following the observation that the optimum bit-width for a DNN strongly varies from one application to another, and even across the different layers of a single DNN [70].

The easiest way to scale precision in arithmetic circuits is to use data gating, i.e. reducing precision by simply zeroing operands' LSBs to avoid unnecessary toggling in the circuit. This only introduces marginal overhead in the MAC periphery, leaving the MAC untouched. In contrast, precision-scalable architectures embed additional features *by design*, inside the MAC, to increase parallelism or to clock-gate unused parts of the circuit. However, these features have a cost. First, they imply some control circuitry, outside of the MAC unit, which can fortunately be shared among an array of processing elements. More importantly, they induce hardware overhead in each MAC, leading to increased area, delay and power consumption. This is why neural processing elements often support a limited set of precisions by design, such as 2b, 4b and 8b inputs.

Two schools of thought exist for scaling DNN computations. Research originally aimed at *weight-only* precision scaling rather than activation, mainly to decrease model sizes. But late works started to quantize both weights and activations, which can be simplified as a *symmetric* scaling scenario, where both weights and activations are scaled at the same precision.

5.2.2 Scope and methodology of the study

This work evaluates the precision-scalable MAC designs listed in Table 5.1, most of them presented at ISSCC 2017 and 2018. They are studied for 2b, 4b and 8b precisions, these values being commonly found in literature. Both symmetric and weight-only scaling scenarios are considered.

Multiple implementations of each architecture are made, varying their *level of scalability* starting from their 8b baseline. For instance, a 1-level scalable design allows to scale from 8b down to 4b by design, the 2b mode carried out by data gating over the 4b mode. A 2-level scalable design directly allows to scale down to 4b and 2b by design. Circuits are implemented in such a way that for any precision scenario, the accumulator bitwidth is exactly 4b larger than the multiplication range.

All architectures are generated from SystemVerilog descriptions with signed-weight and unsigned-activation representations. They are built at the highest level of abstraction without individual optimization (such as LUT-based acceleration [66]). All circuits are synthesized with identical compiler options following a multi-mode timing optimization to find the best delay trade-off for all scaling scenarios. Note that the timing of mode-control signals is not optimized at all, as it is assumed that the same precision mode is used for many cycles in a row (for instance for a full CNN layer). Finally, a design-space exploration is performed across multiple frequency targets to investigate the efficiency at all the achievable throughputs.

For each circuit and each precision mode, Mentor Questa is used to generate Value Change Dump (VCD) switching information from a timed gate-level simulation at best clock period. The simulation for each mode consists out of 10,000 operations with accumulator reset every 50 operations, using Gaussian-distributed random inputs with distributions observed in quantized CNNs trained for CIFAR-10. VCD files are then fed back to Cadence for power estimation. This latter method is very important as the gated or unused logic, depending of the precision used, leads to large differences in dynamic power. Power and area of input registers and overheads that can be shared among multiple processing elements (e.g. control logic or finite state machines) are not included in the reported results.

5.3 Multiply-accumulate units

5.3.1 Data-gated conventional MAC

The baseline of this study is a data-gated conventional MAC unit as in [70]. When scaled precision is used, only the MSBs are used for computation while the LSBs are kept at zero. Hence, the switching activity is reduced. As the critical path going only through the MSBs is shorter, the frequency can dynamically be increased or the supply voltage can be lowered at equal throughput.

This is illustrated in Fig. 5.1, showing from left to right the use with full precision, 4-bit symmetric scaling, and 2-bit weight-only scaling. When using scaled-precision computations, some parts of the multiplier and accumulator logic keep unused, as shown by the grey stripes. However, as this MAC does embed any configurability feature, such as selective clock gating, all registers stay clocked despite no data reach their LSBs.

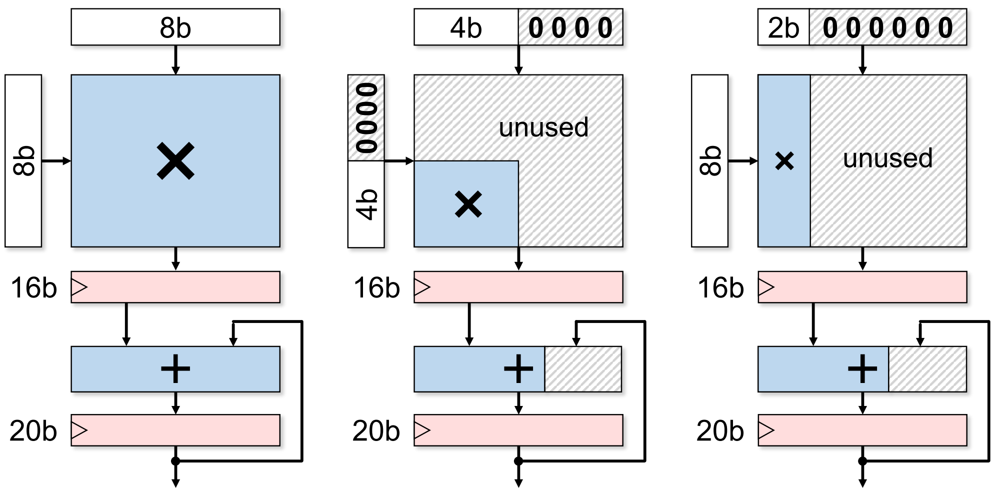


Figure 5.1 – Data-gated conventional MAC with an example of symmetric 4b×4b scaling (middle) or weight-only 2b×8b scaling (right).

5.3.2 DVAFS MAC

Dynamic Voltage-Accuracy-Frequency Scaling (DVAFS) was introduced by Moons *et. al* [26]. Built on an array multiplier circuit, the DVAFS MAC reuses full-adder cells that are inactive at scaled precision. This scales together weight and activation with symmetric subword parallelism, as shown on Fig. 5.2. Similar to the data-gated conventional multiplier, the shortened critical path permits an increased clock frequency. Note that processing at full precision with DVAFS comes at a slight energy and area penalty due to the complex configuration and sign-compensation logic overheads, and the larger registers required for extra accumulation bits.

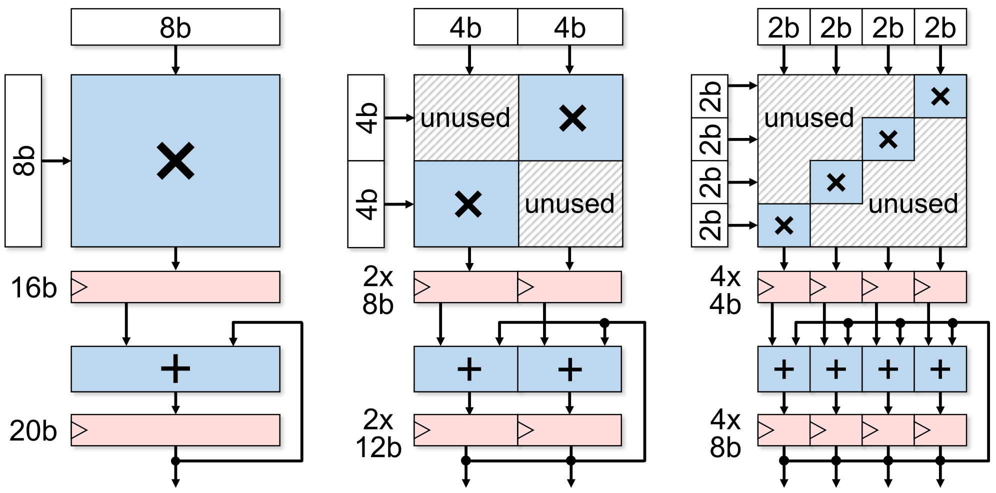


Figure 5.2 – Symmetric precision scaling in a DVAFS MAC configured for either one 8b×8b, two 4b×4b, or four 2b×2b operations per cycle.

5.3.3 Divide-and-conquer strategy

The Deep Neural Processing Unit (DNPU), introduced by Shin *et. al* [66], uses a reconfigurable multiplier with a *divide-and-conquer* (D&C) approach on one operand. As shown on Fig. 5.3, the full-precision multiplier is built out of shifted binary additions of partial products. By this manner, intermediate sums directly correspond to scaled multiplication results.

Only one operand is subword parallel, doubling the number of operations per cycle for each scalability level. The other operand is common to all subword computations, restricting its use to repeated operand, but allowing parallelization when one operand has to stay at full precision. Scaling of the second operand is always possible by data gating.

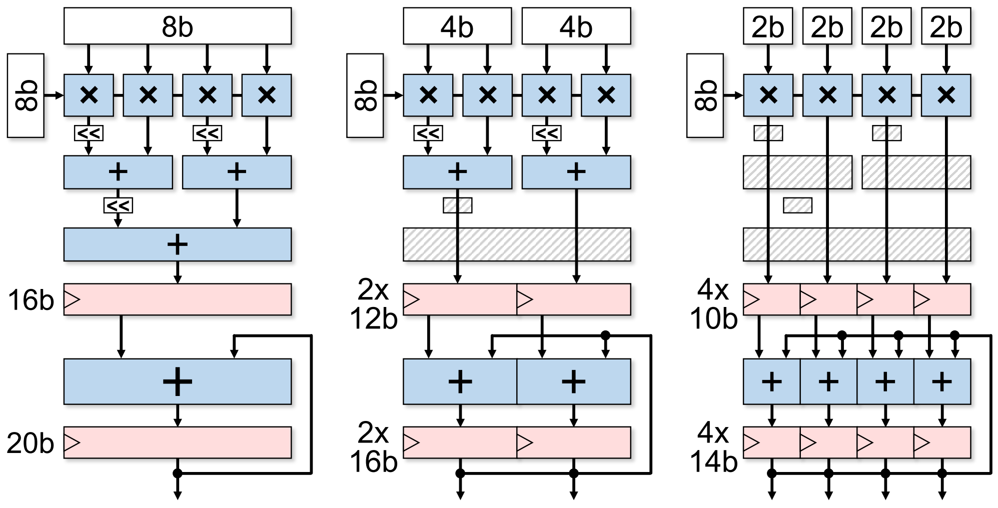


Figure 5.3 – Weight-only precision scaling in a D&C MAC configured for either one 8b×8b, two 4b×8b, or four 2b×8b operations per cycle.

5.3.4 Bit serial design

Bit-serial designs have recently gained attention with both the Unified Neural Processing Unit (UNPU) by Lee *et. al* [67] and the QUEST log-quantized 3D-stacked inference engine by Ueyoshi *et. al* [68]. Indeed, bit-serial operand feeding implicitly allows fully-variable bit precision. Considered in this study, the UNPU bit-serial MAC receives weights through 1-bit iterations while activations are sent in a parallel manner, as illustrated in Fig. 5.4, making this design 2-level scalable. Scaling activation is possible by data gating.

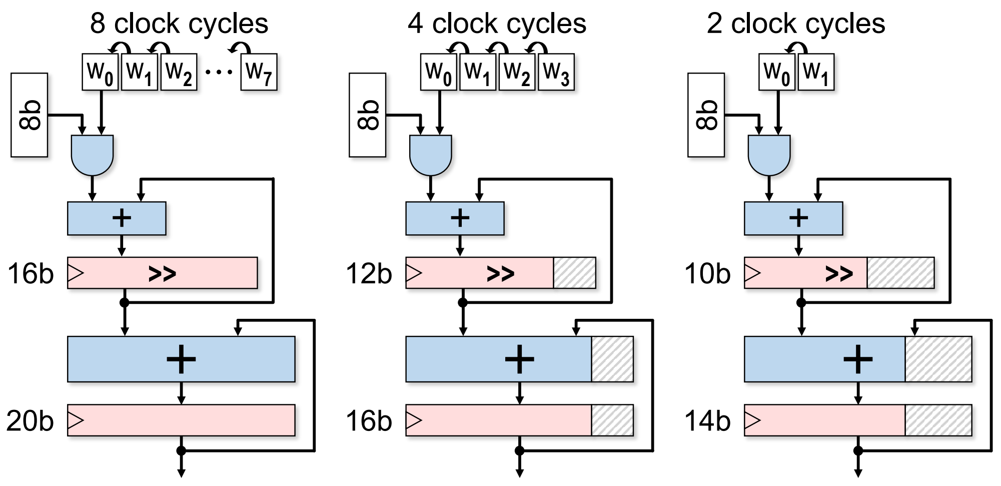


Figure 5.4 – Weight-only precision scaling in a bit-serial MAC configured for either 8b×8b, 4b×8b, or 2b×8b operations.

5.3.5 Multi-bit serial design

This work extends the original bit-serial concept by introducing *multi-bit serial* designs. Fig. 5.5 shows the example of a 4-bit serial MAC, where weights are fed 4 bits at a time. This scheme requires only 2 clock cycles for an 8-bit computation, hence reducing the energy consumed in the clock tree and registers. Lower precision can be obtained by gating the unnecessary bits. This survey includes both 2-bit and 4-bit serial MACs. Note that the 2-bit serial MAC is 2-level scalable by design (8, 4 and 2-bit weight inputs), but the 4-bit serial MAC is only 1-level scalable by design (8 and 4-bit weight inputs, lower precisions being implementable by data gating the 4-bit weight inputs).

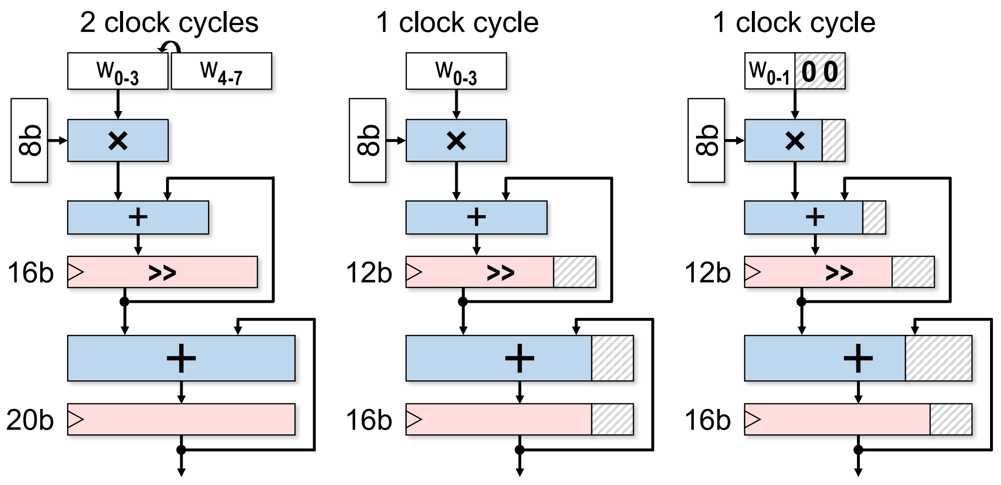


Figure 5.5 – Weight-only precision scaling in a 4-bit serial MAC configured for either 8b×8b, 4b×8b, or 2b×8b (by data-gating the 4b×8b) operations.

5.4 Results and comparative study

5.4.1 Precision-scaling energy breakdown

Figs. 5.6-5.8 show the breakdown of energy per operation when scaling precision for each type of architecture, selecting the circuit with the lowest energy per operation (without consideration for throughput or area). The left subfigures show symmetric scaling scenarios while the right ones show weight-only scaling. Energy values are normalized to the full-precision data-gated conventional MAC drawn with a solid black line.

Energy overhead at full precision

Processing at full precision with scalable designs comes with some energy penalty. 8-bit computations with DVAFS (Fig. 5.6) consume 13 % and 28 % more energy than data gating for embedding 1 and 2 levels of scalability, respectively. For D&C circuits (Fig. 5.7), these overheads are roughly similar with respectively 18 % and 26 % extra energy per 8-bit operation.

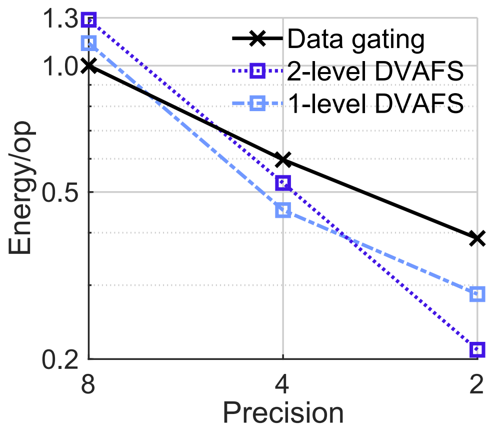
Despite their efficient use of area, serial designs (Fig. 5.8) require much more energy at high precisions due to their need for several clock cycles per computation, diluting the power into the clock tree and the multiplication registers. A full-precision operation in the bit-serial MAC consumes more than 3 times as much as the conventional MAC unit.

These overheads are worse than reported in the original papers. Nevertheless, many things differ in this study compared to each architecture's reference, such as design implementation, benchmark and baseline reference. In [67] for instance, bit-serial and baseline conventional MACs are compared at a single synthesized and operating point of 200 MHz (despite these two antagonistic architectures should have very different optima), for which the bit-serial architecture might effectively be best trade-off. However, our study explores a very large design space of performance targets only to show a single energy-optimal solution (over 5000 synthesized circuits solely for the serial MAC).

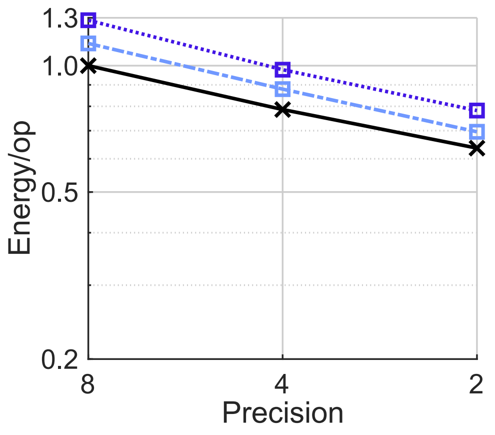
Reassuringly, the proposed multi-bit designs come at a lower energy penalty. The 2-bit serial MAC consumes a little above 2 times the baseline energy, and the 4-bit serial MAC reduces the energy overhead to only 57 %.

Energy and precision scaling

Reducing precision of both weights and activations (left subfigures) with data gating already leads to energy savings in a linear way with respect to the bit precision (40 % in 4-bit mode and 61 % in 2-bit mode). In comparison, precision-scalable MACs show a steeper slope, meaning that they save energy in a superlinear way with bit precision. It is although insufficient for the 1-bit serial design to compensate its significant energy penalty.

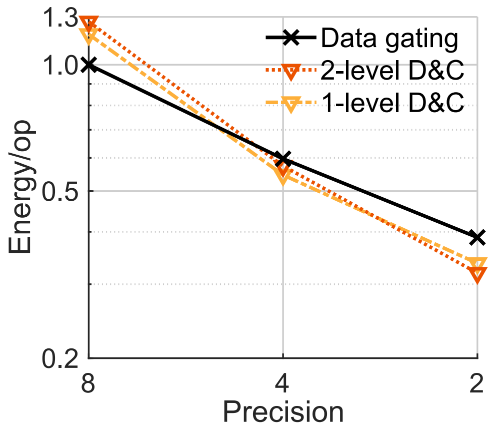


(a) Symmetric scaling

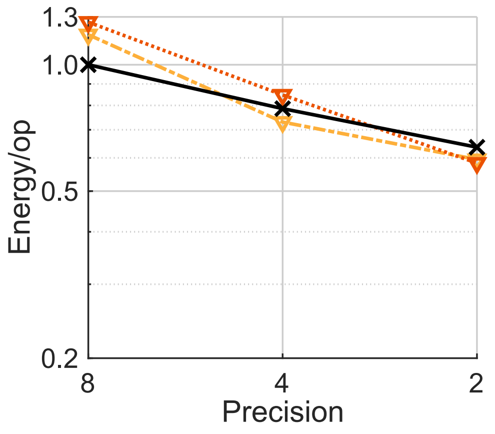


(b) Weight-only scaling

Figure 5.6 – Normalized energy/op with precision scaling in a DVAFS MAC.

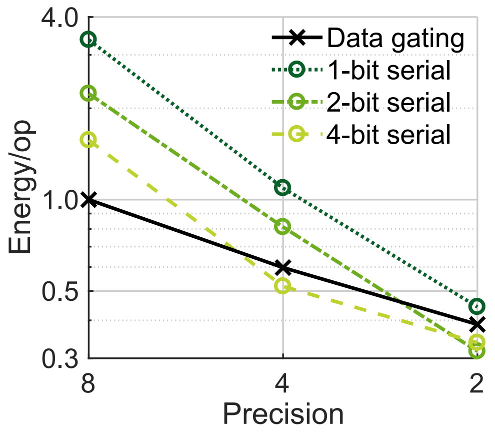


(a) Symmetric scaling

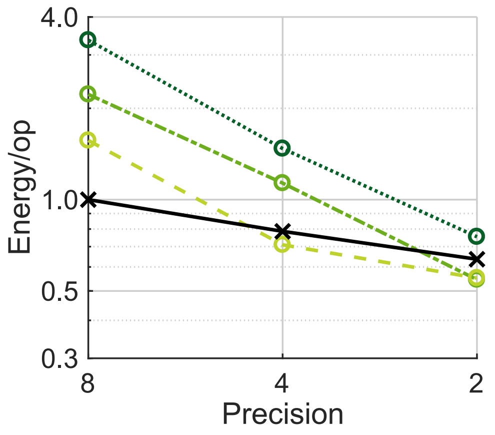


(b) Weight-only scaling

Figure 5.7 – Normalized energy/op with precision scaling in a D&C MAC.



(a) Symmetric scaling



(b) Weight-only scaling

Figure 5.8 – Normalized energy/op with precision scaling in a serial MAC.

Below 4 bits, 1-level scalable MACs (including the 4-bit serial MAC) can only scale precision through data gating, returning to the slope of the baseline. Despite less scalable, these designs exhibit decent energy savings compared to 2-level scalable MACs thanks to lower overheads. Overall, DVAFS MACs display the best energy reductions across the entire precision range.

When preserving full-precision activations (right subfigures), savings are without exception far lower. As it is unsuitable for weight-only scaling, DVAFS designs are penalized by their overhead compared to the baseline MAC. D&C and serial designs both follow similar trends as for symmetric scaling. 1-level scalable designs appear to be the best trade-off for weight-only scaling: compared to 2-level scalable circuits, they are superior in 4 bits and almost equivalent in 2 bits.

5.4.2 Comparative study

Fig. 5.9 gives an overall comparison of the different MAC architectures for each precision scenario with their Pareto frontiers achievable in terms of energy per operation and throughput per area. The best designs are towards the bottom-right corners of subfigures.

Full-precision

At full precision (Fig. 5.9c), data gating is undoubtedly the most efficient technique, capable of the highest throughput per area, followed first by 1-level and then by 2-level scalable designs which suffer from longer critical path.

Symmetric scaling

When scaling precision symmetrically (Figs. 5.9a-b), the DVAFS architecture clearly outperforms all other architectures in terms of energy per operation. In this mode, its subword parallelism also yields to great throughput-area efficiency.

Note that with symmetric precision scaling, 1-level scalable circuits stay the best compromise at 4 bits and above, this trend reverses at 2-bit precision (visible by the inversion of bright and dark curves from Figs. 5.9b-c to Fig. 5.9c), except for 1-bit serial MACs which stay the worst trade-off due to their low speed and energy efficiency.

Interestingly, D&C and 4-bit serial MACs prove capable of good symmetric scaling, despite not being optimized for it. This is due to the optimization of the first adder stage of these designs, which benefits together to all modes, while for DVAFS, the same hardware has to be optimized for different objectives as the critical path changes from one mode to the other.

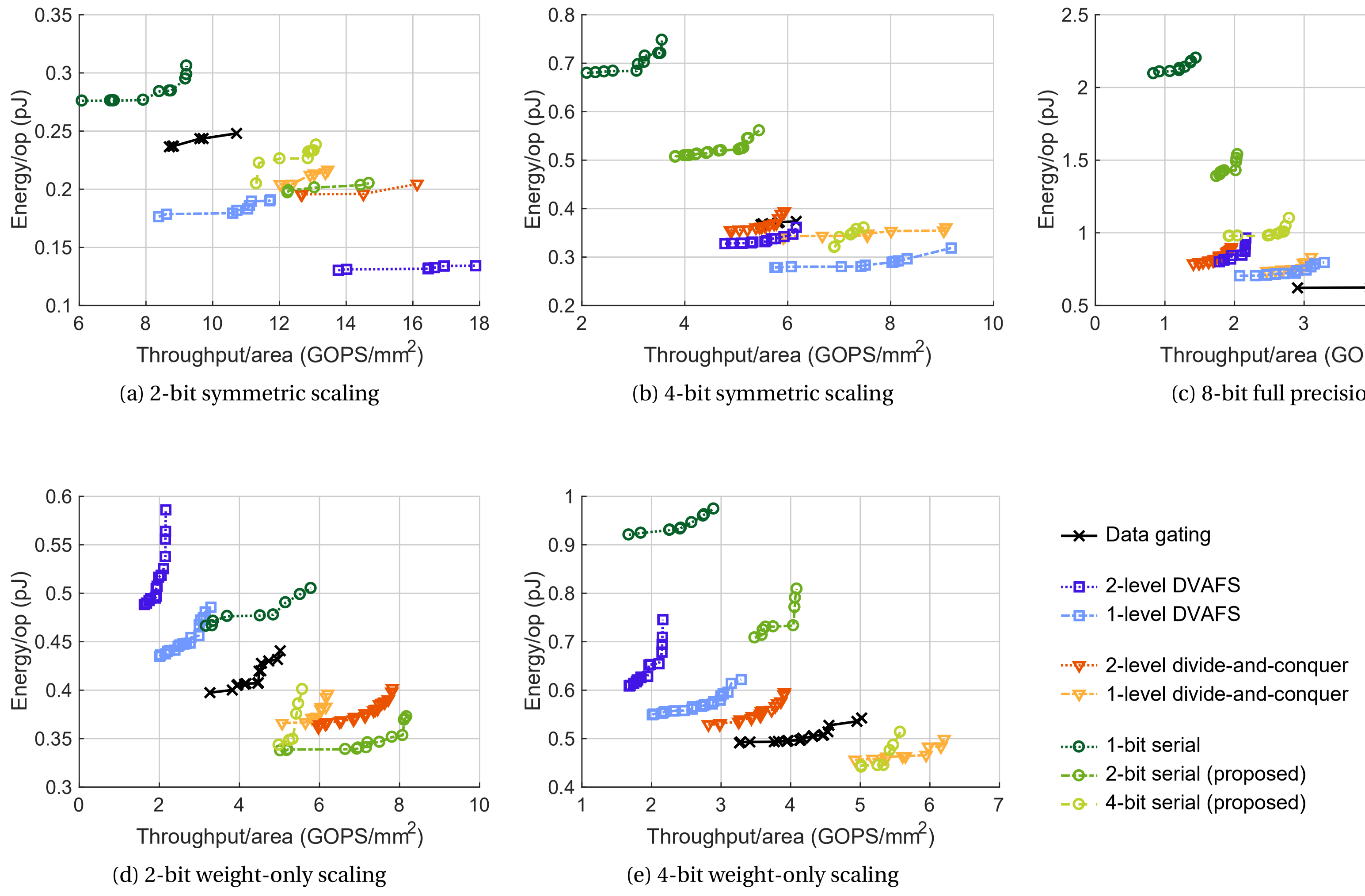


Figure 5.9 – Comparison of MAC architectures in terms of energy/op and throughput/area under each precision mode.

Weight-only scaling

When reducing weight precision only (Figs. 5.9d-e), D&C and multi-bit serial architectures are the best trade-offs between energy and throughput per area. By-passing internal additions, D&C designs are advantaged for throughput, while serial architectures usually display small circuit areas.

Our proposed 4-bit serial design exceeds in terms of energy per operation in both scaled precision modes, while the 2-bit serial design slightly outruns this latter, also in terms of speed, for the lowest precision only.

5.5 Conclusion

This chapter has reviewed different precision-scalable MAC architectures, namely DVAFS, divide-and-conquer and bit serial. This later has been enhanced by introducing multi-bit serial designs. All architectures have been synthesized in a 28 nm process across a wide range of performance and precision scenarios, and compared objectively in terms of energy and throughput per area.

This preliminary study has shown that DVAFS surpasses the state of the art for symmetric scaling, while multi-bit serial and divide-and-conquer strategies exceed when scaling weights only. It has also highlighted that scalability features incur a large energy overhead. In that context, less scalability levels have shown to be a good trade-off thanks to lower circuit overheads. Future works [71] could propose a more extensive analysis of the diverse scaling methods, include advanced optimization techniques, and cover other approximate techniques [72, 73, 74] or precision-scalable designs [75, 76].

Conclusions and perspectives

Conclusions

Power efficiency has become the primary concern for IoT applications, both at the sensor node as well as on its cloud computing counterpart. Unfortunately, achieving high power-efficiency and robustness requires complex and conflicting design constraints and expensive safety margins. For ultra-low-power IoT devices, Process-Voltage-Temperature (PVT) variations are enormous, and designing for worst-case is disastrous for energy efficiency. For cloud computing, high-performance data centers are required to process in real-time the massive amount of data collected from the growing network of IoT devices. Fortunately, the inherent redundancy and noise of such data makes its processing resilient to the use of approximations at different levels of abstraction. This PhD focused on the the development, characterization and validation of several approximate arithmetic units or precision-scalable circuits for image or neural-network applications.

Initial works, discussed in chapter 2, consisted in the development of two conventional techniques for designing approximate arithmetic circuits: the Inexact Speculative Adder (ISA) and Gate-Level Pruning (GLP). The ISA is a new architecture of speculative adder. Like state-of-the-art adders, it segments the addition into several sub-adders with carry speculated from preceding sub-blocks. But the ISA features a shorter speculative overhead coupled with a novel dual-direction error compensation scheme. This shifts most of the speculative overhead out of the critical path and improves control of the average and worst-case errors, which can be set at design time by sizing the number of sub-blocks and error compensation bits. At 10 % worst-case relative error, it offers up to 70 % power-delay-area reductions compared to previous works. The second technique, the GLP, consists in a CAD tool that removes the least-significant gates of a circuit. For adders, this method enables 80 % power-delay-area savings at 10 % mean relative error. These conventional techniques have been successfully combined together or with overclocking, as well as extended to build bigger hardware accelerators, such as multipliers or FPUs. It is to be noted that the ranges of error-engineering possibilities offered by these techniques is extremely large. However, if the level of approximation is too high, reaching too high significance bits, it would not be worth computing lower-significance bits, that could simply be truncated. This would have the advantage to also reduce storage and memory access costs. For instance, despite some figures show ISAs with many speculative paths, the ideal design would rather embed a single speculative path with an approximation of rounding type. Therefore, future works should make sure to first correctly size bitwidths before applying such techniques. The gains from the introduced approximation would necessarily be lower, but there should be room for such a fine-tuned optimization for highly-efficient circuits of GPUs or hardware accelerators.

To overcome the limitations of digital design itself, a novel concept is proposed in chapter 3 to co-optimize circuit timing together with functionality by exploiting artificially-built false critical paths. This method proposes to insert monitor elements that can cut critical paths to prevent their activation, transforming them into false paths. This technique has been applied to approximate arithmetic circuits with the Carry Cut-Back Adder (CCBA) in which high-significance stages can cut the carry propagation chain at lower-significance positions, preventing the carry-chain activation while guaranteeing a high accuracy. This lightweight approach strongly relaxes timing constraints and outperforms all state-of-the-art approximate and truncated adders for worst-case errors. For a bounded relative error of only 0.001 %, it enables energy reductions up to 36 % compared to low-power exact designs. At high accuracy and low speed, the CCBA has even proven possible to surpass truncation for mean errors. It is worth mentioning that, even more easily than speculative circuits, this technique can be made dynamically configurable to embed an exact mode, simply by disabling the cut-backs to allow normal propagation throughout the critical paths. With its exceptionally low circuit overhead and extreme accuracy, this technique could help building highly-efficient hardware accelerators, dynamically configurable, and with a better predictable and controllable impact on their functionality.

ISA and GLP are brought together in chapter 4 with the design and tape-out of a quad-core chip with approximate single-precision FPUs. For this, a novel fast and sign-compatible Inexact Speculative Multiplier (ISM) has been presented based on the ISA concept. The multi-core chip, called Diana and realized in a 65 nm technology, contains 4 FPUs with an identical two-stage pipelined architecture. One FPU is exact and compliant to the IEEE 754 standard, while the three others are approximate variants where mantissa adders and multipliers are replaced by approximate versions of them: *pruned*, *speculative*, and *mixed*. Characteristics and power measurements show power-area savings of 23 % for the pruned FPU, 33 % for the speculative one, and up to 53 % on the mixed FPU. Pruned and mixed FPU cores have similar relative error RMS around 10^{-3}. The speculative FPU shows much lower errors around 10^{-5} for multiplications and 10^{-6} for additions or substractions, which is close to the single-precision machine epsilon itself at 10^{-7}. Contrarily to the others, the speculative FPU can also set an upper bound on the relative error. The use of those FPUs is validated on a HDR-image tone-mapping software, a popular algorithm that locally optimizes and corrects contrast and intensively uses floating-point computations. Results on a set of images of 512×512 pixels tone-mapped with the approximate FPUs show no visible quality loss. Image PSNR ranges from 76 dB using the pruned FPU to 127 dB using the speculative FPU, this latter, extremely accurate, is globally equivalent to using 21 mantissa bits instead of 24 in standard. Since this technique does not leverage the power consumed in the memory and core overhead, energy reductions are limited to the only FPU operator. On the bright side, as speculative circuits contain the full (exact) functional circuit, simply disconnected to be executed in parallel, they can inexpensively be implemented with a configurable exact-computation mode.

Chapter 5 focuses on architectures of precision-scalable MAC unit optimized for neural-network processing. Deep learning has come with enormous computational needs for MAC operations. Fortunately, reduced precision has demonstrated large benefits with minimal impact on accuracy, paving the way towards embedded processing in mobile devices and IoT nodes. Many hardware improvements have recently been proposed, either with extra parallelization capabilities or with bit-serial approaches, all demonstrating precision scalability at run time. In this chapter, the most prominent of these MAC accelerators from ISSCC 2017 and 2018 are reviewed, implemented, and compared in a fair way in terms of power, silicon area and throughput. A new hybrid multi-bit serial MAC is also proposed. This study is carried out in a 28 nm commercial CMOS process, with equivalent code and tool optimizations. Precision is scaled from 8 bits down to 2 bits using both hardware configurability and data gating. This preliminary study has shown that DVAFS surpasses the state of the art for symmetric scaling, while our proposed multi-bit serial design and divide-and-conquer strategy exceed when scaling weights only. This chapter has also highlighted the large energy overhead induced by scalability features, which was not reported in literature. In that context, less scalability levels combined to the use of data gating for low precisions might be a good trade-off thanks to lower circuit overheads. Future works could propose a more extensive analysis of the diverse scaling methods and cover additional precision-scalable architectures, aiming to understand the key trends for reducing computation costs in neural-network processing via precision scaling

Outlooks

Approximate circuits have demonstrated a huge potential for saving resources in comparison to their exact counterparts. They offer many degrees of freedom to introduce or control errors, as well as a rich design space of implementation possibilities.

Many works, included this one, have enhanced approximate circuits like adders and multipliers. But as approximate computing is now reaching an adolescent phase, these techniques need to be integrated on bigger datapaths and applied better and closer to applications. Rather than general-purpose processors, for which an approximate adder or multiplier would bring no gain, application-specific hardware accelerators would offer better prospects for approximate circuits. As much for ASIC as for FPGA, this field should move towards a more cohesive hardware-software co-design.

Another remark that is worth bringing to attention is that techniques with high error levels can easily be outperformed by simply truncated designs. Nevertheless, approximate circuit techniques remain interesting at high accuracy, as it offers more design space and perspectives than truncation. This is why future works should start by correctly sizing bitwidths before applying such techniques. Such strategy would also alleviate memory storage and movement costs.

Due to their static nature, by forcing the energy-accuracy trade-off to be settled at design time, approximate circuits have been facing some resistance going against well-established exact and over-engineered circuits. For this reason, an emphasize on techniques compatible with run-time configurability and scalability is needed, beware of the overhead it could incur.

Additionally, the rise of machine-learning applications, which have proven strongly resilient to approximate computing, has induced a monumental trend and a shift towards this new research direction. With to the proliferation of circuit techniques and designs related to machine-learning processing, some work is needed to enact the advances in circuits and bridge the gap to the software community in order to bring embedded machine-learning processing to an adulthood phase.

Finally, approximate computing is a necessary direction to alleviate the decline of technology scaling despite the growth of multimedia and machine-learning applications. However, many papers have had a limited influence as they were based on ad-hoc hand designs, suffered from a lack of comparability or good quality metrics, or simply were not reaching the application level. To fully exploit the potential of these techniques, future research should focus on bringing together different abstraction levels in a cross-layer holistic approach. Many papers are left to be written.

Appendix

A Characterization with adaptive sample-size inferential statistics (CASSIS)

The characterization of approximate circuits is commonly achieved with exhaustive or random bit-accurate gate-level simulations. However, for high word lengths, the time and memory required for such simulations become prohibitive. Besides, when simulating a random sample, no confidence information is given on the precision of the characterization.

To overcome these limitations, this annex presents a novel method to speed up circuit simulation and ensure result confidence. This method called Characterization with Adaptive Sample-Size Inferential Statistic (CASSIS) [55] exploits statistical properties of the approximated simulation result to reduce its required number of stimuli. From user-defined confidence requirements, the CASSIS method computes the minimal number of simulations to obtain the desired accuracy on the characterization.

A.1 CASSIS method

Commonly, the error statistics of approximate circuits are computed by simulating a given number of random inputs. This straightforward method, that requires Bit-Accurate Logic-Level (BALL) simulation is 100-to-1000 times more complex than running a native operation (addition or multiplication) on the CPU. A BALL simulation of a 16-bit approximate adder takes around 300 times more time than the native addition instruction, and 4000 times longer in the case of an approximate multiplier.

Nevertheless, the quality of the statistical characterization obtained from a random sampling is empirical and highly dependent on the number of samples taken and on the chosen input distribution. Besides, the quality of the estimation of the statistics is not evaluated, and the random sampling based on a fixed number of samples can be ineffective in terms of simulation time. To be used in a real application, a method to characterize inexact operators with a user-defined confidence interval is strongly needed, particularly for high word-length inexact operators.

Inferential statistics [77] aim at reproducing the behavior of a large population using a subset of this population. Instead of simulating exhaustively all the possible input operands combinations, the input operands set is sampled to give an estimation with an accuracy h and a probability p that the estimation is contained within the confidence interval. The CASSIS method computes the minimal number of samples to simulate, to estimate a statistical error metric according to (h, p).

Detailed in [55] and extended in [78], CASSIS uses the fact that multiple big-enough simulation samples are belonging to the same probability set, are independent and identically distributed according to the Central Limit Theorem. By taking into account the Standard Deviation (STD) of errors obtained by the same reasoning, a confidence confidence interval can be generated. Every T stimuli during the simulation, CASSIS recalculates and adjusts the number of extra stimuli needed and ends the simulation when (h, p) are reached.

CASSIS can be applied to many statistical metrics. In this chapter, it is demonstrated for computing the mean error and the error rate of approximate adders. Note that to better fit mathematical statistics, mean error and error rate are denoted μ_e and f, and their empirical computations are noted $\overline{\mu_e}$ and $\overline{f}$ (aforementioned metrics in this thesis are all such empirical values). These estimators are associated to N_{μ_e} and N_f, the minimal numbers of samples to estimate them within confidence intervals CI_{μ_e} and CI_f with a probability p.

A.2 Experimental study

For this experimental study, inexact adders have been selected among three major kinds of topology explored in the literature. The Almost Correct Adder (ACA) [50], most known timing-starved adder, is an interesting case study due to its very low error rate. Errors occur when carry chains are longer than the ACA sub-adder size (main ACA design parameter). Thus, ACA designs have a very low frequency of errors, but of high arithmetic distance. The Inexact Speculative Adder (ISA) [33] is the leading architecture of speculative adders, displaying higher error rates than ACA but with lower error values, depending of the number of sub-blocks and error compensation level (main ISA design parameters). The Carry Cut-Back Adder (CCBA) [45] exploits a novel idea of artificially-built false paths, with error rate ranges similarly as for the ISA, but the error values are lower than those generated by the ACA and the ISA, depending of the number of cuts and cutting distance (main CCBA design parameters).

The proposed experimental study aims at showing that 1) the CASSIS method correctly estimates error characteristics of circuits for various bitwidths, 2) this estimation keeps consistent for higher bitwidths where exhaustive simulation is not possible, and finally that 3) for the majority of approximate adders, CASSIS overcomes naive random BALL simulation. Indeed, two cases are shown: CASSIS requires less samples and thus converges faster towards an accurate error estimation, or CASSIS requires more samples than the traditional random BALL simulation which is, in this case, not accurate enough.

Implementations of each above-mentioned adder architecture have been synthesized, varying their bitwidths, from 8 to 32 bits, as well as their main design parameters, in order to cover a large spectrum of error behaviors. CASSIS characterizations have been completed with $h = 5\,\%$ and $p = 95\,\%$ on an Intel Core i7-6700 processor.

A.3 Results

Quality of the estimation for small word-lengths

To first check the quality of the CASSIS method, small word-length inexact adders have been characterized with CASSIS, as well as with an exhaustive characterization using BALL simulations to obtain their real error characteristics. Table A.1 reports the confidence intervals on Mean Error Distance (μ_e) and Error Rate (f) obtained by CASSIS, compared to their real values, and the numbers of samples N used for the CASSIS characterization. For both 8-bit and 16-bit adders, the CASSIS confidence intervals almost always contain the real values, demonstrating that CASSIS is accurate. The ACA_8 is the only design for which the CASSIS confidence intervals do not contain the real values (c.f. bold numbers), but the relative error between confidence interval bound and real value is extremely small.

For most operators, only a few tens of thousands of simulated samples were required to get precise error characteristics. For 16-bit ACA, the number of simulated samples has been saturated at 25 millions (c.f. bold sample number). This is due to the fact that ACA adders have a large standard deviation in error values. Though, the CASSIS method outputs very accurate estimated values of f and μ_e. The largest relative error on the estimated values compared to the exhaustive characterization is on the estimation of f of the operator ACA_12, and is equal to 2.5 %.

Consistency of the estimation for 32-bit operators

Table A.2 reports the results for 32-bit inexact adder characterization. To check the consistency of the CASSIS method for this larger word-length, the CASSIS characterization has been compared to random BALL simulation with 5 million samples from [46], which is the typical inexact circuit characterization method as exhaustive simulation is not feasible. The chosen CCBA and ACA adders are Pareto-optimal designs shown in the comparative study of [46]. Those adders are realistic designs to be implemented, and thus represent ideal subjects for CASSIS characterization.

In the case of 32-bit operators, it is to be noted that both characterizations (CASSIS and random BALL) are statistical estimates. In case the two methods do not converge towards the same estimation, bold numbers represent values obtained with higher amount of samples, assumed more accurate. For 2 out of 8 designs (CCBA_1_5 and ISA_2_8), the CASSIS confidence intervals obtained with less simulation samples than the BALL method do not contain the error values

Table A.1 – Estimation results and comparison with exhaustive characterization for operators of small word-lengths.

Bitwidth	**Architecture**	**Operator name**	**CI$_{\mu_e}$**		**μ_e**	**CI$_f$**		**f**	**max(N$_{\mu_e}$,N$_f$)**
8-bit	ISA	ISA_2_2	0.8633	0.9550	0.8750	0.1079	0.1194	0.1094	11,765
		ISA_2_4	0.0416	0.1384	0.0938	0.0104	0.0346	0.0234	578
	ACA	ACA_6	1.6718	1.9856	1.7500	0.0151	0.0177	0.0156	35,873
16-bit	CCBA	CCBA_1_6	0.7299	0.8175	0.75	0.1825	0.2043	0.1875	5041
	ISA	ISA_2_4	1.9535	2.0568	1.9688	0.0305	0.0321	0.0308	178,930
		ISA_2_6	0.1725	0.2688	0.2422	0.0054	0.0084	0.0076	11,602
	ACA	ACA_12	9.4957	9.9386	9.6875	0.0005	0.0005	0.0005	**25,000,000**
		ACA_8	170.5876	172.4411	**169.6680**	0.0157	0.0158	**0.0156**	**25,000,000**

Table A.2 – Estimation results and comparison with 5-million BALL simulations for 32-bit operators.

Bitwidth	**Architecture**	**Operator name**	**CI$_{\mu_e}$**		**$\overline{\mu_e}$ 5M**	**CI$_f$**		**$\overline{f}$ 5M**	**max(N$_{\mu_e}$,N$_f$)**
32-bit	CCBA	CCBA_1_5	15.6445	15.7430	**15.7593**	0.1222	0.1230	**0.1231**	2,792,512
		CCBA_1_6	18.7698	**18.8922**	18.9718	0.0287	0.0288	0.0287	17,008,400
		CCBA_1_7	0.2132	0.2613	0.2420	0.0067	0.0082	0.0076	50,176
		CCBA_1_9	0.4421	0.5482	0.5017	0.0017	0.0021	0.0020	172,676
	ISA	ISA_2_2	8,166.3384	**8,183.3155**	8,189.5880	0.1246	**0.1249**	0.1250	**25,000,000**
		ISA_2_8	3.8263	3.9330	**3.7626**	0.0079	0.0081	0.0079	3,130,201
		ISA_2_10	0.9125	1.0115	1.0027	0.00045	0.00049	0.00049	3,084,740
	ACA	ACA_17	**14,333.5418**	18,116.4251	13,905.1700	0.00005	0.00006	0.00005	**25,000,000**

from this latter. Nevertheless, the obtained estimated values of f and μ_e are very close to the random characterization. Inversely, for 3 of them (CCBA_1_6, ISA_2_2 and ACA_17), the CASSIS method has converged into different confidence intervals than the BALL simulations, as it has determined that more samples were required for safe estimation. This is coherent, as by user decision, the confidence interval has only 95 % chance to contain the real value. The most critical case concerns ACA_17. For this characterization, naive BALL simulation has dangerously underestimated μ_e compared to CASSIS. This is due to the very low error rate of the 32-bit ACA, for which 5 million samples is insufficient to make good statistics on errors.

Number of simulations required for accurate estimation

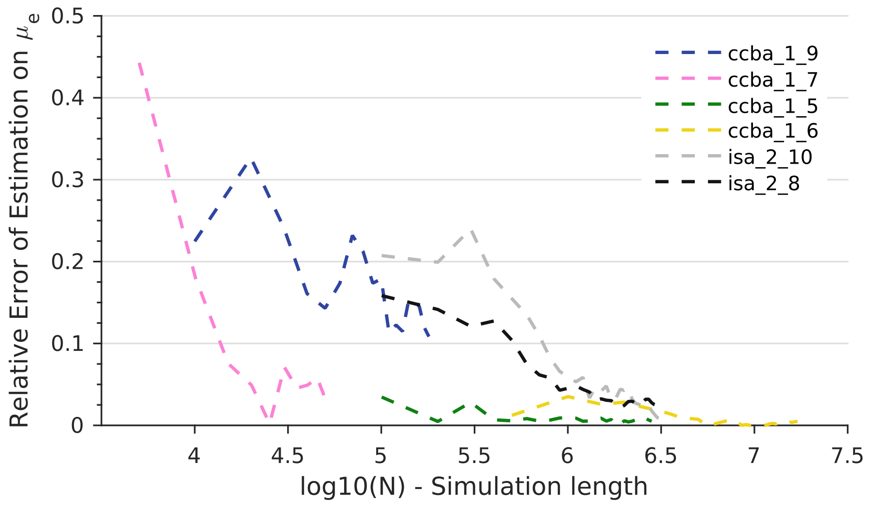
The algorithm implemented by CASSIS refines the estimation of μ_e and f given a refreshment period T. Fig. A.1 illustrates the convergence of the estimation on μ_e and f. The different curves, corresponding to the different operators, have different starting points depending on the chosen T.

The final estimated values are all very accurate since the relative error of estimation is always lower than 0.1 %. Small bumps can be noted in the convergence of the estimated values due to the random sampling processed in each iteration of the algorithm. Besides, the speed of convergence strongly varies depending of the chosen operator. This is why an adaptive sample-size method like CASSIS better fits any operator rather than naive random BALL simulations.

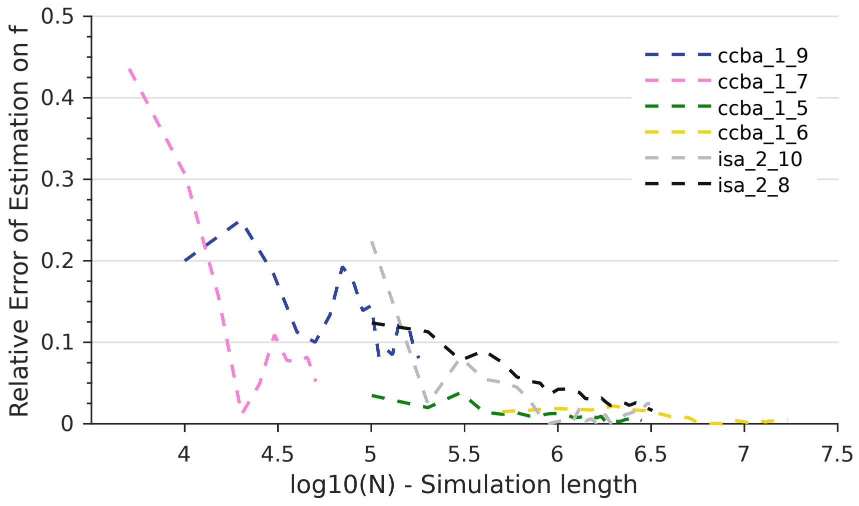
A.4 Conclusion

The proposed CASSIS method automatically adjusts the number of simulations required by using statistical properties of the approximation error. It is studied and demonstrated for the estimation of the error rate and mean error distance of various inexact adders: ACA, ISA and CCBA circuits. Validated by its accurate estimation of error characteristics on 8 to 16-bit circuits, CASSIS has been proven coherence and consistency on larger word-lengths, with 32-bit circuits, where exhaustive simulation is not feasible.

This experimental study has demonstrated that CASSIS overcomes naive random BALL simulations with a fixed number of samples, either by converging towards a more accurate characterization, or by drastically reducing the amount of samples required for an accurate estimation, saving time and resources.



(a) Estimation of μ_e



(b) Estimation of f

Figure A.1 – Convergence of CASSIS for different 32-bit adders with the number of simulated samples for $p = 95\,\%$ and $h = 0.5\,\%$.

B Neural networks using logarithmic quantization

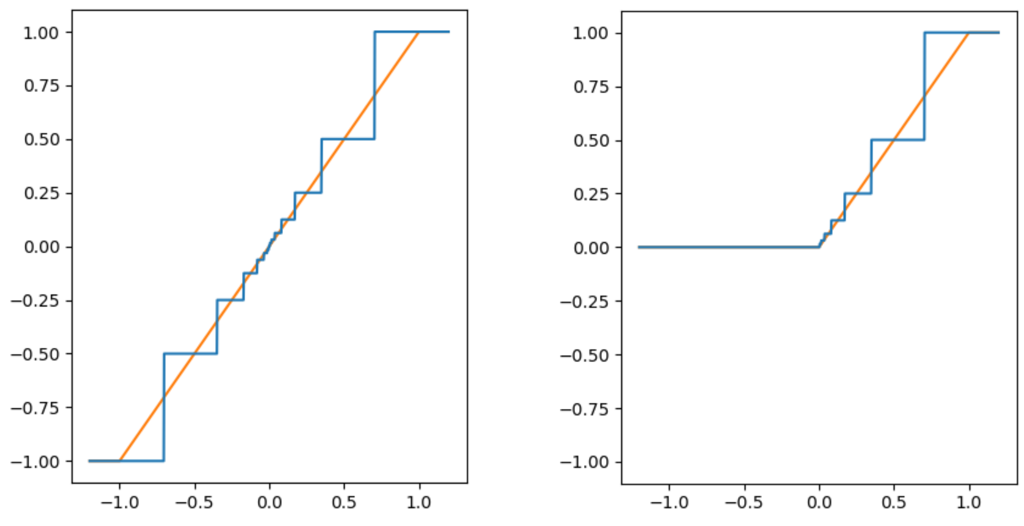
This annex presents an algorithm-level study of Quantized Neural Networks (QNNs) using logarithmic data encoding and computations for weights and activations on the CIFAR-10 [79] image-classification dataset, a good representative of real-life machine-learning applications.

Neural networks are very computationally and memory intensive. In order to be embed such in IoT nodes, their energy consumption should be strongly reduced. Fortunately, they have been proven highly fault tolerant, allowing precision scaling with 4 to 10 bits without loss of accuracy instead of the previously-used 16 or 32-bit floats [70]. This reduces energy and resources required for both computation and memory.

Exploiting the fact that weights and activations are non-uniformly distributed, LogNets (logarithmic neural networks) [80] have demonstrated excellent classification accuracies at low bitwidths (particularly without the need of quantized training as required by traditional QNNs). They have recently grown in popularity [80, 81, 82, 68] as another advantage of logarithmic computations is its ability to eliminate bulky digital multipliers, that could help further reduce the complexity and energy budget in neural-network processing thanks to simplified hardware using only bit-shifts and additions for convolution operations.

B.1 Log-quantized neural-network framework

In this preliminary study, the logarithmic quantization has been emulated in a Keras-TensorFlow framework in order to evaluate its accuracy performance compared to the QNN model [83]. Fig. B.1 shows the resulting quantization, it includes clipping values within $[-1,1]$ for weights or $[0,1]$ for activations. Note that this framework quantizes weights and activations *during*



(a) Weight quantization (b) Activation quantization

Figure B.1 – Logarithmic quantization of weights (a) and activations (b).

training in the forward pass (inference). Computing the gradient to adjust the weights, the back propagation (learning) stays unquantized (thought recent works suggest it could be possible to use stochastic quantization [84]).

B.2 Considerations

In this experimental study, multiple network topologies are evaluated, varying bitwidth precision, network depth as well as layer widths, as in [83]. As shown in Fig. B.2, the networks consist in 4 stages: 3 quantized convolutional stages (CONV) and one non-quantized fully-connected (FC) layer. In each CONV, a QNN layer is repeated 1 to 3 times (A, B or C times in the figure), followed by a max-pooling layer. In a CONV stage, the number of filter is the same for each QNN layer (W_A, W_B or W_C in the figure), varying from 64 to 256. Finally, the bitwidth is kept the same within the entire network (Qb in the figure), and varied between 2, 4 and 8 bits. Only the Pareto-optimal set of QNN topologies from [83] is analyzed, which represents 12 networks.

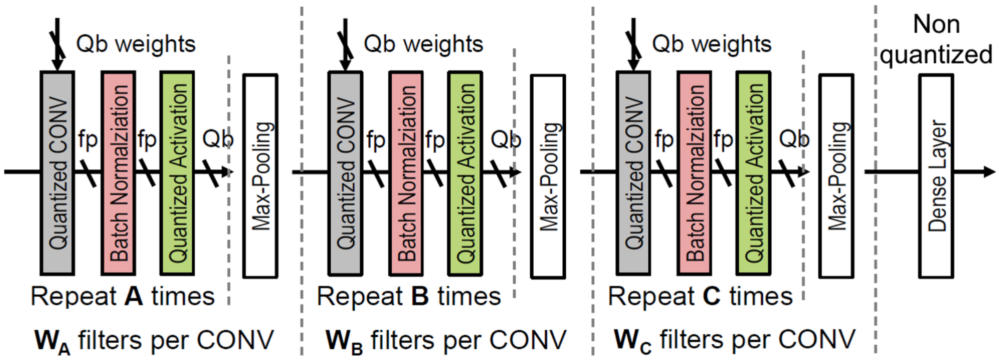


Figure B.2 – CNN topology used in the experimental study [83].

B.3 Results

In this preliminary study, the only resource cost considered for the moment is the model size, which takes into account both network topology and selected bitwidth, and is a good estimate of the memory requirement and its energy consumption.

Fig. B.3 shows the Pareto frontiers of log-quantized and linearly-quantized QNNs for CIFAR-10 according to model size and error rate. At equivalent model size, logarithmic quantization appears similar or slightly better than linear quantization with 0.25 % lower error rate in average.

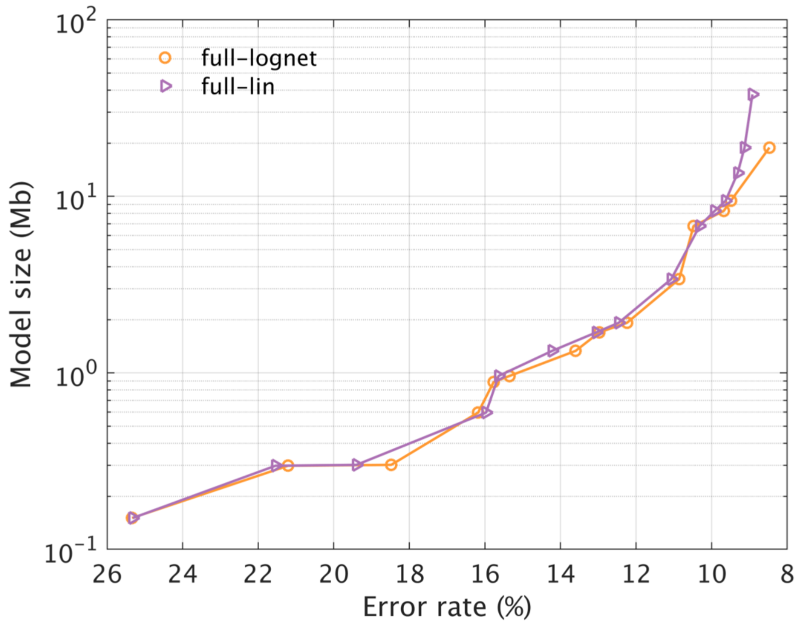


Figure B.3 – Model size and accuracy of log-quantized and traditional QNNs for CIFAR-10.

B.4 Remarks and perspectives

A first study of logarithmic quantization in neural networks has proved similar or barely better accuracy results on small-task CNNs. Beyond this modest result, there are multiple other motivations for log computation, either at hardware level with an opportunity for multiplier-free circuit accelerators, or at algorithm level with its singular robustness to bitwidth reduction without (re)training, which could offer either a dynamic low-power classification mode or a strategy for a hierarchical cascaded classifier. Besides, optimization algorithms able to train quantized networks are relatively new. Future tools might better fit non-linear quantization, pushed by the rising popularity of fixed-point, low-precision or binary neural networks.

Bibliography

[1] J. Heinlein (vice president of ARM), "presentation at ARM TechCon," Oct. 2016.

[2] M. Alioto, *Enabling the Internet of Things: From Integrated Circuits to Integrated Systems*, 1st ed. Springer Publishing Company, Incorporated, 2017.

[3] B. Moons and M. Verhelst, "An energy-efficient precision-scalable convnet processor in 40-nm cmos," in *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 52, no. 4, April 2017, pp. 903–914.

[4] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "DVAFS: Trading computational accuracy for energy through dynamic-voltage-accuracy-frequency-scaling," in *Design, Automation and Test in Europe (DATE), 2017 IEEE Conference*, March 2017, pp. 488–493.

[5] D. Mohapatra, G. Karakonstantis, and K. Roy, "Significance driven computation: A voltage-scalable, variation-aware, quality-tuning motion estimator," in *Low Power Electronics and Design (ISLPED), 2009 ACM/IEEE International Symposium on*, Aug. 2009, pp. 195–200.

[6] P. Stanley-Marbell, A. Alaghi, M. Carbin, E. Darulova, L. Dolecek, A. Gerstlauer, G. Gillani, D. Jevdjic, T. Moreau, M. Cacciotti *et al.*, "Exploiting errors for efficiency: A survey from circuits to algorithms," in *arXiv*, 2018.

[7] K. He, A. Gerstlauer, and M. Orshansky, "Controlled timing-error acceptance for low energy IDCT design," in *2011 IEEE Design, Automation & Test in Europe (DATE)*, March 2011.

[8] X. Jiao, V. Camus, M. Cacciotti, Y. Jiang, C. Enz, and R. Gupta, "Combining structural and timing errors in overclocked inexact speculative adders," in *2017 IEEE Design, Automation & Test in Europe (DATE)*, 2017.

[9] L. N. Chakrapani, P. Korkmaz, B. E. Akgul, and K. V. Palem, "Probabilistic system-on-a-chip architectures," in *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 12, no. 3, 2007, p. 29.

[10] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: A low-power pipeline based on circuit-level timing

speculation," in *36th IEEE/ACM International Symposium on Microarchitecture (MICRO-36)*, Dec 2003, pp. 7–18.

[11] S. Das, C. Tokunaga, S. Pant, W. Ma, S. Kalaiselvan, K. Lai, D. M. Bull, and D. T. Blaauw, "RazorII: In situ error detection and correction for PVT and SER tolerance," in *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 44, no. 1, Jan 2009, pp. 32–48.

[12] M. Fojtik, D. Fick, Y. Kim, N. Pinckney, D. M. Harris, D. Blaauw, and D. Sylvester, "Bubble Razor: Eliminating timing margins in an ARM Cortex-M3 processor in 45 nm CMOS using architecturally independent error detection and correction," in *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 48, no. 1, Jan 2013, pp. 66–81.

[13] I. J. Chang, D. Mohapatra, and K. Roy, "A priority-based 6T/8T hybrid SRAM architecture for aggressive voltage scaling in video applications," in *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, vol. 21, no. 2, 2011, pp. 101–112.

[14] G. Karakonstantis, C. Roth, C. Benkeser, and A. Burg, "On the exploitation of the inherent error resilience of wireless systems under unreliable silicon," in *2012 49th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2012, pp. 510–515.

[15] L. Yang and B. Murmann, "SRAM voltage scaling for energy-efficient convolutional neural networks," in *2017 18th International Symposium on Quality Electronic Design (ISQED)*, March 2017, pp. 7–12.

[16] T. Liu and S.-L. Lu, "Performance improvement with circuit-level speculation," in *Microarchitecture (MICRO-33), 2000 33rd Annual IEEE/ACM International Symposium on*, Dec. 2000, pp. 348–355.

[17] H. Jiang, L. Liu, F. Lombardi, and J. Han, *Approximate Arithmetic Circuits: Design and Evaluation.* Springer, 2019.

[18] N. Zhu, W.-L. Goh, and K.-S. Yeo, "An enhanced low-power high-speed adder for error-tolerant application," in *Integrated Circuits (ISIC), 12th IEEE International Symposium on*, Dec. 2009, pp. 69–72.

[19] M. Weber, M. Putic, H. Zhang, J. Lach, and J. Huang, "Balancing adder for error tolerant applications," in *Circuits and Systems (ISCAS), 2013 IEEE International Symposium on*, May 2013, pp. 3038–3041.

[20] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," in *Transactions on Circuits and Systems I (TCAS-I), IEEE*, vol. 57, no. 4, April 2010, pp. 850–862.

[21] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "IMPACT: Imprecise adders for low-power approximate computing," in *17th IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2011, pp. 409–414.

[22] A. Lingamneni, C. Enz, K. Palem, and C. Piguet, “Synthesizing parsimonious inexact circuits through probabilistic design techniques,” in *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 12, no. 2s, 2013, p. 93.

[23] A. Lingamneni, C. Enz, J.-L. Nagel, K. Palem, and C. Piguet, “Energy parsimonious circuit design through probabilistic pruning,” in *IEEE/ACM Design, Automation & Test in Europe (DATE)*, March 2011, pp. 1–6.

[24] L. Cavigelli and L. Benini, “Origami: A 803-GOp/s/W convolutional network accelerator,” in *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, vol. 27, no. 11, Nov 2017, pp. 2461–2475.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *MIT Press, Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[26] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, “Envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm FDSOI,” in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, Feb. 2017, pp. 246–247.

[27] A. Mercat, J. Bonnot, M. Pelcat, W. Hamidouche, and D. Menard, “Exploiting computation skip to reduce energy consumption by approximate computing, an HEVC encoder case study,” in *Design, Automation and Test in Europe (DATE), 2017 IEEE Conf.*, March 2017, pp. 494–499.

[28] S. Sidiroglou-Douskos, S. Misailovic, H. Hoffmann, and M. Rinard, “Managing performance vs. accuracy trade-offs with loop perforation,” in *19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, 2011, pp. 124–134.

[29] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, and D. Grossman, “EnerJ: Approximate data types for safe and general low-power computation,” in *ACM SIGPLAN Notices*, vol. 46, no. 6, 2011, pp. 164–174.

[30] J. Park, H. Esmaeilzadeh, X. Zhang, M. Naik, and W. Harris, “FlexJava: Language support for safe and modular approximate programming,” in *2015 ACM Joint Meeting on Foundations of Software Engineering and ACM SIGSOFT Symposium on the Foundations of Software Engineering*, 2015, pp. 745–757.

[31] N. Zhu, W.-L. Goh, G. Wang, and K.-S. Yeo, “Enhanced Low-power High-speed Adder for Error-tolerant Application,” in *SoC Design Conference (ISOCC), 2010 IEEE International*, Nov. 2010, pp. 323–327.

[32] Y. Kim, Y. Zhang, and P. Li, “An Energy Efficient Approximate Adder with Carry Skip for Error Resilient Neuromorphic VLSI Systems,” in *Computer-Aided Design (ICCAD), 2013 IEEE/ACM International Conference on*, Nov. 2013, pp. 130–137.

[33] V. Camus, J. Schlachter, and C. Enz, “Energy-efficient inexact speculative adder with high performance and accuracy control,” in *Circuits and Systems (ISCAS), 2015 IEEE Int. Symposium*, May 2015, pp. 45–48.

[34] V. Camus, J. Schlachter and C. Enz, “Energy-efficient digital design through inexact and approximate arithmetic circuits,” in *New Circuits and Systems Conference (NEWCAS), 2015 IEEE 13th International*, June 2015, pp. 1–4.

[35] J. Schlachter, V. Camus, C. Enz, and K. Palem, “Automatic generation of inexact digital circuits by gate-level pruning,” in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2015, pp. 173–176.

[36] J. Schlachter, V. Camus, K. V. Palem, and C. Enz, “Design and applications of approximate circuits by gate-level pruning,” in *Transactions on Very Large Scale Integration Systems (TVLSI), IEEE*, vol. 25, no. 5, May 2017, pp. 1694–1702.

[37] J. Schlachter, V. Camus, and C. Enz, “Near/sub-threshold circuits and approximate computing: The perfect combination for ultra-low-power systems,” in *2015 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2015, pp. 476–480.

[38] X. Jiao, A. Rahimi, B. Narayanaswamy, H. Fatemi, J. P. de Gyvez, and R. K. Gupta, “Supervised learning based model for predicting variability-induced timing errors,” in *New Circuits and Systems Conference (NEWCAS), 2015 IEEE 13th International*, 2015, pp. 1–4.

[39] H. Cherupalli and J. Sartori, “Graph-based dynamic analysis: Efficient characterization of dynamic timing and activity distributions,” in *Computer-Aided Design (ICCAD), 2015 IEEE/ACM International Conference on*, 2015.

[40] D. H. C. Du, S. H. C. Yen, and S. Ghanta, “On the general false path problem in timing analysis,” in *Design Automation Conference (DAC), 26th ACM/EDAC/IEEE*, June 1989, pp. 555–560.

[41] P. C. McGeer and R. K. Brayton, “Efficient algorithms for computing the longest viable path in a combinational network,” in *Design Automation Conference (DAC), 26th ACM/EDAC/IEEE*, June 1989, pp. 561–567.

[42] M. Abadir, J. Zeng, and J. Bhadra, “Design analysis tool for path extraction and false path identification and method thereof,” Patent US 8 627 249 (B1), Aug. 2002.

[43] B. Siarkowski, “Method and system for false path analysis,” Patent US 7 958 470 (B1), May 2007.

[44] S. Rahim and M. Jain, “Method for modeling and verifying timing exceptions,” Patent US 7 650 581 (B2), Nov. 2008.

[45] V. Camus, J. Schlachter and C. Enz, “A low-power carry cut-back approximate adder with fixed-point implementation and floating-point precision,” in *Design Automation Conference (DAC), 53rd ACM/EDAC/IEEE*, June 2016, pp. 127:1–127:6.

[46] V. Camus, M. Cacciotti, J. Schlachter, and C. Enz, "Design of approximate circuits by fabrication of false timing paths: The carry cut-back adder," in *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS)*, 2018.

[47] V. Camus, J. Schlachter, and C. Enz, "System and method for optimization of digital circuits with timing and behavior co-designed by introduction and exploitation of false paths," Patent US 2 017 337 319 (A1), Nov. 2017.

[48] D. Mohapatra, V. K. Chippa, A. Raghunathan, and K. Roy, "Design of voltage-scalable meta-functions for approximate computing," in *Design, Automation and Test in Europe (DATE), 2011 IEEE Conference and Exhibition on*, March 2011, pp. 1–6.

[49] J. Hu and W. Qian, "A new approximate adder with low relative error and correct sign calculation," in *Design, Automation and Test in Europe (DATE), 2015 IEEE Conference and Exhibition on*, March 2015, pp. 1449–1454.

[50] A. K. Verma, P. Brisk, and P. Ienne, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," in *Design, Automation and Test in Europe (DATE), 2008 IEEE Conference and Exhibition on*, March 2008, pp. 1250–1255.

[51] A. B. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in *Design Automation Conference (DAC), 49th ACM/EDAC/IEEE*, June 2012, pp. 820–825.

[52] C. Liu, J. Han, and F. Lombardi, "An analytical framework for evaluating the error characteristics of approximate adders," in *Transactions on Computers (TC), IEEE*, vol. 64, no. 5, May 2015, pp. 1268–1281.

[53] J. Miao, K. He, A. Gerstlauer, and M. Orshansky, "Modeling and synthesis of quality-energy optimal approximate adders," in *Computer-Aided Design (ICCAD), 2012 IEEE/ACM International Conference on*, Nov. 2012, pp. 728–735.

[54] H. Jiang, C. Liu, L. Liu, F. Lombardi, and J. Han, "A review, classification and comparative evaluation of approximate arithmetic circuits," in *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, Aug. 2017, pp. 60:1–60:34.

[55] J. Bonnot, V. Camus, K. Desnos, and D. Menard, "CASSIS: Characterization with adaptive sample-size inferential statistics applied to inexact circuits," in *26th IEEE European Signal Processing Conference (EUSIPCO)*, Sept. 2018.

[56] C. Yu and M. Ciesielski, "Analyzing imprecise adders using BDDs – a case study," in *VLSI (ISVLSI), 2016 IEEE Computer Society Annual Symposium on*, July 2016, pp. 152–157.

[57] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han, and F. Lombardi, "Design of approximate radix-4 booth multipliers for error-tolerant computing," in *Transactions on Computers (TC), IEEE*, vol. 66, no. 8, Aug. 2017, pp. 1435–1441.

[58] M. Franceschi, V. Camus, A. Ibrahim, C. Enz, and M. Valle, "Approximate FPGA implementation of CORDIC for tactile data processing using speculative adders," in *2017 New Generation of CAS (NGCAS), IEEE*, Sept. 2017, pp. 41–44.

[59] V. Camus, J. Schlachter, C. Enz, M. Gautschi, and F. K. Gurkaynak, "Approximate 32-bit floating-point unit design with 53 % power-area product reduction," in *European Solid-State Circuits (ESSCIRC), 42nd IEEE Conference*, Sept. 2016, pp. 465–468.

[60] Wikipedia contributors, "Single-precision floating-point format — Wikipedia, the free encyclopedia," 2018, [Online; accessed 7-November-2018].

[61] M. Cacciotti, V. Camus, and C. Enz, "A signed-compatible carry cut-back multiplier with configurable exact computation mode," in *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2019, in submission.

[62] F. Conti, D. Rossi, A. Pullini, I. Loi, and L. Benini, "Energy-efficient vision on the PULP platform for ultra-low power parallel computing," in *Signal Processing Systems (SiPS), 2014 IEEE Workshop on*, Oct 2014, pp. 1–6.

[63] N. Moroney, "Local color correction using non-linear masking," in *Color Imaging Conference (CIC), 8th IS&T/SID*, Nov 2000, pp. 108–111.

[64] M. Verhelst and B. Moons, "Embedded deep neural network processing: Algorithmic and processor techniques bring deep learning to IoT and edge devices," in *IEEE Solid-State Circuits Magazine*, vol. 9, no. 4, Nov. 2017, pp. 55–65.

[65] V. Sze, Y. H. Chen, T. J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," in *Proceedings of the IEEE*, vol. 105, no. 12, Dec. 2017, pp. 2295–2329.

[66] D. Shin, J. Lee, J. Lee, and H. Yoo, "DNPU: An 8.1TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, Feb. 2017, pp. 240–241.

[67] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H. J. Yoo, "UNPU: A 50.6TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision," in *2018 IEEE International Solid-State Circuits Conference (ISSCC)*, Feb. 2018, pp. 218–220.

[68] K. Ueyoshi, K. Ando, K. Hirose, S. Takamaeda-Yamazaki, J. Kadomoto, T. Miyata, M. Hamada, T. Kuroda, and M. Motomura, "QUEST: A 7.49TOPS multi-purpose log-quantized DNN inference engine stacked on 96MB 3D SRAM using inductive-coupling technology in 40nm CMOS," in *2018 IEEE International Solid-State Circuits Conference (ISSCC)*, Feb. 2018, pp. 216–218.

[69] V. Camus, C. Enz, and M. Verhelst, "Survey of precision-scalable multiply-accumulate units for neural-network processing," in *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, March 2019.

[70] B. Moons, B. D. Brabandere, L. V. Gool, and M. Verhelst, "Energy-efficient ConvNets through approximate computing," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, March 2016.

[71] V. Camus, L. Mei, C. Enz, and M. Verhelst, "Review and benchmark of precision-scalable multiply-accumulate unit architectures for embedded neural-network processing," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2019, in submission.

[72] M. Franceschi, A. Nannarelli, and M. Valle, "Tunable floating-point for artificial neural networks," in *25th IEEE International Conference on Electronics Circuits and Systems (ICECS)*, 2018.

[73] M. Franceschi, A. Nannarelli and M. Valle, "Tunable floating-point for embedded machine learning algorithms implementation," in *2018 15th IEEE International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, July 2018, pp. 89–92.

[74] H. Jiang, L. Liu, and J. Han, "Special session paper: an efficient hardware design for cerebellar models using approximate circuits," in *2017 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, Oct 2017.

[75] L. Mei, M. Dandekar, D. Rodopoulos, J. Constantin, P. Debacker, R. Lauwereins, and M. Verhelst, "Sub-word parallel precision-scalable MAC engines for efficient embedded DNN inference," in *1st IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, March 2019.

[76] H. Sharma, J. Park, N. Suda, L. Lai, B. Chau, J. K. Kim, V. Chandra, and H. Esmaeilzadeh, "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural networks," in *45th IEEE International Symposium on Computer Architecture (ISCA)*, June 2018, pp. 764–775.

[77] G. Demengel, P. Bénichou, and N. Boy, *Probabilités, statistique inférentielle, fiabilité : outils pour l'ingénieur.*, ser. Mathématiques appliquées. Paris : Ellipses. impr. 1997, cop. 1997., 1997.

[78] J. Bonnot, V. Camus, K. Desnos, and D. Menard, "Adaptive simulation-based framework for error characterization of inexact circuits," in *Elsevier, Microelectronics Reliability*, 2019, in submission.

[79] A. Krizhevsky, "Learning multiple layers of features from tiny images," in *technical report*, 2009.

[80] E. H. Lee, D. Miyashita, E. Chai, B. Murmann, and S. S. Wong, "LogNet: Energy-efficient neural networks using logarithmic computation," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017, pp. 5900–5904.

[81] D. Miyashita, E. H. Lee, and B. Murmann, “Convolutional neural networks using logarithmic data representation,” in *arXiv*, 2016.

[82] K. Hirose, K. Ando, K. Ueyoshi, M. Ikebe, T. Asai, M. Motomura, and S. Takamaeda-Yamazaki, “Quantization error-based regularization in neural networks,” in *International Conference on Innovative Techniques and Applications of Artificial Intelligence.* Springer, 2017, pp. 137–142.

[83] B. Moons, K. Goetschalckx, N. V. Berckelaer, and M. Verhelst, “Minimum energy quantized neural networks,” in *IEEE 51st Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, Oct 2017, pp. 1921–1925.

[84] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, “DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients,” in *arXiv*, 2016.

List of publications

[1] V. Camus, J. Schlachter, and C. Enz, “Energy-efficient inexact speculative adder with high performance and accuracy control,” in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2015, pp. 45–48.

[2] J. Schlachter, V. Camus, C. Enz, and K. Palem, “Automatic generation of inexact digital circuits by gate-level pruning,” in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2015, pp. 173–176.

[3] V. Camus, J. Schlachter, and C. Enz, “Energy-efficient digital design through inexact and approximate arithmetic circuits,” in *2015 IEEE 13th International New Circuits and Systems Conference (NEWCAS)*, June 2015, pp. 1–4.

[4] J. Schlachter, V. Camus, and C. Enz, “Near/sub-threshold circuits and approximate computing: The perfect combination for ultra-low-power systems,” in 2015 *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2015, pp. 476–480.

[5] V. Camus, J. Schlachter, C. Enz, M. Gautschi, and F. K. Gurkaynak, “Approximate 32-bit floating-point unit design with 53 % power-area product reduction,” in *European Solid-State Circuits (ESSCIRC), 42nd IEEE Conference*, Sept. 2016, pp. 465–468.

[6] J. Schlachter, V. Camus, and C. Enz, “Design of energy-efficient discrete cosine transform using pruned arithmetic circuits,” in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2016, pp. 341–344.

[7] V. Camus, J. Schlachter and C. Enz, “A low-power carry cut-back approximate adder with fixed-point implementation and floating-point precision,” in *53rd ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2016, pp. 127:1–127:6.

[8] X. Jiao, V. Camus, M. Cacciotti, Y. Jiang, C. Enz, and R. Gupta, “Combining structural and timing errors in overclocked inexact speculative adders,” in *2017 IEEE Design, Automation & Test in Europe (DATE)*, 2017.

[9] M. Franceschi, V. Camus, A. Ibrahim, C. Enz, and M. Valle, “Approximate FPGA implementation of CORDIC for tactile data processing using speculative adders,” in *2017 IEEE New Generation of CAS (NGCAS)*, Sept. 2017, pp. 41–44.

[10] V. Camus, J. Schlachter, and C. Enz, “System and method for optimization of digital circuits with timing and behavior co-designed by introduction and exploitation of false paths,” Patent US 2 017 337 319 (A1), Nov. 2017.

[11] V. Camus, M. Cacciotti, J. Schlachter, and C. Enz, “Design of approximate circuits by fabrication of false timing paths: The carry cut-back adder,” in *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS)*, July 2018.

[12] M. Cacciotti, V. Camus, J. Schlachter, A. Pezzotta, and C. Enz, “Hardware acceleration of HDR-image tone mapping on an FPGA-CPU platform through high-level synthesis,” in *2018 IEEE International System-on-Chip Conference (SOCC)*, Sept. 2018.

[13] J. Bonnot, V. Camus, K. Desnos, and D. Menard, “CASSIS: Characterization with adaptive sample-size inferential statistics applied to inexact circuits,” in *26th IEEE European Signal Processing Conference (EUSIPCO)*, Sept. 2018.

[14] V. Camus, C. Enz, and M. Verhelst, “Survey of precision-scalable multiply-accumulate units for neural-network processing,” in *1st IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, March 2019.

[15] J. Bonnot, V. Camus, K. Desnos, and D. Menard, “Adaptive simulation-based framework for error characterization of inexact circuits,” in *Elsevier, Microelectronics Reliability*, in submission.

[16] M. Cacciotti, V. Camus, and C. Enz, “A signed-compatible carry cut-back multiplier with configurable exact computation mode,” in *IEEE Transactions on Circuits and Systems II: Express Briefs*, in submission.

[17] V. Camus, L. Mei, C. Enz, and M. Verhelst, “Review and benchmark of precision-scalable multiply-accumulate unit architectures for embedded neural-network processing,” in *IEEE Transactions on Circuits and Systems I: Regular Papers*, in submission.

Information

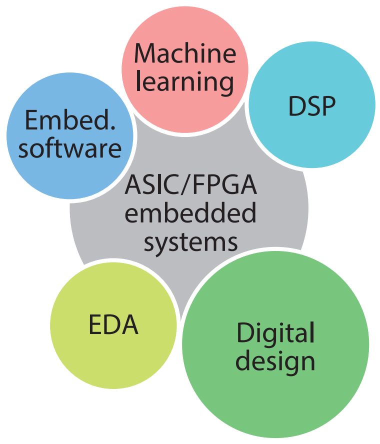
vincent.camus.1988@gmail.com

linkedin.com/in/vcamus

orcid.org/0000000347797742

github.com/vincent-camus

Interests



Skills

Digital design

Scripting & automation

UNIX & EDA tool support

Embedded systems

Software development

Activities

Piano and music
15-year conservatory, concerts as chorister, organist and pianist

Swimming, hiking and skiing

Experience

10/2013-12/2018	**EPFL, Switzerland** PhD on approximate and precision-scalable circuits

R&D

- Designed circuits with advanced gating/timing/multi-mode features
- Designed approximate arithmetic/MAC/FIR/FPU circuits
- Designed precision-scalable MACs for neural-network acceleration
- Taped-out and measured a 65 nm 4-core processor using ATPG
- Developed high-dynamic-range image tone mapping to run on-chip
- Developed neural networks with custom-hardware models for image classification (CIFAR-10, MNIST)

Writing

- 14 peer-reviewed articles, including 2 top conf. (DAC, ESSCIRC) and 1 best student paper award nomination (ISCAS)
- 1 US patent pending, 1 in progress (wrote claims and description)

Team support

- Managed IT (server admin, support for EDA/design kits, webmaster)
- Organized conferences, special sessions and public scientific events

12/2017-06/2018 **KU Leuven, Belgium**
Visiting scientist on energy-efficient neural-network processing

11/2012-09/2013 **CSEM, Switzerland**
Master thesis intern, digital design engineer intern
- Designed ultra-low-voltage digital standard cells
- Modeled double-modulated front-end amplifier for pulse oximeter

08/2012-03/2012 **National Institute of Informatics, Japan**
Research intern on quantum technology and photonics
- Invented a high-purity single-photon source architecture

Education

10/2013-12/2018 **PhD in electrical engineering**
EPFL, Switzerland

09/2010-04/2013 **International MSc in micro and nano-technologies**
EPFL, Switzerland / Grenoble INP, France / Polytechnic of Turin, Italy

Competences

Coursework

Europractice	Verification with SystemVerilog and UVM
Europractice	Low-power advanced digital physical implementation flow
Coursera	Machine learning
EPFL	Management of innovation and technology transfer
Europractice	Hardware security

Computer languages

HDL	SystemVerilog, VHDL, Verilog, Verilog-AMS (basics)
ML	Keras, Tensorflow
General	Python, Perl, Tcl, Bash, Csh, C, C++, LaTeX, Asm (basics)

Software

ASIC	Cadence Genus/Virtuoso, Synopsys DC, Mentor Modelsim/IC/Eldo
FPGA	Xilinx Vivado, Altera Quartus II (basics)
Other	MatLab, Maple, LabVIEW (basics)

Languages

French	Mother tongue
English	Fluent
Japanese	Conversational

Ce document a été imprimé au Centre d'impression EPFL,
imprimerie climatiquement neutre, certifiée myClimate.