Trading off Energy versus Accuracy in Modern Computing Systems: From Digital Circuit Design to Programming Techniques

THÈSE N^{O} 8162 (2018)

PRÉSENTÉE LE 12 JANVIER 2018
À LA FACULTÉ DES SCIENCES ET TECHNIQUES DE L'INGÉNIEUR
LABORATOIRE DE CIRCUITS INTÉGRÉS
PROGRAMME DOCTORAL EN MICROSYSTÈMES ET MICROÉLECTRONIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Jérémy Lucien Maurice SCHLACHTER

acceptée sur proposition du jury:

Prof. P.-A. Farine, président du jury
Prof. C. Enz, directeur de thèse
Prof. O. Sentieys, rapporteur
Prof. K. Palem, rapporteur
Prof. A. Burg, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2018

If we knew what it was we were doing,
it would not be called research,
would it? — Albert Einstein

Abstract

The slowdown of Moore's law, which has been the driving force of the electronics industry over the last 5 decades, is causing serious problem to Integrated Circuits (ICs) improvements. Technology scaling is becoming more and more complex and fabrication costs are growing exponentially. Furthermore, the energy gains associated to technology scaling are slowing down. Meanwhile, the expected boom of Internet of Things (IoT) devices requires ultra-low power ICs to be able to operate for several years without any user intervention, and energy-efficient computing system on the server side to treat all the gathered data.

Approximate computing has emerged as an alternative way to improve energy-efficiency of both, high-performance and low-power computing systems by tolerating small and occasional errors. This energy-accuracy tradeoff can be applied on a wide range of *over-engineered* applications, particularly those involving human senses such as video and image processing.

This thesis first presents an approximate circuit design technique called Gate-Level Pruning, which consists in selectively removing logic gates from any conventional circuit in order to reduce energy consumption, critical path delay, and area occupied on silicon. A Computer Aided Design (CAD) tool has been developed and integrated in the standard digital flow and has been evaluated on several arithmetic circuits, achieving up to 78 % energy-delay-area savings. It is then shown how this methodology can be applied on more complex systems made of multiple arithmetic blocks but also memory: the discrete Cosine Transform (DCT), which is a key building block for image and video processing applications. Then, the speculative adder technique is presented. It consists in cutting carry chains to significantly relax the circuit timing constraints', and therefore drastically reduce energy consumption, area and delay. It is shown that this technique leads to errors of different nature than those produced by gate-level pruning. It is therefore worth combining GLP and speculative adders to obtain even higher savings. This has been verified on IEEE-754 floating point units integrated in a 65 nm process within a low-power multi-core processor. Silicon measurements show up to 27 % power, 36 % area and 53 % power-area savings.

The second part of this thesis introduces software techniques to achieve similar energy-accuracy tradeoffs on commercially available processors. By switching from double precision to single precision floating-point data type and by exploiting vectorization capabilities of modern processors, a factor 2 energy can be saved on a Newton method for solving nonlinear equations. To further investigate the origins of these savings, an energy model based on Energy Per Instructions (EPI) has been built. It turns out that less than 6 % of the total energy is consumed by arithmetic operations and that savings are achieved mainly by reducing the amount of data transferred between registers, cache and main memory. One way to reduce those power-hungry data movements is to use application specific hardware accelerators. Unfortunately, a commercial processor cannot embark accelerators for all the possible applications. To that extent, hardware accelerators are implemented on a Field Programmable Gate Array (FPGA) interconnected with a general-purpose processor to further reduce the energy consumption.

Key words: Approximate computing, inexact circuits, approximate circuits, gate-level pruning, energy efficiency, approximate arithmetic circuits, low power, digital circuits.

Résumé

Le ralentissement de la loi de Moore, qui fut la force motrice de l'industrie électronique pendant les 5 dernières décennies, pose de sérieux problèmes à l'amélioration des circuits intégrés. La miniaturisation des transistors devient de plus en plus complexe et les coûts de fabrication augmentent de manière exponentielle. De plus, les gains de consommation résultants de la miniaturisation diminuent. En même temps, l'explosion du nombre d'objets connectés nécessite d'une part des circuits intégrés à ultra-basse consommation capables de fonctionner pendant plusieurs années sans intervention humaine, et d'autre part des serveurs de calcul efficaces en énergie afin de traiter la masse de données collectée. Le calcul approximatif est une méthode alternative capable d'améliorer l'efficacité énergétique des systèmes informatiques à haute performance et à très basse consommation en tolérant des erreurs de calcul faibles et occasionnelles. Ce compromis entre énergie précision peut être appliqué à un grand nombre d'applications qui sont implémentées de manière trop complexe, tout particulièrement celles impliquant les sens humains tels que le traitement de vidéo ou d'image.

Cette thèse présente tout d'abord le *Gate-Level Pruning* (GLP), une technique de conception de circuits approximatifs qui consiste à enlever des portes logiques d'un circuit conventionnel afin de réduire la consommation énergétique, le délai du chemin critique et la surface occupée sur le silicium. Un outil de Conception Assistée par Ordinateur (CAO) a été développé et intégré à la méthodologie standard de conception de circuits numériques. Cet outil a été testé sur de multiple circuits arithmétiques, montrant jusqu'à 78 % de réduction en énergie-délai-surface. Il est ensuite démontré de quelle façon cette méthodologie peut être appliquée à des systèmes complexes constitués de multiples blocs arithmétiques et mémoires : la transformée en cosinus discrète, qui est un élément de base pour le traitement d'image et de vidéo. Par la suite, la technique d'additionneur spéculatif est présentée. Cette technique consiste à couper la chaîne de retenue afin de détendre les contraintes de temps appliquées au circuit, et par conséquent réduire la consommation, la surface et le délai. Il apparaît que cette technique produit des erreurs de nature différente à celles produites par le pruning. C'est pourquoi il convient de combiner le GLP et l'addition spéculative afin d'obtenir des gains encore plus élevés. Ceci a été vérifié sur des unités de calcul à virgule flottante IEEE-754 intégrées à un processeur multi-coeur basse consommation fabriqué dans une technologie 65 nm. Les mesures effectuées sur le circuit intégré montrent des

baisses de 27 % de puissance, 36 % de surface, et 53 % de puissance-surface.
La deuxième partie de cette thèse présente des techniques de programmation pour accomplir le même compromis énergie-précision mais sur des processeurs disponibles sur le marché. En passant de la précision double à la précision simple pour les nombres à virgule flottante et en exploitant les capabilités de vectorisation des processeurs modernes, un facteur 2 en énergie peut être économisé sur une méthode de Newton pour résoudre des équations non linéaires. Afin d'étudier les origines de ces gains, un modèle énergétique basé sur les Energies Par Instructions (EPI) a été développé. Il s'avère que moins de 6 % de l'énergie totale est consommée par les opérations arithmétiques et que les gains sont obtenus principalement en réduisant le volume de données transféré entre les registres, le cache et la mémoire principale. Une façon de réduire ces transferts coûteux est d'utiliser des accélérateurs matériels. Malheureusement, un processeur commercial ne peut pas embarquer des accélérateurs matériels couvrant toutes les applications possibles. Dans cette mesure, ces accélérateurs matériels sont implémentés sur des circuits programmable (FPGA) interconnectés avec un processeur généraliste afin de réduire d'avantage la quantité d'énergie consommée.

Mots clefs : Calcul approximatif, circuits inexacts, circuits approximatifs, gate-level pruning, circuits arithmétiques approximatifs, efficacité énergétique, faible consommation, circuits numériques.

Acknowledgements

First and foremost, I would like to thank my thesis advisor, Prof. Christian Enz, for his continuous support. Besides all the the technical advices, he always kept me motivated and taught me how to be methodological and rigorous in my research work. I must also thank Prof. Krishna Palem who was not officially my co-advisor but acted as if he was. He helped me a lot with insightful advices and by guiding me towards new research directions. Thanks to him, I had the chance to spend some time at Rice University as a visiting scholar, to meet and collaborate with many remarkable people and visit prestigious research institutions such as the Argonne National Laboratory or the European Center for Middle Range Weather Forecast.

A big part of this work is based on the research done by Dr. Avinash Lingamneni. I want to express my sincere gratitude to him for handing me over his work and supervising me at the beginning of my PhD. I also want to thank Dr. Peter Düben with whom I collaborated on the climate modelling application, and Dr. Mike Fagan who helped me with his expertise in programming and microbenchmarking.

I would also like to thank all members of my thesis jury, Prof. Christian Enz, Prof. Pierre-André Farine, Prof. Andreas Burg, Prof. Krishna Palem and Prof. Olivier Sentieys for carefully reading my PhD manuscript and providing constructive questions and comments.

A big thanks to all my colleagues, Mattia, Vincent, Vladimir, Francesco, Assim, Nino, Farzan, Alessandro, Chunmin, Huaiqi, Arnout, Raffaele, Claudio, Raghav, Anurag and Maria-Anna. They all made the time spent at work fun and enjoyable. I also greatly appreciated all the administrative help I received during these for years from Lysiane Bourquin, Catherine Falik, Sandrine Piffaretti, Lucie Auberson and Marie Halm.

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

Jérémy Schlachter
November 22, 2017

Contents

Abstract (English/Français) **i**

Acknowledgements **v**

List of figures **xi**

List of tables **xv**

1 Introduction **1**

2 State of the art **5**
2.0.1 Approximate circuits exploiting technological knobs 5
2.0.2 Approximation by architectural modifications 9

3 Gate-Level Pruning **15**
3.1 Methodology and implementation in the digital flow 15
3.1.1 Significance and Activity ranking 16
3.1.2 Pruning 18
3.2 Pruned arithmetic circuits 20
3.2.1 Error characterization and metrics 20
3.2.2 Adders implementation 22
3.2.3 Significance and SAP-based pruning 24
3.2.4 Error distributions 25
3.2.5 Activity based pruning 27
3.2.6 Pruned sub-threshold adders 28
3.2.7 Pruned multipliers 30
3.2.8 Pruning versus truncation 32
3.3 Concluding remarks 36

4 Inexact Speculative Adders and Gate-level Pruning **39**
4.1 The inexact speculative adder 40
4.1.1 Error Compensation 41
4.1.2 Design Strategy 42
4.2 Comparison of gate-level pruning and ISA 43

4.2.1 Pruning or Speculation Applied Individually 43
4.3 Mixing of Pruning and Speculation 47
4.4 Comparing approximate adders and truncation 49
4.5 Concluding remarks 51

5 Approximate applications and accelerators design 53
5.1 Discrete Cosine Transform 53
5.1.1 Conventional DCT 53
5.1.2 Quality testing 55
5.1.3 Pruning methodology 56
5.1.4 Results 57
5.2 Tone-mapping of High Dynamic Range images with approximate floating-point units 60
5.2.1 Approximate Floating-Point Units 61
5.2.2 Inexact Speculative Multiplier 61
5.2.3 Error characterization 62
5.2.4 Power Measurements 63
5.2.5 Tone-mapping Algorithm 64
5.2.6 Results 65
5.2.7 Remarks 69
5.3 Estimating the energy gains in an approximate general-purpose processor with weather modelling as a case study 69
5.3.1 Modelling processor energy consumption 70
5.3.2 Modelling the energy gains 70
5.3.3 Case study of IGCM for climate and weather modelling 72
5.4 Concluding remarks 74

6 Programmable systems 77
6.1 Energy measurements and EPI of commercial processors 77
6.2 Energy-precision tradeoff of the Newton method 81
6.2.1 Measurements with performance counters 84
6.3 EPI-based energy model 84
6.3.1 Accuracy of the energy model 86
6.3.2 Energy breakdown 86
6.4 Increasing energy efficiency with Programmable Logic 90
6.5 Concluding remarks 92

7 Conclusion 95
7.0.1 Personal contributions 95
7.0.2 Final conclusion and future work 97

Bibliography 105

Curriculum Vitae

List of Figures

2.1 Energy-probability of correctness of a probabilistic inverter [10] 6
2.2 Pipeline augmented with razor latch [5] 7
2.3 6T and 8T bit-cell schematic [3] 9
2.4 Example of K-Maps of the (a) initial Correct Function (Carry Logic of a Full Adder) (b) function with a favorable 0 to 1 bit flip (c) function with a favorable 1 to 0 bit flip (d) function with a non-favorable 0 to 1 bit flip [13] 11

3.1 Standard digital flow with gate-level pruning 16
3.2 Gate-level netlists of a 3-bit adder and the associated significance attribution (a), the stars indicate the nets that are pruned first. The same netlist with one pruned node (b), and two pruned nodes (c). 17
3.3 CAD framework for Gate-Level Pruning. 19
3.4 Normalized savings of pruned 32-bit adders synthesized at (a) 3.3 GHz and (b) 1.25 GHz 23
3.5 Error distribution of 32-bit adders at 1.25 GHz 26
3.5 Error distributions of 32-bit adders at 1.25 GHz 27
3.6 Activity based pruning of 32-bit adders at 3.3 GHz 28
3.7 Pruned 32-bit adders synthesized with a 180 nm sub-threshold library . 29
3.8 Pruned 16-bit multiplier synthesized in a UMC 65nm technology at 1.2 GHz. Error characterization with a set of 5 million uniformly distributed random inputs. 31
3.9 Pruned 16-bit multiplier synthesized in a UMC 65nm technology at 0.25 GHz. Error characterization with a set of 5 million uniformly distributed random inputs. 32
3.10 Pruned and truncated 32-bit adders synthesized in a UMC 65 nm technology at 3 GHz. Error characterization with a set of 5 million uniformly distributed random inputs. 33
3.11 Maximum Error of pruned and truncated 32-bit adders synthesized in a UMC 65 nm technology at 3 GHz. Error characterization with a set of 5 million uniformly distributed random inputs. 34
3.12 Pruned and truncated 32-bit adders synthesized in a UMC 65 nm technology at 0.8 GHz. Error characterization with a set of 5 million uniformly distributed random inputs. 35

3.13 Maximum Error of pruned and truncated 32-bit adders synthesized in a UMC 65 nm technology at 0.8 GHz. Error characterization with a set of 5 million uniformly distributed random inputs. 36

4.1 General block diagram of an Inexact Speculative Adder (ISA). Each speculative segment consists of a carry speculator (SPEC), a regular adder (ADD) and an error compensation block (COMP). 40
4.2 Example of ISA addition arithmetic with 2-bit speculation, 1-bit correction and 1-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 1st MSB of the preceding sum is flipped. 41
4.3 CAD framework for ISA design. 42
4.4 Error characteristics and normalized cost of 32-bit pruned and speculative adders synthesized at 3.3 GHz 44
4.5 Error characteristics and normalized cost of 32-bit pruned and speculative adders synthesized at 1.6 GHz 45
4.6 Error distribution of a 32-bit ISA synthesized at 1.25 GHz 46
4.7 Error characteristics and normalized cost of 32-bit pruned and speculative adders synthesized at 1.6 GHz 48
4.8 32-bit state-of-the-art approximate adders compared with truncated adders synthesized at a frequency of 800 MHz 50

5.1 2D DCT architecture based on 1D stages 54
5.2 Architecture of the 8 x 8 2D DCT. 55
5.3 Test setup for quality measurement 55
5.4 Image quality versus circuit area 57
5.5 Pictures of Lena resulting from the test setup using the conventional DCT (a) and the approximate versions (b,c,d). p_i denotes the number of pruned nodes per adder and subtractor in stage i. 58
5.6 Floorplan and die microphotograph of the chip. Die size is 1.56 mm^2. . 61
5.7 Measured power consumptions of the 4 FPUs for 3 frequencies 63
5.8 Tone-mapping algorithm. 65
5.9 Original image (a) and pictures tone-mapped by each of the 4 cores (b-e). PSNR is indicated for images processed by the approximate FPUs. Images size is $512 \cdot 512$ pixels. 66
5.10 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. 68

6.1 Measured Energy consumption for varying precision with a problem size of 10000 82
6.2 Modelled and measured energy consumption versus problem size . . . 85

6.3 Energy breakdown of the laplace problem with a problem size of 10000 87
6.3 Energy breakdown of the laplace problem with a problem size of 10000 88
6.4 Energy consumed by the Newton applications on the Xilinx ZC702 board with a problem size of 100000. Convergence accuracy 10^{-5} 91

List of Tables

3.1 Comparison of the two pruning techniques for 10 % MRE 22

4.1 Comparison of the three inexact design techniques. 49

5.1 Power, area and quality of the 4 selected DCTs 59
5.2 Error characteristics of the approximate FPUs 62
5.3 Power, area and power-area product of the FPUs. Measured at 1.2 V, 300 MHz, room temperature in a UMC 65 nm technology. 64
5.4 List of operations with their associated relative energy costs in virtual Joules. Energy numbers are normalized to the cost of an integer operation (add, subtract or multiply) 70
5.5 Normalized energy consumption, global mean error for geopotential height in [m] at day 2 (averaged over 5 forecasts) and normalized operations per virtual Joule (OPVJ) for the different simulations. were presented in [10] and [11]. 73

6.1 Measured arithmetic EPI 79
6.2 Measured data movement EPI 80
6.3 EPIs for simplified model 84

1 Introduction

Over the last 5 decades, Moore's law has been driving Integrated Circuits' (IC) improvements. Technology scaling enabled to double the number of transistors integrated on an IC every two years. This continuous improvement allowed new technologies to emerge and lead to many societal changes. First around the 1980's with the democratization of Personal Computers (PC), one of the first successful commercial example being the Commodore 64. Then, the 1990's saw the boom of the internet and the explosion online shopping in the early 2000's. A few years later smartphones arrived on the market and radically changed our way to communicate. Nowadays we are assisting to the boom of connected objects and the emergence of the industry 4.0, where connected machines can prevent failure and trigger maintenance autonomously. Indeed, the number of Internet of Things (IoT) devices is expected to reach 21 Billion by 2020 [1]. All these technological jumps were made possible thanks to Moore's law: technology scaling drastically reduced the energy consumption of ICs and increased the capabilities of chips by integrating more and more transistors on a single die. Current technology nodes feature transistors with gate length of 14 nm or 10 nm, which is equivalent to 5 to 10 atoms only. Such small gate lengths are extremely hard to build and come at the cost of increased leakage, Process-Voltage-Temperature (PVT) variations and ever-increasing fabrication costs.

Due to those increasing costs associated to the slow-down in the energy-efficiency improvements between two successive technology nodes, Moore's law is reaching its end. Even Intel, one of the leading semiconductor manufacturing company already announced that the pace at which they release new technology nodes is slowing down [2]. Improving energy efficiency of modern computing systems is the main challenge in today's digital circuits design. On the one hand, emerging IoT objects have ultra-low power constraints in order to operate up to several years without any user intervention. On the other hand, these objects generate a huge amount of data that needs to be stored and processed in power hungry data centers, which require complex cooling systems.

Over the last 10 to 15 years, *approximate computing* has emerged as a way to improve energy efficiency of both high-performance and low-power computing, by allowing small and occasional errors to occur. Many applications are *over-engineered* and can run on a reduced arithmetic precision without degrading the end result. Applications involving human senses such as image and video processing are particularly good candidates for approximations since the brain tends to correct artefacts. A good commercial example illustrating this is the Samsung Galaxy S7, which features a display with a WQHD maximal resolution of 2560 x 1440 but is set to be used with a full HD resolution of 1920 x 1080 by default in order to improve battery life.

Approximate computing has several synonyms such as *inexact computing*, or *adequate computing*. It can be applied through different abstraction layers ranging from technology, hardware design, up to algorithm or software level. The first part of this thesis focuses on two approximate hardware design techniques, which consists in introducing a small amount of inaccuracy in exchange for significant area, power and delay savings. The first technique selectively removes logic gates from any standard digital circuit while the second technique invokes architectural changes in arithmetic circuits in order to cut the critical path. The second part of the thesis then shows how programming techniques can be used to achieve similar energy-accuracy tradeoffs on commercially available hardware such as processors and Field Programmable Gate Arrays (FPGA). The detailed organisation of this thesis is as follow:

- Chapter 2 presents the state of the art of approximate computing with a strong focus on approximate hardware design.
- Chapter 3 introduces the Gate-Level Pruning technique (GLP) to systematically remove logic gates starting from any conventional design and it shows how GLP can be applied on arithmetic circuits.
- Chapter 4 presents an approximate adder in which carry chains are cut to relax the constraints: the Inexact Speculative Adder (ISA). GLP and ISA are compared and it is then shown how gate-level pruning can be applied on the ISA to further improve energy-efficiency.
- Chapter 5 shows how these approximate circuits can be used to build accelerators and floating-point units for different applications such as image and video processing.
- Chapter 6 then presents how similar energy-accuracy tradeoffs can be achieved on programmable systems, and more importantly analyses the main energy consumers and gives a few insights on how to reduce their energy consumption. It is then shown how hardware accelerators implemented on an FPGA can further increase the energy-efficiency.

References

[1] Simona Jankowski, James Covello, Heather Bellini, Joe Ritchie, and Daniela Costa. The internet of things: Making sense of the next mega-trend. *Goldman Sachs*, 2014.

[2] Tom Simonite. Intel puts the brakes on moore s law. *MIT Technology Review (Mar. 2016)*, 2016.

2 State of the art

Approximate computing is currently a subject undergoing intense study. In the last 5 to 10 years, the number of papers presenting approximate circuits has exploded. This chapter briefly presents the state of the art by categorizing approximate circuits into two categories: circuits exploiting technological knobs such as supply voltage or noise and circuits based on architectural tweaks to introduce approximations.

2.0.1 Approximate circuits exploiting technological knobs

Probabilistic sources of error

In the early 2000s, noise was predicted to reach significant levels in future technologies. In this regard, noise was seen as a serious impediment to technology scaling. However, some innovations did not consider noise as a hurdle, but rather exploited it to achieve an energy-accuracy trade-off using probabilistic circuit architectures.

One of these innovations is the Probabilistic CMOS (PCMOS) switch [4] wherein the output of each CMOS device or inverter is correct with a certain probability. Figure 2.1 shows that reducing the probability of correctness by only a few percent leads to exponential energy savings. It has been shown that this probabilistic behaviour can be exploited to replace expensive software pseudo random generator, required in applications such as Bayesian networks.

This energy-correctness relationship was initially exploited to create inexact arithmetic blocks [6]. For example, the Biased Voltage Scaled adder, where elements computing the more significant bits are powered at a higher voltage, and thus have a higher probability of being correct, whereas elements computing bits of lower significance are powered with a lower voltage and a lower probability of correctness.

Later on, the Probabilistic System on Chip architecture (PSoC) has been proposed

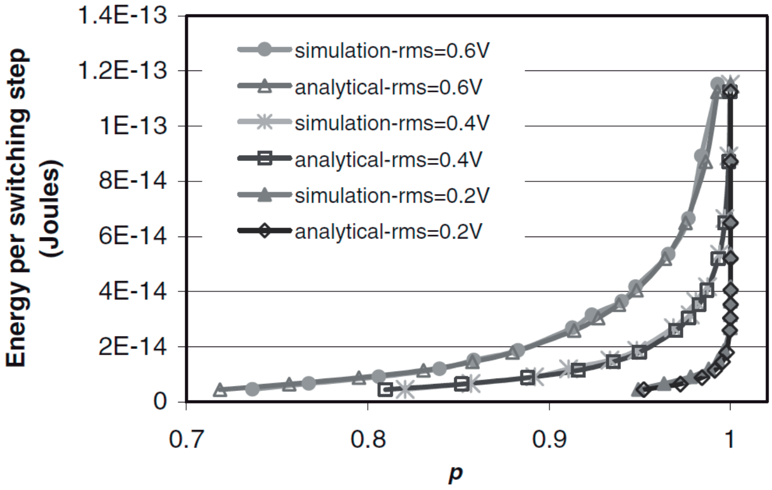


Figure 2.1 – Energy-probability of correctness of a probabilistic inverter [10]

[1]. A standard exact processor combined with a PCMOS based co-processor are computing data together. This platform shows great ability to execute algorithms having probabilistic steps such as random neural network, Bayesian inference and hyper-encryption achieving up to 560X reduction of the energy-performance product.

However, all the previously mentioned techniques were using injected artificial noise because noise levels were not an issue in that time's technologies. Now, in current technologies, noise levels are still not as significant as they were predicted to be by these studies and such techniques may never be applicable. Nevertheless, those works defined the foundations of approximate circuits and presented how the energy-quality tradeoff can be exploited in digital circuits.

Voltage over-scaling

The power consumption of a digital circuit can be expressed as [8]:

$$P = P_{dyn} + P_{stat} = C_{eff} V_{dd}^2 + V_{dd}(I_{sub} + I_{gate}) \tag{2.1}$$

where C_{eff} is the switching capacitance of the circuit, f is the frequency of operation, I_{sub} is the sub-threshold current and I_{gate} is the gate tunnelling leakage current. Due to the quadratic dependence of voltage on dynamic power, voltage scaling is considered as one of the key knob to reduce the energy consumption . In conventional circuits, the maximal operating frequency is determined by the critical path delay at a given power supply voltage V_{dd}. Usually, when the power supply voltage is lowered, the clock frequency is reduced in the same manner in order to avoid any delay violation leading to uncontrollable errors. However, various design techniques using voltage scaling without frequency reductions have been proposed to achieve an energy-accuracy tradeoff while keeping the errors at a rational level. Scaling the voltage below the critical point, where timing errors start to appear is referred as *voltage over-scaling*.

In arithmetic circuits, not all the bits do have the same importance. The significance of each bit grows by a power of 2 when going from the Least Significant Bits (LSB) to the Most Significant Bits (MSB). The error magnitude therefore depends on the bit position. In this regard, various authors [6, 2, 16] proposed to apply non-uniform voltage scaling on adders, where MSBs are powered with a higher voltage than LSBs. In [2], a study shows that for a given operating frequency and power consumption, the non-uniformly scaled adder achieves a lower magnitude of error than the uniformly voltage scaled adder since most of the error are confined to the least significant bits. However, this technique would lead to a large overhead since the routing of a large number of voltage planes is required.

Timing failures

The Razor flip-flop [5] is probably the most famous technique used to get rid of the safety margins in digital circuits. As shown in Figure 2.2, this specific flip-flop 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 a hardware overhead.

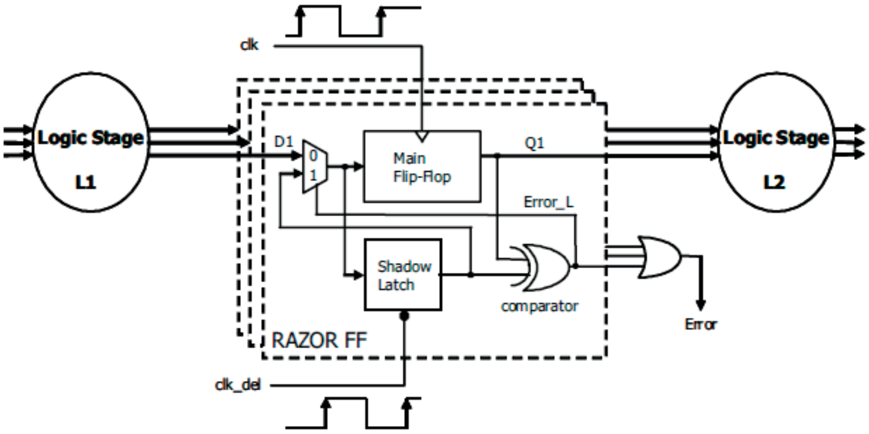


Figure 2.2 – Pipeline augmented with razor latch [5]

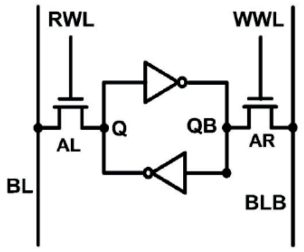
Inspired by Razor, the Adaptive Voltage Over-Scaling (AVOS) [11] monitors real time error rate with the same double-latch than Razor. The difference lies in the fact that AVOS does not feature error correction, and the power supply voltage is dynamically adjusted to reach the error rate set by the designer. With this technique, the safety margins for PVT variations can be strongly reduced and each single Die is powered at the optimal voltage.

In opposition to methods that were described previously, Critical Path Isolation for Timing Adaptiveness (CRISTA) does not detect timing errors but prevents them from appearing. CRISTA is a design methodology that consists in making critical paths rare and predictable under parametric variations, so that with reduced supply voltage, possible delay errors under single-cycle operation can be avoided by a two-cycle operation. The main idea is to monitor a circuit's inputs to detect the critical path activation, and allow an extra clock cycle to ensure error free operations.

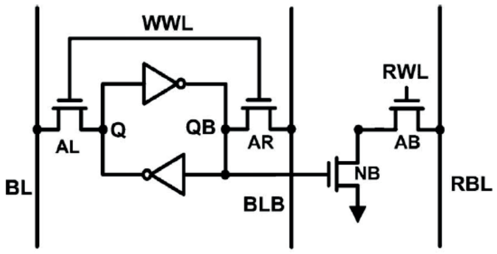
Voltage over-scaling in memories

Memory is an ubiquitous part of today's circuits. The demand of growing speed for wireless communications as well as high quality multimedia systems lead to an increasing quantity of memory in systems. Therefore, memory access power is a non-negligible part of the circuit power consumption. For some applications, as in MPEG-4 video processors, it is even the primary source of power dissipation. Power consumption can be lowered by voltage scaling. Unfortunately, the conventional 6T SRAM bit-cells shown in Fig. 2.3a are very sensitive to power supply scaling, leading to a significant number of read / write errors. To overcome these issues, an 8T SRAM bit cell shown in Figure 2.3b, which is robust to power supply has been proposed. However replacing all 6T SRAM cells by 8T SRAM cells would lead to a huge area overhead (approximately 30% [3]). Therefore, Chang *et. al* [3] proposed a hybrid SRAM architecture for video memory. Owing the fact that the human visual system is mostly sensitive to higher order bits of luminance pixels in video data, 8T SRAM cells are used for these significant bits, whereas, 6T bit cells are affected to the lower order, not so important bits that can tolerate some failures. In [3] a 32 kbit SRAM array was implemented where only few MSB's are 8T bit-cells and other bits are 6T bit-cells. Simulations showed that such a scheme provides 100-150mV over-scaling compared to 6T-only array in the 65 nm technology. Due to this over-scaling, 32% powers saving are achieved in the hybrid SRAM array as compared to the conventional 6T array.

Another application requiring a large amount of memory is wireless communications system, particularly, the amount of memory needed increases with the growing speed of communications. Therefore, [9] proposed to take advantage of the inherent resilience to errors of wireless systems (i.e. "wireless systems are able to recover the transmitted data even when the signal has been heavily distorted by noise and interference due to bad channel conditions"). It was shown that as in video processing, not all bits of wireless systems have the same significance. Hence, the hybrid SRAM (6T/8T) architecture can be implemented for significant power savings.



(a) 6T Bit-cell (LSB's of Hybrid SRAM)



(b) 8T Bit-cell (MSB's of Hybrid SRAM)

Figure 2.3 – 6T and 8T bit-cell schematic [3]

Since timing error are very difficult to predict, voltage over-scaling is possible at the cost of error correction or monitoring circuitry for computational path. The works presented in the following section shows that it is possible to trade accuracy for power consumption, critical path delay, and silicon area without any overhead.

2.0.2 Approximation by architectural modifications

Many papers presented inexact arithmetic circuits based on complexity reduction. As an example, in the Bio-inspired adder [15], the 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 possible approach for complexity reduction consists in removing some transistors of the mirror adder [7], which is a widely used and economical implementation of the full adder cell. The proposed approximate full adder cells feature a very low error rate and thus, can be used to build arithmetic circuits.

In the ripple carry adder, the carry chain can be split to reduce the critical path delay, thus allowing more aggressive voltage scaling and reduced spurious switching. This principle is used in [18, 17], and combined with carry speculation and output balancing to reduce the error magnitude.

Approximate multipliers have been proposed as well, in [12], the authors noticed that changing only one word of the Karnaugh map of a 2x2 multiplier would significantly simplify the multiplier architecture and provide nearly 50% area savings. Error occurs only when performing the 3x3 multiplication (decimal). The result of this operation is 7 instead of 9. Thus the errors occurs with a probability of $\frac{1}{16}$ and a magnitude of 2. It is clear that a 2x2 multiplier is not appropriate for many applications, and larger bit-width are required. Therefore, the authors show how to use this inaccurate 2x2 block and how to combine them with precise blocks in order to build larger multiplier. Simulations show 30% - 50% power saving for 1.39% - 3.35% mean error.

All the previously mentioned works show great improvements in terms of power, area and delay. However, all the previously mentioned approximate circuit have all been tested empirically and all show varying amount of error. It is not clear how these circuits should be used for a generic application. In this perspective, the probabilistic pruning and the probabilistic logic minimization have been proposed to create inexact counterparts of conventional circuits.

In [13, 14], Lingamneni *et. al.* proposed a zero hardware overhead technique called probabilistic logic minimization, where bit-flips are introduced in the minterms of Karnaugh maps as depicted in Figure 2.4, to further minimize circuit components' or nodes' Boolean function guided by the significance and the input combinations probabilities. Only bits flips having the least input probabilities are performed, so that the error rate stays as low as possible. This technique is applied to datapath elements such as adders and multipliers, resulting in savings up to 9X in the energy-delay-area product with an error magnitude below 6%.

The probabilistic pruning technique consists in removing circuits' blocks and their associated wires in order to trade exactness of computation against power, area and delay savings without any overhead. The amount of pruning is dictated by the application's error tolerance. In [13], it has been shown that pruning generate and propagate blocks of a 64 bit Kogge-Stone adder can lead to 7.5 X savings in the energy-delay-area product at the cost of only 10 % relative error magnitude.

In this work, the pruning was done manually and applied to some specific architectures such as the Kogge-Stone or the Han-Carlson adders. In a real digital design flow, the designer never chooses a specific adder architecture but rather sets a timing constraint. The synthesis tool then generates the best architecture for the given timing constraint and technology library. The following chapter therefore presents a Gate-Level Pruning (GLP) tool, which is fully automatized and embedded in a standard digital flow.

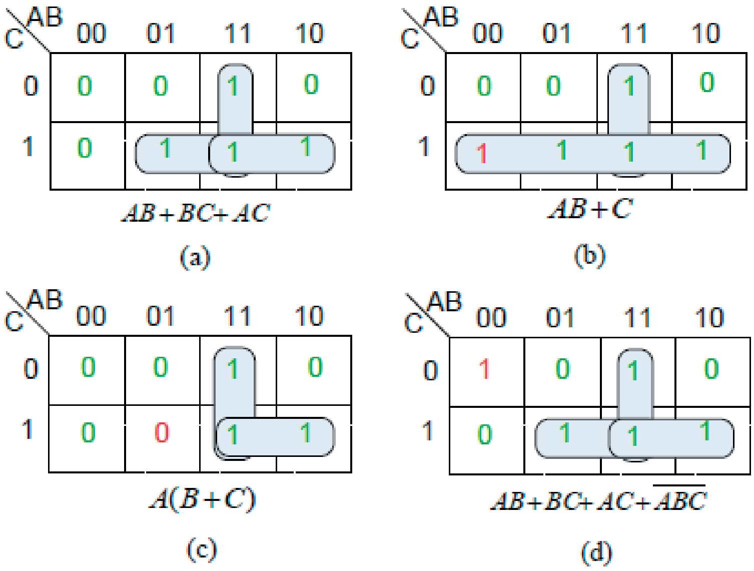


Figure 2.4 – Example of K-Maps of the (a) initial Correct Function (Carry Logic of a Full Adder) (b) function with a favorable 0 to 1 bit flip (c) function with a favorable 1 to 0 bit flip (d) function with a non-favorable 0 to 1 bit flip [13]

References

[1] Lakshmi N Chakrapani, Pinar Korkmaz, Bilge ES Akgul, and Krishna V Palem. Probabilistic system-on-a-chip architectures. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 12(3):29, 2007.

[2] Lakshmi NB Chakrapani, Kirthi Krishna Muntimadugu, Avinash Lingamneni, Jason George, and Krishna V Palem. Highly energy and performance efficient embedded computing through approximately correct arithmetic: A mathematical foundation and preliminary experimental validation. In *Proceedings of the 2008 international conference on Compilers, architectures and synthesis for embedded systems*, pages 187–196. ACM, 2008.

[3] Ik Joon Chang, Debabrata Mohapatra, and Kaushik Roy. A priority-based 6t/8t hybrid sram architecture for aggressive voltage scaling in video applications. *IEEE transactions on circuits and systems for video technology*, 21(2):101–112, 2011.

[4] Suresh Cheemalavagu, Pinar Korkmaz, Krishna V Palem, Bilge ES Akgul, and Lakshmi N Chakrapani. A probabilistic cmos switch and its realization by exploiting noise. In *IFIP International Conference on VLSI*, pages 535–541, 2005.

[5] S. Das, C. Tokunaga, S. Pant, Wei-Hsiang Ma, S. Kalaiselvan, K. Lai, D.M. Bull, and D.T. Blaauw. Razorii: In situ error detection and correction for pvt and ser tolerance. *Solid-State Circuits, IEEE Journal of*, 44(1):32–48, Jan 2009. ISSN 0018-9200. doi: 10.1109/JSSC.2008.2007145.

[6] Jason George, Bo Marr, Bilge ES Akgul, and Krishna V Palem. Probabilistic arithmetic and energy efficient embedded signal processing. In *Proceedings of the 2006 international conference on Compilers, architecture and synthesis for embedded systems*, pages 158–168. ACM, 2006.

[7] Vaibhav Gupta, Debabrata Mohapatra, Sang Phill Park, Anand Raghunathan, and Kaushik Roy. Impact: imprecise adders for low-power approximate computing. In *Proceedings of the 17th IEEE/ACM international symposium on Low-power electronics and design*, pages 409–414. IEEE Press, 2011.

[8] Georgios Karakonstantis and Kaushik Roy. Voltage over-scaling: A cross-layer design perspective for energy efficient systems. In *Circuit Theory and Design (ECCTD), 2011 20th European Conference on*, pages 548–551. IEEE, 2011.

[9] Georgios Karakonstantis, Christoph Roth, Christian Benkeser, and Andreas Burg. On the exploitation of the inherent error resilience of wireless systems under unreliable silicon. In *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, pages 510–515. Ieee, 2012.

[10] Pinar Korkmaz, Bilge ES Akgul, Krishna V Palem, and Lakshmi N Chakrapani. Advocating noise as an agent for ultra-low energy computing: Probabilistic complementary metal–oxide–semiconductor devices and their characteristics. *Japanese journal of applied physics*, 45(4S):3307, 2006.

[11] Philipp Klaus Krause and Ilia Polian. Adaptive voltage over-scaling for resilient applications. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*, pages 1–6. IEEE, 2011.

[12] Parag Kulkarni, Puneet Gupta, and Milos Ercegovac. Trading accuracy for power with an underdesigned multiplier architecture. In *VLSI Design (VLSI Design), 2011 24th International Conference on*, pages 346–351. IEEE, 2011.

[13] A Lingamneni, C. Enz, J. L Nagel, K. Palem, and C. Piguet. Energy parsimonious circuit design through probabilistic pruning. In *Design, Automation Test in Europe Conference (DATE), 2011*, pages 1–6, March 2011. doi: 10.1109/DATE.2011.5763130.

[14] Avinash Lingamneni, Christian Enz, Krishna Palem, and Christian Piguet. Synthesizing parsimonious inexact circuits through probabilistic design techniques. *ACM Transactions on Embedded Computing Systems (TECS)*, 12(2s):93, 2013.

[15] Hamid Reza Mahdiani, Ali Ahmadi, Sied Mehdi Fakhraie, and Caro Lucas. Bio-inspired imprecise computational blocks for efficient vlsi implementation of soft-computing applications. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 57(4):850–862, 2010.

[16] Krishna V Palem, Lakshmi NB Chakrapani, Zvi M Kedem, Avinash Lingamneni, and Kirthi Krishna Muntimadugu. Sustaining moore's law in embedded computing through probabilistic and approximate design: retrospects and prospects. In *Proceedings of the 2009 international conference on Compilers, architecture, and synthesis for embedded systems*, pages 1–10. ACM, 2009.

[17] Matthew Weber, Mateja Putic, Hang Zhang, John Lach, and Jiawei Huang. Balancing adder for error tolerant applications. In *Circuits and Systems (ISCAS), 2013 IEEE International Symposium on*, pages 3038–3041. IEEE, 2013.

[18] Ning Zhu, Wang Ling Goh, and Kiat Seng Yeo. An enhanced low-power high-speed adder for error-tolerant application. In *Integrated Circuits, ISIC'09. Proceedings of the 2009 12th International Symposium on*, pages 69–72. IEEE, 2009.

3 Gate-Level Pruning

Gate-level pruning is an approximate circuit design technique that consists of removing circuits blocks and their associated wires in order to trade exactness of computation against power, area and delay savings without adding any hardware overhead. These blocks can be logic gates, full adder cells, or even bigger functional blocks. The error resulting from the pruning process is proportional to the amount of pruned blocks and is dictated by the application's error tolerance. This chapter presents the automatic gate-level pruning tool, which is fully embedded in the standard digital flow. Section 3.1 introduces the gate-level pruning technique and describes the tools that have been built to automatize the pruning process. Section 3.2 evaluates the proposed methodology on arithmetic circuits such as adders and multipliers, which are key building blocks of computing systems, and investigates the errors resulting from the pruning process.

3.1 Methodology and implementation in the digital flow

An example of standard digital flow is depicted Fig. 3.1. The first step is to describe the circuit to build using Hardware Description Languages (HDL) such as Verilog or VHDL. This high-level description is called the Register Transfer Level (RTL) model and its behavior can be verified by logic simulation using a testbench.

Once the functionality is verified, the RTL can be synthesized with timing, area, power constraints and can be mapped to a certain technology using a standard cell library. The output of the synthesis is a Verilog gate-level netlist of the circuit and its corresponding timing data, which are verified once again with logic simulation.

At this point, the gate-level pruning tool is inserted in the flow. It takes the Verilog gate-level netlist of the exact circuit as an input to generate several approximate variations of the circuit, each with a different number of gates pruned and therefore different amount of error. The GLP process requires several steps as depicted on Fig. 3.3, which

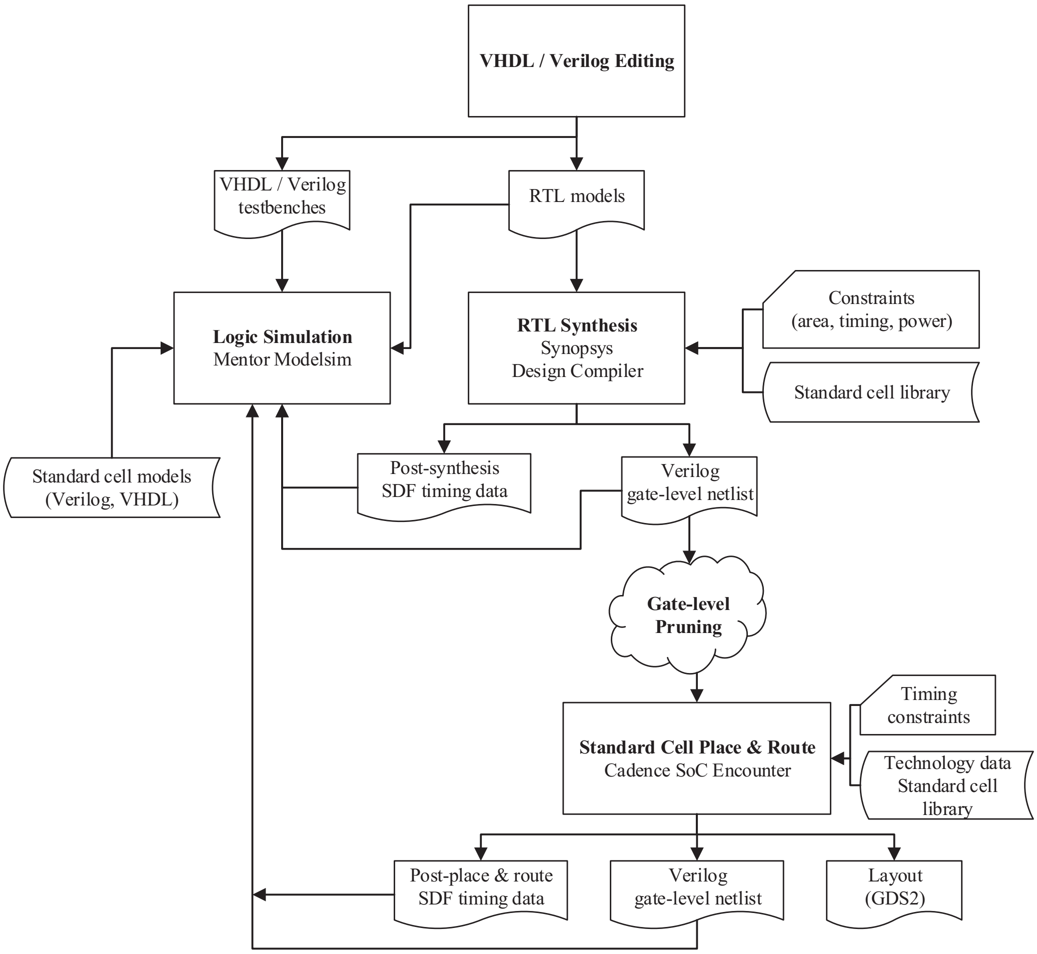


Figure 3.1 – Standard digital flow with gate-level pruning

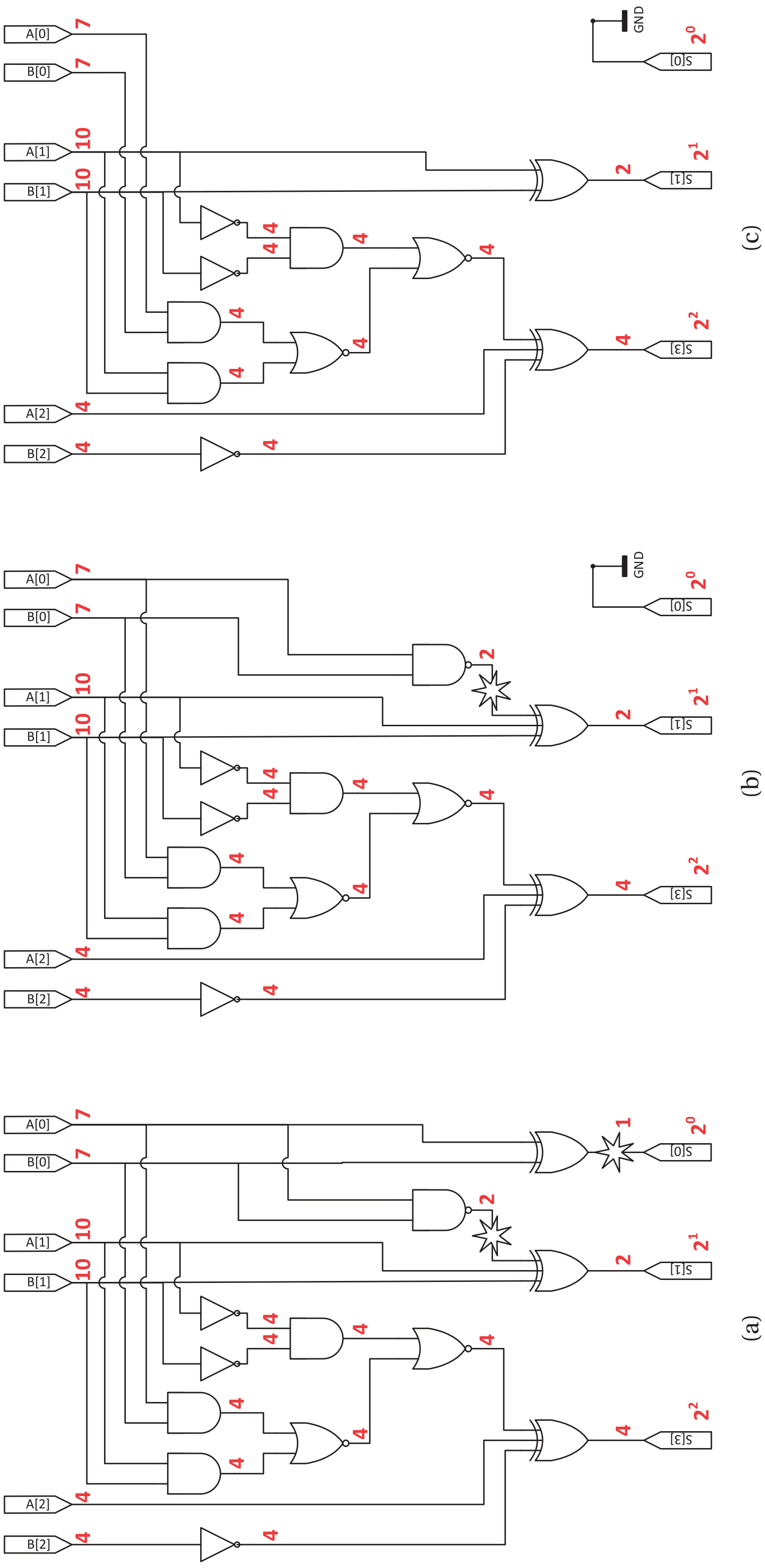
will be explained in the following section.

The GLP process results in an approximated gate-level netlist of the initial design, which can then be used as any normal design for the place and route step, which provides the layout of the circuit. Additional Verilog netlist and SDF timing data are provided to verify the functionality using logic simulation after place and route.

3.1.1 Significance and Activity ranking

This section first explains the methodology of the pruning process and then details how the pruning process is automatized and integrated together with standard industrial design tools.

A circuit netlist as depicted in Fig. 3.2 can be represented by a directed acyclic graph,



(a) (b) (c)

Figure 3.2 – Gate-level netlists of a 3-bit adder and the associated significance attribution (a), the stars indicate the nets that are pruned first. The same netlist with one pruned node (b), and two pruned nodes (c).

where the nodes are components such as gates, and whose edges are wires. The decision to prune a node is generally 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 requirements, the designer may choose to prune nodes according to the activity only in order to minimize the error rate, or by significance only in order to shorten design time by skipping the gate-level simulation process.

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 (TC) of each wire, as well as the time spent at the logic levels 0 and 1 (T0 and T1 respectively). While TC is used to rank the nodes, T0 and T1 are used later in the pruning process to set unconnected gate inputs to a specific value. Note that to get an accurate activity estimation, the system should be simulated with an input stimulus representative of the *real operation* of the circuit.

The significance of each primary output can be set by the designer depending on the application's requirement. However, the experiments performed on adders and multipliers in this work assume an automatic weighted significance attribution, where each bit position has a significance 2 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:

$$\sigma_i = \sum \sigma_{desc(i)} \tag{3.1}$$

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. 3.2: a significance of 2^n is attributed to each primary output of the 3-bit adder (S[0] to S[3]) with n equal to the bit position. The significances are then propagated towards the inputs using reverse graph traversal and eq. 3.1.

3.1.2 Pruning

Once the nodes are ranked according to their significance-activity product, significance only or activity only, the gate-level netlist is modified in order to remove *unessential* nodes from the design. For the sake of simplicity, and in order to maximize the use of the existing EDA tools, the probabilistic pruner does not literally remove the gates from the netlist, but it disconnects the corresponding wires. Gates whose outputs are unconnected will automatically be removed by the synthesis tool. However, leaving gate inputs unconnected would fail the re-synthesis of the design. For this reason, and in order to minimize the error, those inputs are set to 0 if they statistically spend most of the time at 0 (i.e. T0 $\geq$ T1). Otherwise they are connected to 1 (i.e. T0 $<$ T1). This

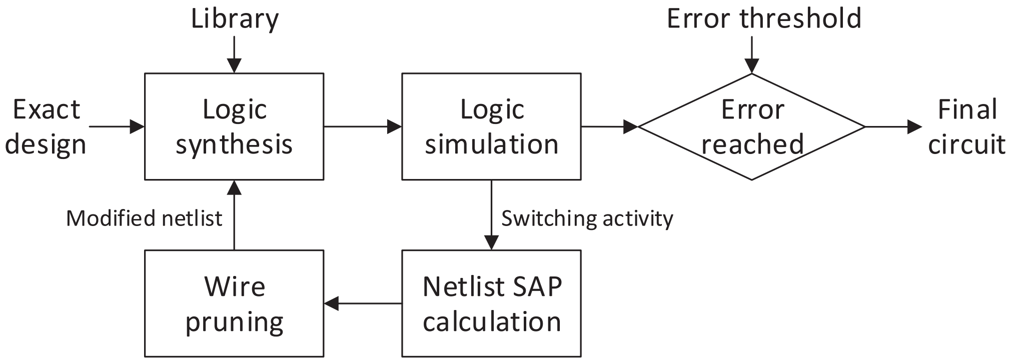


Figure 3.3 – CAD framework for Gate-Level Pruning.

should allow to statistically reduce the error magnitude. The synthesis of the modified netlist therefore improves the design in two ways:

- One or more gates having their outputs unconnected are removed, allowing direct area, power and delay savings.
- Gates having their inputs set to 1 or 0 can then be replaced by lower complexity ones.

Furthermore, the resulting circuit is optimized for the timing and area constraints set by the designer. Fig. 3.3 shows the functional diagram of the presented pruning tool. The initial design is synthesized and mapped to a technology in order to get the gate-level netlist. This netlist then enters a pruning loop composed of four steps:

1. Hardware simulation to monitor the activity of the circuit and to check if the amount of error introduced by the pruned netlist can still fit the application.
2. The significance-activity product is calculated depending on the designer's requirements (weighted or uniform pruning). The significance is computed with a Matlab script that converts the Verliog netlist into a Directed Acyclic Graph (DAG) using the bioinformatics toolbox. A reverse graph traversal is performed to attribute the significance to each node using eq. 3.1. The significance activity product is then computed and the nodes are ranked by a perl script.
3. Wires are pruned in the Verilog gate-level netlist according to the ranking of the nodes. This is done with Perl and Python scripts.
4. Re-synthesis of the netlist is performed in order to remove or replace non-essential gates.

Synthesis and hardware simulations are performed using existing software, whereas scripting languages are used for SAP calculation and wire pruning. The entire framework is controlled by Bash and Perl scripts to fully automatize the tool. This framework outputs all the gate-level netlists ranked by growing order of inexactness, i.e., by decreasing energy-delay-area product. A significant advantage of the proposed tool and methodology is that they can be embedded in an existing standard digital flow, making them fully compatible with any synthesizable HDL code. Moreover, that same flow can be used indifferently for inexact ASIC or FPGA design.

Fig. 3.2 illustrates the netlists provided by the automatic pruning tool for a 3-bit adder. Fig. 3.2a is the conventional circuit where the significance of each node is indicated in red. The two first wires to be pruned, i.e. the ones with the lowest significance, are indicated by stars. The approximate circuit with one pruned node is obtained as follow: the wire with a significance of ’1’ is disconnected and the primary output S[0] is connected to ground assuming it statistically spends most of the time at the logic level ’0’ (T0 > T1). As shown in Fig. 3.2b, the XOR gate preceding the output S[0] can be removed as it becomes useless. Similarly, the circuit with 2 pruned nodes is obtained by disconnecting the net having a significance of ’2’ from the circuit 3.2b. This operation has multiple advantages: the NAND gate can be removed and the 3-inputs XOR gate can be replaced by a 2-inputs XOR. Since the least significant output is connect to ground, this approach could be mixed up with truncation or bit-width reduction, however the main difference here is that the least significant inputs are still used to the calculation, and the carry chain remains intact so that the MSBs of the sum remain exact.

3.2 Pruned arithmetic circuits

The gate-level pruning methodology and tools presented here are technology independent, meaning that similar relative savings can be obtained when applying pruning on circuits implemented in a 65 nm technology or a 28 nm technology. However, to do a fair comparison of the different designs, all the circuits in this work are synthesized using the same UMC 65 nm process.

3.2.1 Error characterization and metrics

In order to get an accurate error characterization of arithmetic circuits, extensive simulations need to be performed. To cover all the possible cases, a 64-bit adder would have to be simulated with 2^{128} different input combinations, which is not possible within a reasonable time. Moreover, the simulation time would need to be multiplied by the number of approximate operators generated by the pruning tool.

Approximate adders are commonly characterized and validated through the simulation of random sets of inputs. Hence, using a set of five million uniformly distributed random inputs allows to get a fairly good estimate of the error characteristics within a reasonable simulation time, but the presented results are statistical estimations depending on the random sample distribution.

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

$$ED = \left|S_{approx} - S_{correct}\right| \tag{3.2}$$

$$RE = \left|\frac{S_{approx} - S_{correct}}{S_{correct}}\right| \tag{3.3}$$

where S_{approx} and $S_{correct}$ are respectively the approximate and correct sums of an addition. Four interesting metrics are considered:

- *Error Rate* – The error rate corresponds the ratio of erroneous computations over the entire set of computations and is defined as follow:

  $$Error\ Rate = \frac{Number\ of\ erroneous\ computations}{Total\ number\ of\ computations} \tag{3.4}$$

- *Mean Relative Error* (*MRE*) – The mean of *RE* is a good estimator of accuracy over a given set of inputs and is interesting at the application level, where for example a few erroneous pixels over an image might have a limited visual impact. It is defined as:

  $$MRE = \frac{1}{N}\sum_{n=1}^{N} RE_n \tag{3.5}$$

  where N is the total number of computations.

- *Relative Error RMS* (RE_{RMS}) – The root mean square (RMS) of *RE* is a good estimator of accuracy and is interesting for many applications, particularly in image and video processing. It is defined as:

  $$RE_{RMS} = \sqrt{\frac{1}{N}\sum_{n=1}^{N} {RE_n}^2} \tag{3.6}$$

- *Maximum Relative Error* (RE_{MAX}) – RE_{MAX} represents the largest relative error of an adder and defines its worst-case accuracy. It is here obtained over a set of computations.

The performances of each implementation are evaluated based on energy consumption, silicon area and critical path delay. The Power-Delay-Area Product (PDAP) is used as a figure of merit to compare each circuit implementation.

3.2.2 Adders implementation

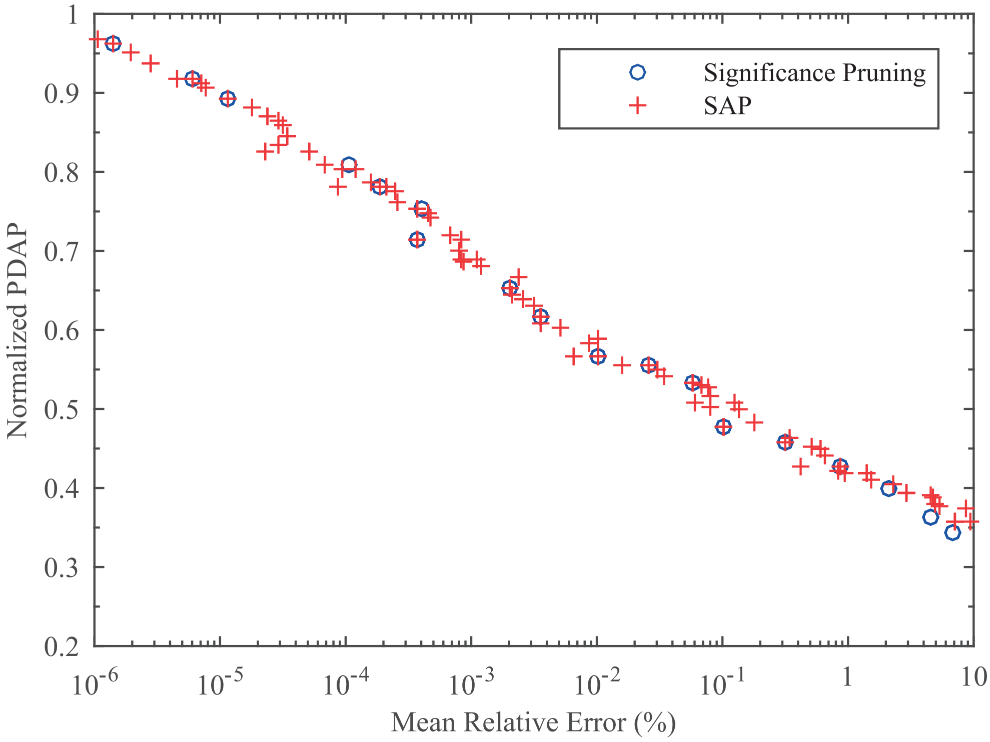
In previous works [3], *Probabilistic Pruning* has been applied manually on several traditional 64-bit adder architectures such as Kogge-Stone and Han-Carlson. However, it is very rare that the designer selects one of these specific architectures. In fact, arithmetic operations are implemented with high-level description languages and the designer does not specify the architecture. As an example, an adder or a multiplier can easily be implemented using the standard *sum* and *product* HDL operands. Low-level structural details are handled by the synthesis tool which selects the optimal architecture based on many optimization scripts and arithmetic IP libraries to fit the given design constraints as shown on Fig. 3.1.

One of the key strength of the proposed tool is that it is able to prune *any digital circuit*, among which those produced by behavioral description in HDL codes, the only condition being that the HDL code is synthesizable.

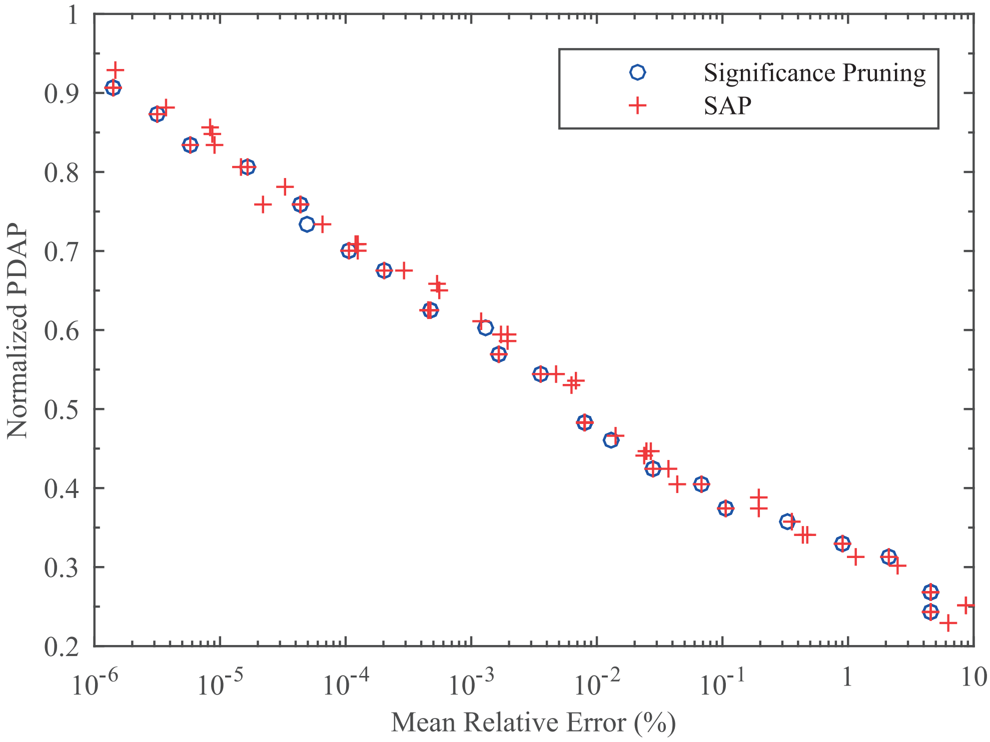
In addition, the previous work [3] exposes the pruning of 64-bit adders, but this approach is a bit too optimistic since highly pruned 64-bit adders could certainly be replaced by 32-bit adders. Moreover, a random uniform distribution over 64 bits features mostly very large numbers and even errors at the bit position 32 almost have no impact on the MRE. The two techniques, GLP and the previous work are compared on a 64-bit adder basis in Table 3.1. It is shown that the finer granularity of the GLP enables much higher savings for similar Mean Relative Error.

Table 3.1 – Comparison of the two pruning techniques for 10 % MRE

Pruning Technique	Area gains	Energy gains	Delay gains	PDAP gains
Gate-level	21X	7.82X	1.07X	175X
Previous work [3]	1.8X	1.8X	2.3X	7.5X



(a) Pruned 32-bit adders at 3.3 GHz



(b) Pruned 32-bit adders at 1.25 GHz

Figure 3.4 – Normalized savings of pruned 32-bit adders synthesized at (a) 3.3 GHz and (b) 1.25 GHz

3.2.3 Significance and SAP-based pruning

Since 32-bit adders are more common and wide spread, this work focuses on this bit-width. Fig. 3.4 shows the savings of pruned 32-bit adders, for two different timing constraints: 3.3 GHz and 1.25 GHz. The errors have been characterized using a set of 5 million random uniformly distributed inputs. Here, the synthesis tool generates the best architecture for each timing constraint, providing an optimized netlist. The two most efficient types of pruning for arithmetic circuits are compared: Significance pruning (circles) and SAP pruning (crosses). It is shown in Fig. 3.4 that both pruning types can provide similar savings for a Relative Error Magnitude of 10 %: 64 % PDAP reduction at 3.3 GHz and up to 78 % PDAP reduction at 1.25 GHz. In some cases, the SAP driven pruning and the Significance driven pruning lead to exactly the same circuit. However, SAP pruning offers a larger range of tradeoffs compared to Significance based pruning, i.e. there is a higher number of pruned designs satisfying a similar error specification when using SAP pruning. This is particularly true for larger circuits. This means that Significance based pruning can be used for a fast first design, and SAP pruning can be used for fine tuning if required. It should be noted that estimating the switching activity of each gate is particularly time consuming as it requires gate-level simulations. Obtaining the SAP pruned netlists for a single 32-bit adder can therefore take up to 15-20 minutes with a set of 5 million inputs, whereas the Significance pruned netlists can be obtained in less than 30 seconds with an Intel Core i7-4770 processor equipped with 16 GB of RAM memory.

Gate-Level Pruning can be applied to circuit synthesized under any frequency constraint, but this parameter influences a lot the savings obtained. Indeed, the timing constraint influences a lot the gate-level architecture: for low frequencies the synthesis tool would implement a ripple-carry adder, which is the most energy-efficient architecture as it requires only as many full adder cells as the word length. For instance, a 32-bit ripple carry adder only requires 32 full adder cells. Those adders can be found in the 300 to 500 MHz range for this technology. When the frequency increases, area and energy consuming architectures are required to meet the timing constraints. Indeed, high frequency adders, such as those presented in Fig. 3.4, are generally large circuits featuring expensive parallelism. Removing a small portion of this kind of circuit poorly affects the correctness of the results and can lead to significant savings, this is why high frequency adder are particularly good candidates for gate-level pruning. On the other hand, adders synthesized at low frequency, which turn out to be ripple carry adders built from a chain of full adder cells, are bad candidates for Gate-Level Pruning. Due to their serial architecture, pruning would rapidly break the carry chain and lead to large errors. Even though a few percent power and area savings are possible, it does not really make sense since the ripple carry adder is one of the most power efficient architecture. Moreover, the savings achieved would be imperceptible at the system level due to their small size. It should be noted that among high frequency adders, the best pruned circuit is not always the one with a higher frequency, and Fig. 3.4 illustrates this fact: the pruned adders at 1.25 GHz have a slightly lower normalized PDAP than the ones at 3.3 GHz. it can also be noticed that the design space is smaller at 1.25 GHz than 3.3 GHz. Indeed, the lower speed allows the use of less parallel architectures, and therefore with a lower number of gates available for pruning.

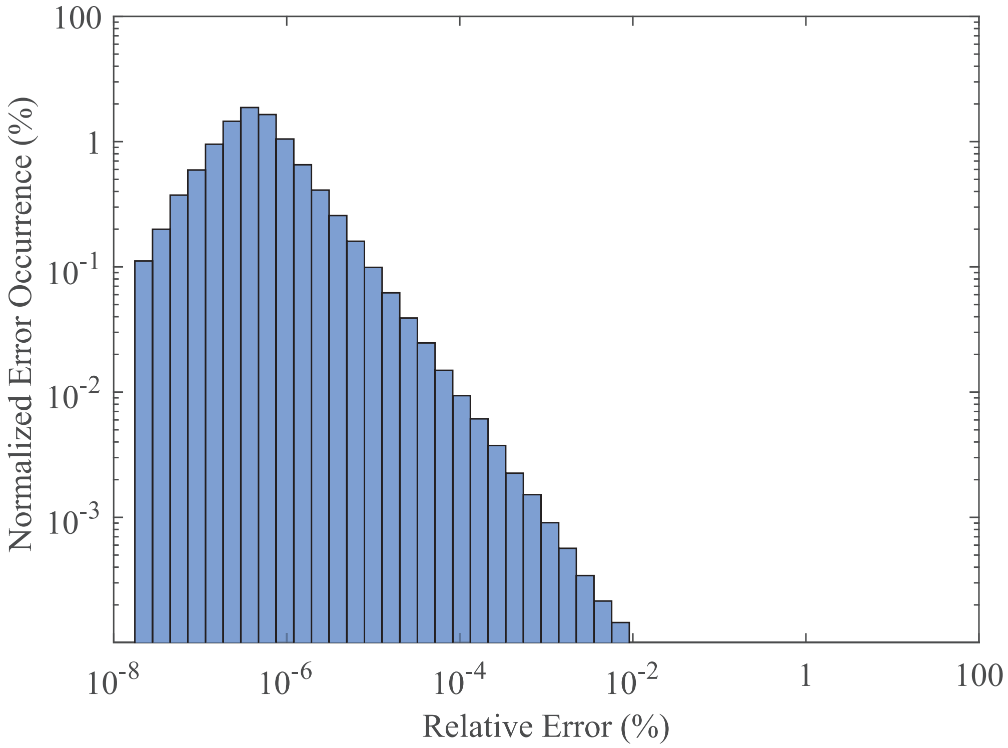
It seems that for pruned adders, the PDAP savings always follow the equation,

$$\text{Normalized PDAP} = \alpha \cdot \log(\text{MRE}) + \beta \tag{3.7}$$

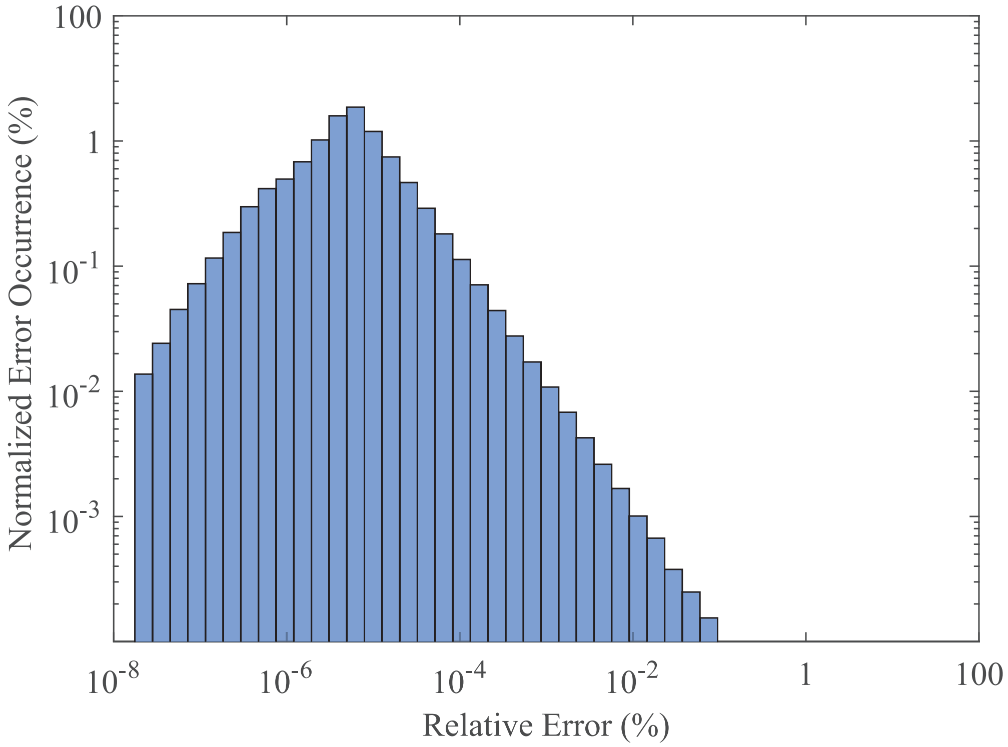
In this work, $\alpha = -0.1$ for all the pruned adders and β depends of the frequency. For instance, $\beta = 0.42$ at 3.3 GHz and $\beta = 0.21$ at 1.25 GHz.

3.2.4 Error distributions

A key property of approximate adders in order to enable their widespread use is that their failures have to remain small, at least relative to the expected exact results. In this regard, the error distributions of pruned adders have been investigated with a set of five million uniformly distributed random inputs. Fig. 3.5 plots the error distribution of 32-bit adders synthesized at a frequency of 1.25 GHz with 10, 20 and 30 nodes pruned. Those pruning levels correspond to 12.5 %, 14.3 % and 37.6 % PDAP savings respectively. It is of particular interest to note that the errors with the highest occurrence are low relative errors. In other words, the highest relative errors are the ones with the lowest occurrence, ensuring a *fail safe* behaviour. It can be observed that for a small number of pruned nodes Fig. 3.5a, most of the errors remain below 10^{-2} %, and as the number of pruned nodes increases, the shape of the distribution remains the same but errors are shifted towards higher magnitudes (Fig. 3.5b and 3.5c).

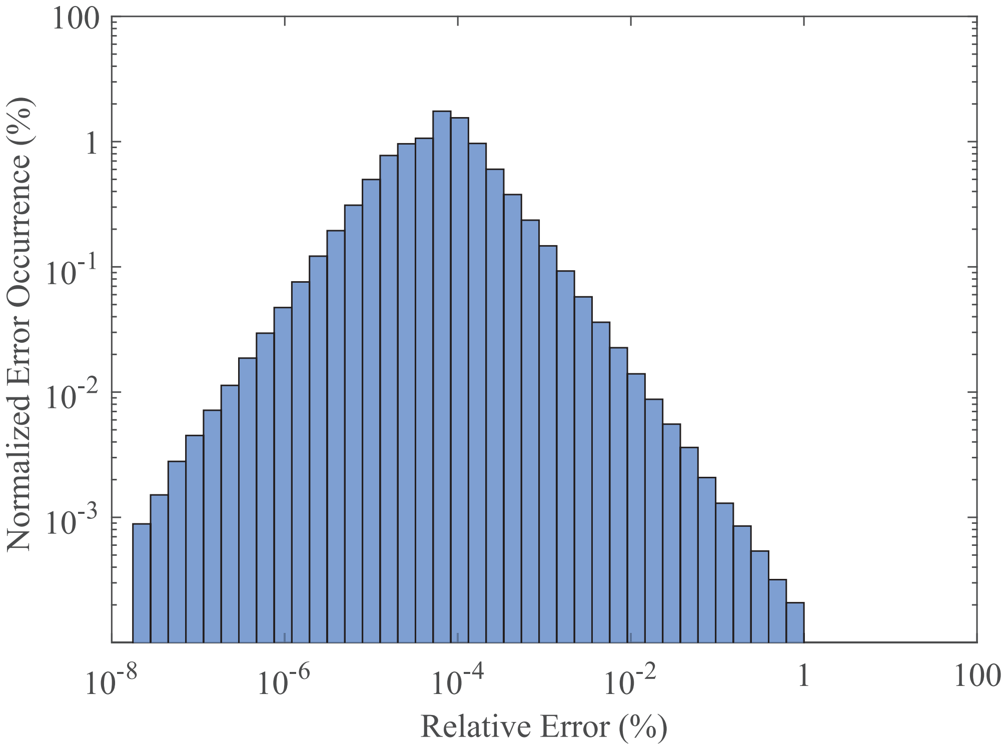


(a) 10 pruned nodes (5.9% area and 7.2% power saved)



(b) 20 pruned nodes (12.2% area and 15% power saved)

Figure 3.5 – Error distribution of 32-bit adders at 1.25 GHz



(c) 30 pruned nodes (20% area and 23% power saved)

Figure 3.5 – Error distributions of 32-bit adders at 1.25 GHz

3.2.5 Activity based pruning

For some applications, the error rate might be more important than the error magnitude. That is to say that only the number of errors matters, regardless of their magnitude. For this reason, the error rate is used to characterize activity based pruned adders. In this case, only activity should be considered to rank the nodes for pruning. Fig. 3.6 shows an activity based pruning of a 32-bit adder at a frequency of 3.3 GHz. The number of possible energy-accuracy tradeoffs is much smaller compared to SAP and significance pruning, but it is still possible to save up to 18 % PDAP for an error rate inferior to 10 %. In the specific case of an adder, the gates close to the MSB are generally pruned first since they are the ones with the lowest activity as they are at the end of the carry chain. For this reason, this pruning methodology leads to very high error magnitudes when applied on arithmetic circuits. Nevertheless, it could be useful for circuits where there is no notion of bit significance.

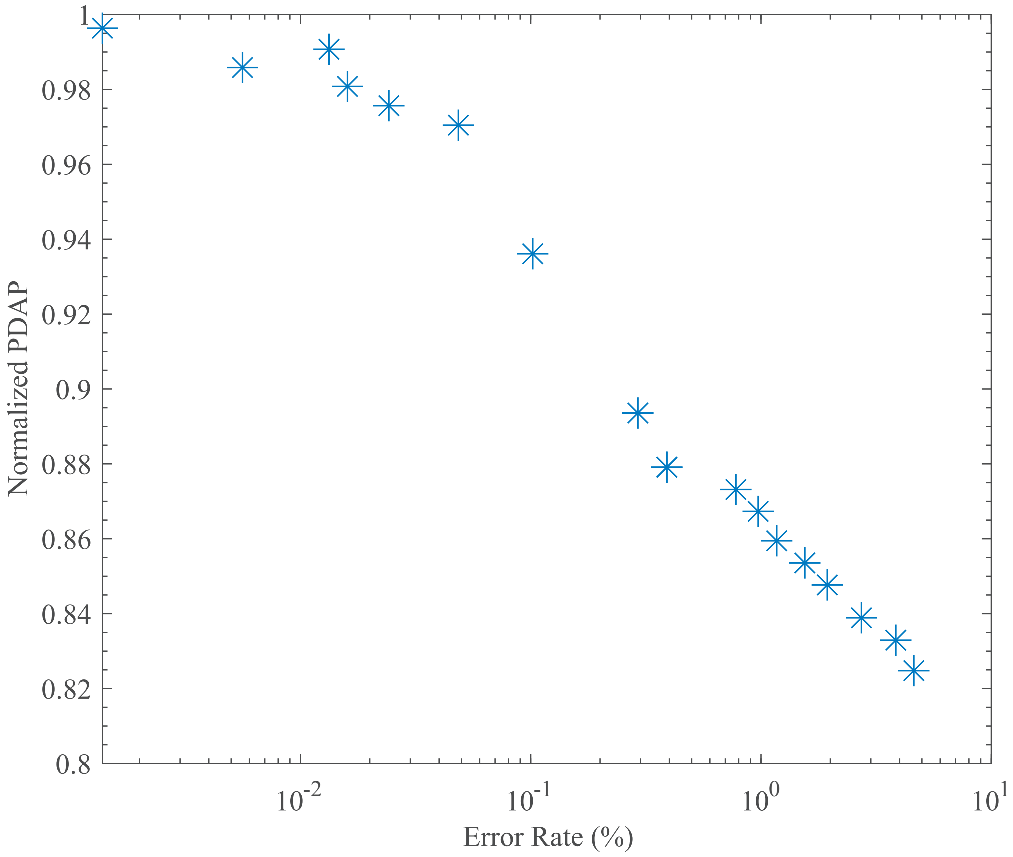


Figure 3.6 – Activity based pruning of 32-bit adders at 3.3 GHz

3.2.6 Pruned sub-threshold adders

Sub-threshold circuits, for which the supply voltage is lower than the threshold voltage V_{th}, are capable of ultra-low power operations and offer a superior energy efficiency than conventional super-threshold circuits. However, with this approach, the PVT variations become huge and thus, significantly increases the complexity of the design process to meet the timing and area constraints. Moreover, operating speed is reduced by orders of magnitude attaining frequencies in the kHz and MHz range and memories require a large area to be functional in the sub-threshold domain. It would therefore only be possible to execute low-complexity applications on a sub/near-threshold hardware. A potential solution to overcome these limitations is the use of approximate circuits. Indeed, applying GLP on sub/near-threshold hardware can significantly relax the design constraints, speed up the nominal frequency of operation and eventually further increase the energy efficiency at the cost of some occasional errors.

In order to verify that sub-threshold circuits can benefit from approximate computing, gate-level pruning has been applied on a 32-bit adder synthesized with a 180 nm sub-threshold library, at a frequency of 22.7 MHz. Sub-threshold pruned adders Fig. 3.7 show similar savings and error characteristics than the ones synthesized with the UMC 65 nm technology.

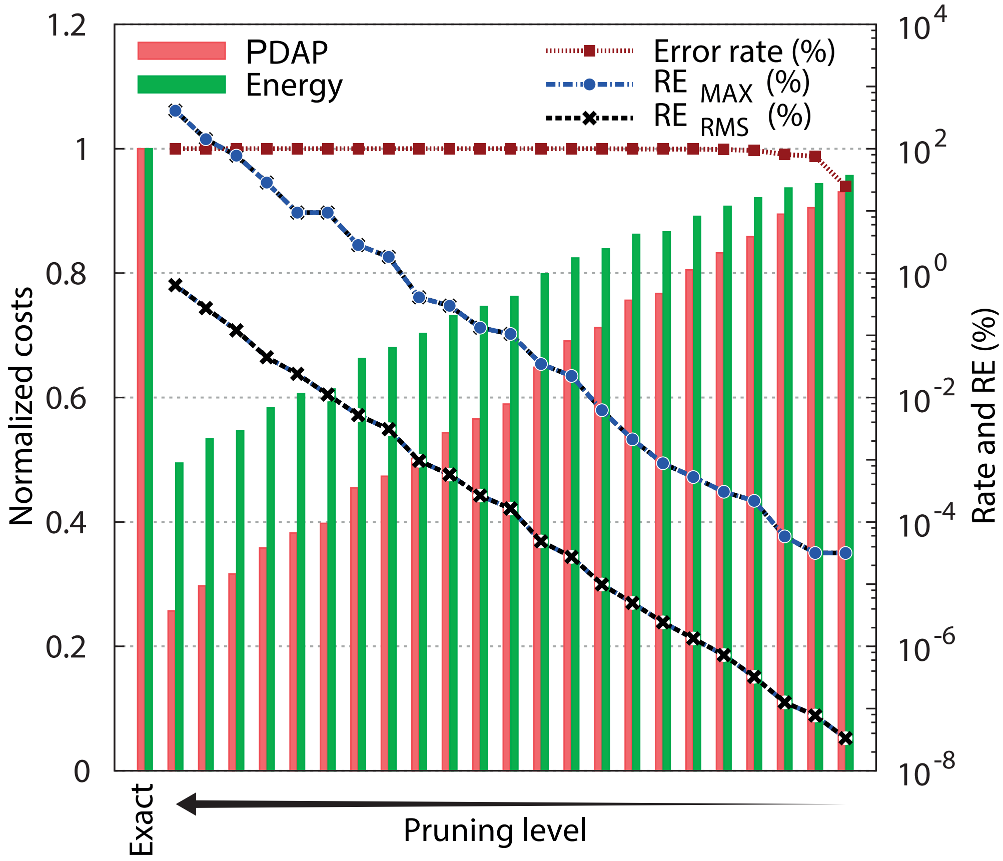


Figure 3.7 – Pruned 32-bit adders synthesized with a 180 nm sub-threshold library

Each couple of bars Fig. 3.7 represents the normalized PDAD and energy consumption of one specific pruned design, and each of these design are characterized with three error metrics. Adders with a low number of pruned nodes (on the right of Fig. 3.7)feature very low error levels of 10^{-4} % RE_{MAX} and 10^{-7} % RE_{RMS} with approximately 10 % PDAP saving and 5 % energy saving. The left part of Fig. 3.7 shows the adders with the highest number of pruned nodes, those can achieve up to 75 % PDAP and 50 % energy savings at the cost of 1 % RE_{RMS} and 600% RE_{MAX}. The first couple of bar on the left represents the exact 32-bit adder which is used as a reference for normalizing the circuit costs. The root mean square of the relative error (RE_{RMS}) and the maximum error (RE_{MAX}) grow linearly as the number of pruned nodes and the energy and PDAP savings are increased. It is also interesting to note that the RE_{RMS} and the RE_{MAX} follow exactly the same trend. Again, the error rate reaches 100 % as soon as one or more primary outputs of the adder are pruned and set to a constant value. These results show that the gains achieved using GLP on sub-threshold adders are similar to the gains obtained in a conventional super-threshold technology, confirming that the results obtained through GLP are technology independent.

3.2.7 Pruned multipliers

Adders are not the only arithmetic circuits that can be found in common digital circuits. Multipliers are heavily used for Digital Signal Processing and they occupy much more area and consume more energy than the adders. For instance, a 32-bit adder synthesized at 1.2 GHz occupies 600 $\mu\mathrm{m}^2$ in a UMC 65 nm technology whereas a 16-bit multiplier synthesized at the same frequency occupies 6300 $\mu\mathrm{m}^2$, which is an order of magnitude difference. Here, the wordlength refers to the number of bits at the input, a 32-bit adder has two input operands of 32-bit and the resulting sum is on 33 bits due to the carry out. The multiplier has two 16-bit input operands and the result of the multiplication is a 32-bit word. Since multipliers cost much more than adders in terms of circuit costs, it would be very beneficial to save some area and power on them using the gate-level pruning tool.

Fig. 3.8 shows the normalized PDAP for a pruned 16-bit high-frequency (1.25 MHz) multiplier. In opposition to the results presented results for adders, the PDAP doesn't decrease gently when the mean relative error increases. This could be due to the fact that the timing constraints a lot the design, and the synthesis results in a highly parallel architecture. The re-synthesis after the wire pruning then produces very different results when different gates are pruned. It is shown on Fig. 3.8 that up to 40 % PDAP reduction can be obtained on this 16-bit multiplier at 1.2 GHz. Unfortunately the PDAP savings achieved on multipliers are lower than the one achieved on adders.

Fig. 3.9 shows the normalized PDAP for a 16-bit low-power (0.25 GHz) multiplier. At this frequency the PDAP decreases gently when the mean relative error increases like for pruned adders, leading to 20 % PDAP saving for a 10 % mean relative error. As for adders, the normalized PDAP follows the equation:

$$\text{Normalized PDAP} = \alpha \cdot \log(\text{MRE}) + \beta \tag{3.8}$$

with $\alpha = -0.03$ and $\beta = 0.82$. These lower relative savings when compared to the ones that can be achieved on adders can be explained by the fact that errors can propagate in multiplicative paths and grow much faster. In addition to that, pruning is applied on a lower bit-width hardware, giving less margin to this technique. Again, pruning is more efficient on high-frequency multipliers than low-power ones. It should also be taken into account than multipliers costs much more in terms of area and power

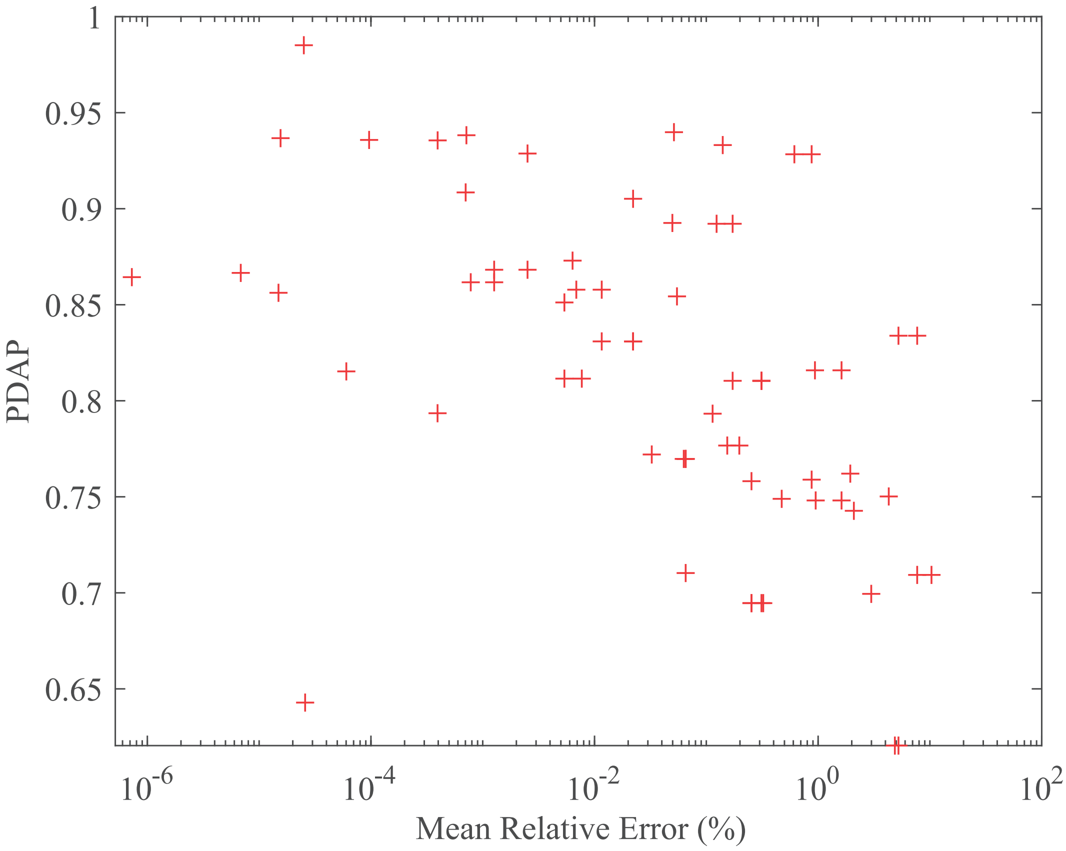


Figure 3.8 – Pruned 16-bit multiplier synthesized in a UMC 65nm technology at 1.2 GHz. Error characterization with a set of 5 million uniformly distributed random inputs.

than adders. Therefore, 40 % PDAP reduction on a multiplier that occupies 6300 μm^2 is actually more interesting than saving 50 % EDAP on an adder that occupies 600 μm^2 only.

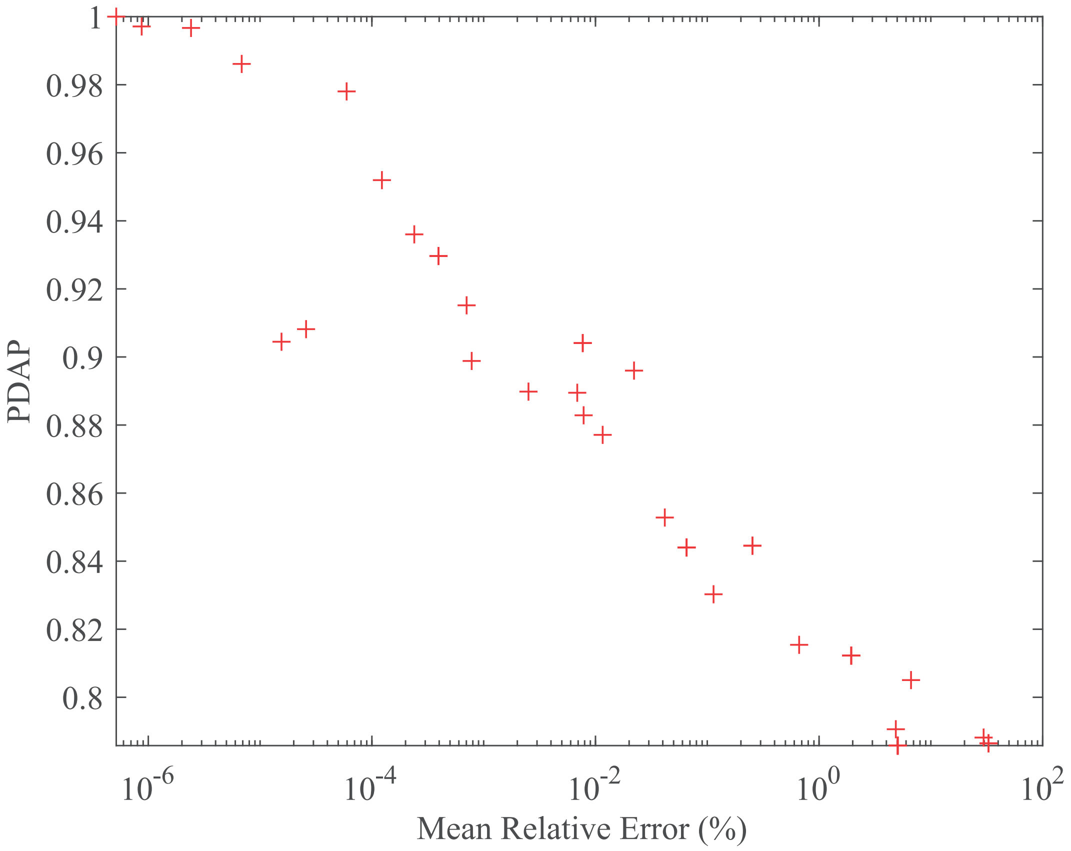


Figure 3.9 – Pruned 16-bit multiplier synthesized in a UMC 65nm technology at 0.25 GHz. Error characterization with a set of 5 million uniformly distributed random inputs.

3.2.8 Pruning versus truncation

Truncation is certainly the most obvious way to reduce precision in an arithmetic circuit. However, approximate circuits are almost never compared with truncated circuits even though reducing bitwidth is the most simple, and sometimes the most efficient way to introduce approximations. Hence, pruned adders and truncated adders have been synthesized in a UMC 65 nm technology at two frequencies: 3 GHz and 0.8. GHz. Fig. 3.10 and 3.11 plot the Power Delay Area product of truncated and pruned 32-bit adders at 3 GHz versus their mean relative error and maximum error distance respectively. Similarly, Fig. 3.12 and 3.13 show the same metrics for pruned and truncated adders at 0.8 GHz. Here truncated adders have been generated simply by reducing the bitwidth.

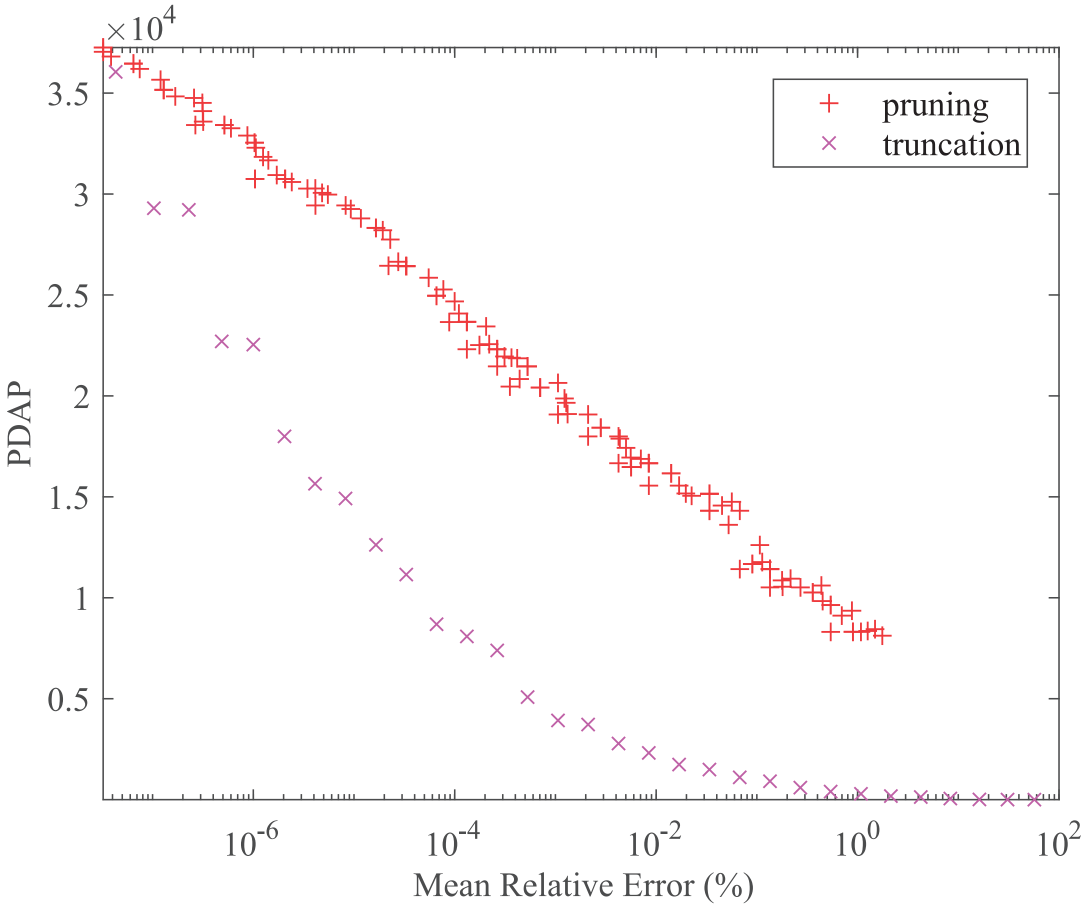


Figure 3.10 – Pruned and truncated 32-bit adders synthesized in a UMC 65 nm technology at 3 GHz. Error characterization with a set of 5 million uniformly distributed random inputs.

With these error metrics, truncation is way more efficient than pruning. All the truncated adders have a lower mean relative error and occupy less silicon area than the pruned adders. It should nonetheless be noted that the design space is much wider with pruning. Indeed, there is a large number of pruned adders that can fit any mean relative error specification between 10^{-8} % and 10 %. This is particularly true for low errors where only five truncated adders have MRE below 10^{-7} % Fig. 3.10 whereas at least 20 pruned adders can fit the same error specification. On Fig. 3.10, it can clearly be seen that the tools and design kit are optimized to produce adder architectures with an even wordlength. On the left side, truncated adders are grouped by pair meaning that designs with a n odd number of bits cost as much in terms of PDAP than the adder with one bit more. This trend is however not visible anymore for lower bitwidth since the adder are not anymore under a tight timing constraint. Since the mean relative error increases with the number of truncated bits, it can be seen that truncated adders in the 10 % MRE range only have 3 to 4 bits wordlength and therefore have a very small dynamic range. The same remarks are also valid with the maximum error distance Fig. 3.11.

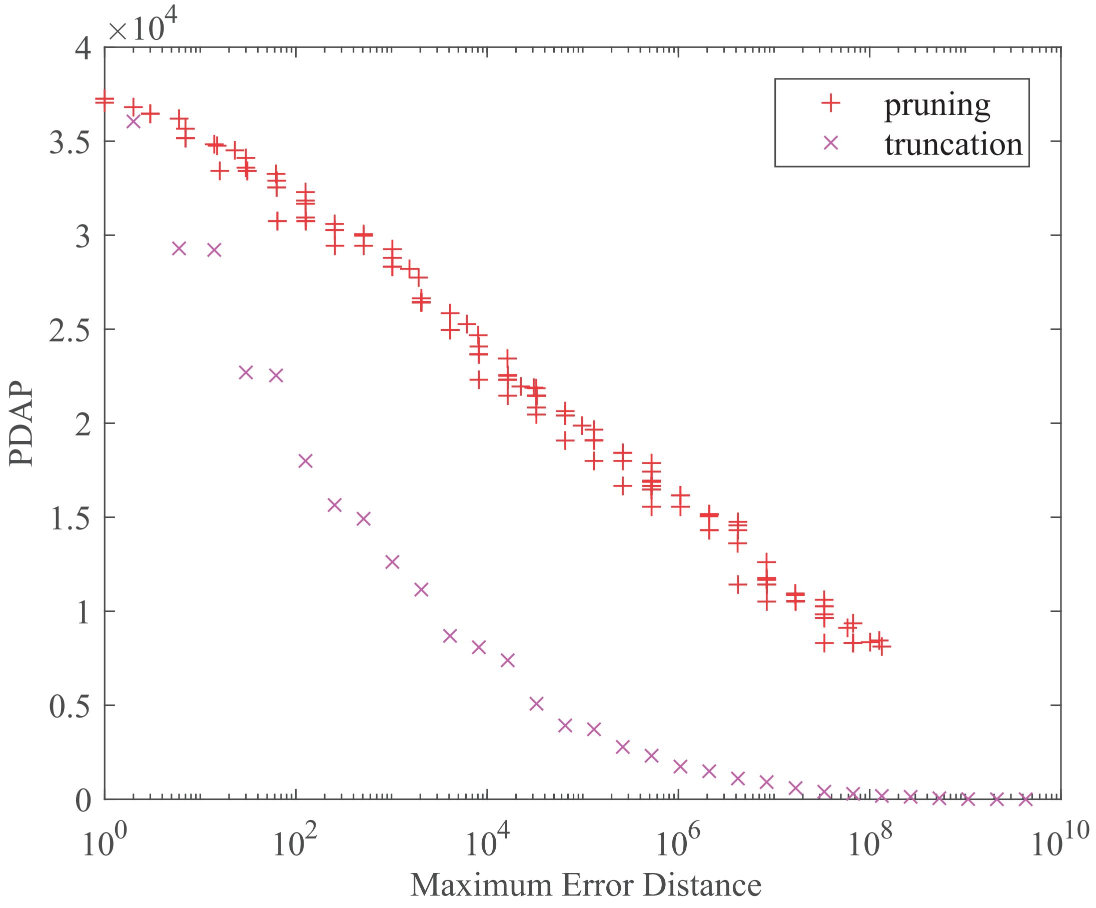


Figure 3.11 – Maximum Error of pruned and truncated 32-bit adders synthesized in a UMC 65 nm technology at 3 GHz. Error characterization with a set of 5 million uniformly distributed random inputs.

At 0.8 GHz as shown Fig. 3.12 and 3.13, the results are similar, the PDAP gap between pruning and truncation is however reduced. Again, pruning is particularly interesting in the ultra-low error range where almost no truncated adder are present.

The pruning results are rather disappointing when compared to truncation and can partially be explained by the fact that Electronic Design Automation (EDA) tools are optimized for truncation. On the on hand, synthesis tools are optimized for truncation, they are built to find the optimal architecture for any given bitwidth and timing constraints. On the other hand, pruning is applied after the synthesis step on a circuit which has an optimal architecture for the given bitwidth and timing constraint. The pruning will then degrade this architecture instead of rebuilding the entire circuit with a different circuit topology once the precision constraints are too far from the initial precision. This issue could potentially be solved by integrating the pruning algorithm

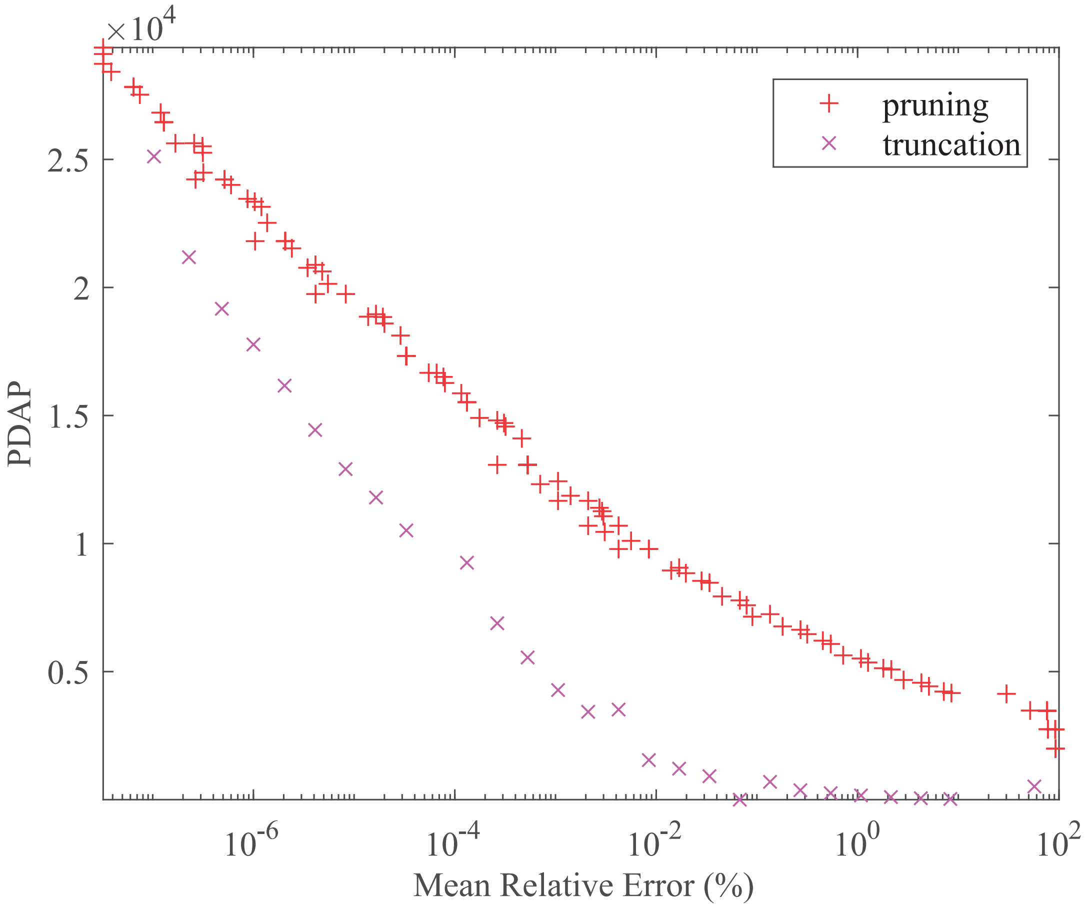


Figure 3.12 – Pruned and truncated 32-bit adders synthesized in a UMC 65 nm technology at 0.8 GHz. Error characterization with a set of 5 million uniformly distributed random inputs.

in the synthesis tool so that pruning is always applied on the optimal circuit topology for any given constraints.

This work compared pruning with truncation, but the same observations can be made for almost all approximate circuits presented in literature [1]. Truncation is almost always more efficient than approximate circuit techniques. Gate-level pruning is however still an interesting technique to use for fine tuning since it offers a wider design space than truncation where two consecutive bitwidth are usually separated by a wide gap in PDAP and error characteristics. The general methodology for designing approximate arithmetic circuits could therefore be:

1. Find the minimum bitwidth for the application under test.
2. Apply GLP for fine tuning.

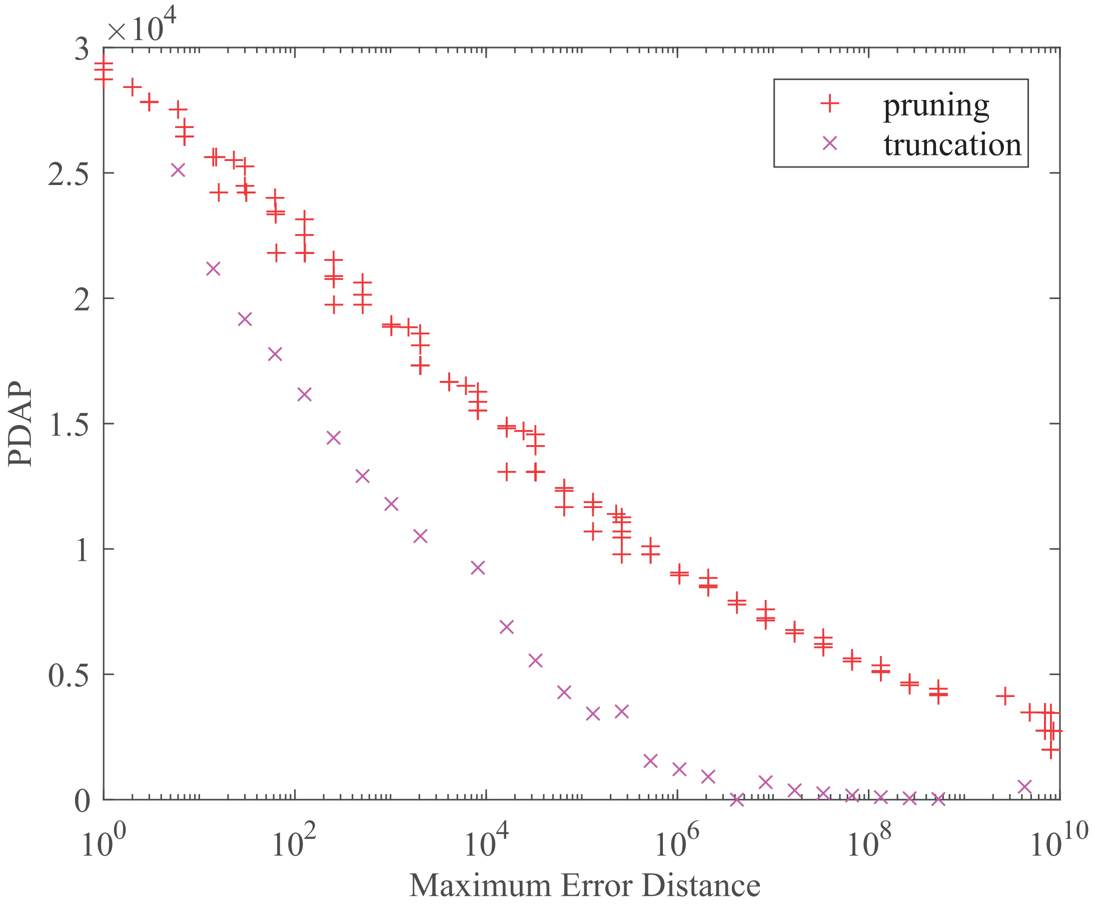


Figure 3.13 – Maximum Error of pruned and truncated 32-bit adders synthesized in a UMC 65 nm technology at 0.8 GHz. Error characterization with a set of 5 million uniformly distributed random inputs.

3.3 Concluding remarks

This chapter presented an automatic Gate-Level pruning tool, which is fully integrated in the standard digital flow. This provides the designer a wide range of energy-accuracy tradeoffs for arithmetic circuits and more generally for any combinational circuit as demonstrated in [4, 2]. It is particularly efficient on circuits in which there is a notion of bit significance. Indeed, it is possible to achieve up to 78 % PDAP reduction at a 10 % mean relative error for 32-bit adders. Unfortunately, at such high error levels, truncation is way more efficient than the pruning technique since 3 to 4-bit adders can achieve the same error level. Moreover, arithmetic circuits are usually very small and consume very little energy compared to other elements such as memories and registers. However GLP is still interesting for very low error levels as it offers a multitude of tradeoffs between two consecutive truncated adders. The general design methodology should therefore consist in first using truncation to find the optimal bit-width and remove unnecessary registers and memories, and then apply GLP for fine tuning. This two-step design methodology is certainly also valid for approximation techniques other than GLP.

This work is based on the significance and activity criteria and reverse graph traversal to search which gates to prune, but different criteria and search method could be tested to improve the pruning efficiency. The significances could for instance be propagated from the primary inputs of the circuit, or one could try to perform an exhaustive search by pruning each gate one by one and evaluate the resulting error and circuit savings.

Chapter 5 will give more details on how to achieve gains on full systems composed of multiple registers and memories using gate-level pruning on arithmetic circuits. This work also does not address formal verification, which is generally very challenging for any approximate circuit. The functionality of the circuit is tested by gate-level simulations, which is a cumbersome and time-consuming process. To enable the industrial use of approximate circuits and to speed-up design time, new verification techniques would have to be developed.

References

[1] Benjamin Barrois, Olivier Sentieys, and Daniel Menard. The hidden cost of functional approximation against careful data sizing–a case study. In *EEE/ACM Design Automation and Test in Europe (DATE)*, 2017.

[2] Vincent Camus, Jeremy Schlachter, Christian Enz, Michael Gautschi, and Frank K Gurkaynak. Approximate 32-bit floating-point unit design with 53% power-area product reduction. In *European Solid-State Circuits Conference, ESSCIRC Conference 2016: 42nd*, pages 465–468. IEEE, 2016.

[3] A Lingamneni, C. Enz, J. L Nagel, K. Palem, and C. Piguet. Energy parsimonious circuit design through probabilistic pruning. In *Design, Automation Test in Europe Conference (DATE), 2011*, pages 1–6, March 2011. doi: 10.1109/DATE.2011.5763130.

[4] Jeremy Schlachter, Vincent Camus, and Christian 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*, pages 476–480. IEEE, 2015.

4 Inexact Speculative Adders and Gate-level Pruning

In opposition to gate-level pruning which consits in removing logic gates from a circuit which is almost final, speculative adders [6] rely on architectural changes. They 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 subpaths 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 with different approaches in order to reduce the error frequency or magnitude. The ETAII adder[9] consists of regular sub-adder blocks with input carries speculated from Carry Look Ahead (CLA) blocks of the same length. In the ETAIIM version, several of the most significant CLA blocks are chained to increase accuracy. The ETBA adder[7], direct descendent of the ETAIIM, adds variable speculation signs and sub-adder sum balancing multiplexed blocks to mitigate relative errors. The ETAIV[10] and CSA[5] adders have enhanced accuracy by considering two prior carry speculation blocks instead of one, coupled respectively with a carry select or a carry skip technique, with the latter also using sum balancing over several sub-adder blocks. On the other side, ISA[1] and CSC[3] adders have recently improved circuit performance and efficiency by introducing off-critical path error reduction techniques. The ISA adder concept[1] has also proposed an optimal and generalized approach of speculative compensated adders, encompassing aforementioned adders, and has introduced a simple methodology to allow designers to generate efficient architectures from a delay-accuracy specification.

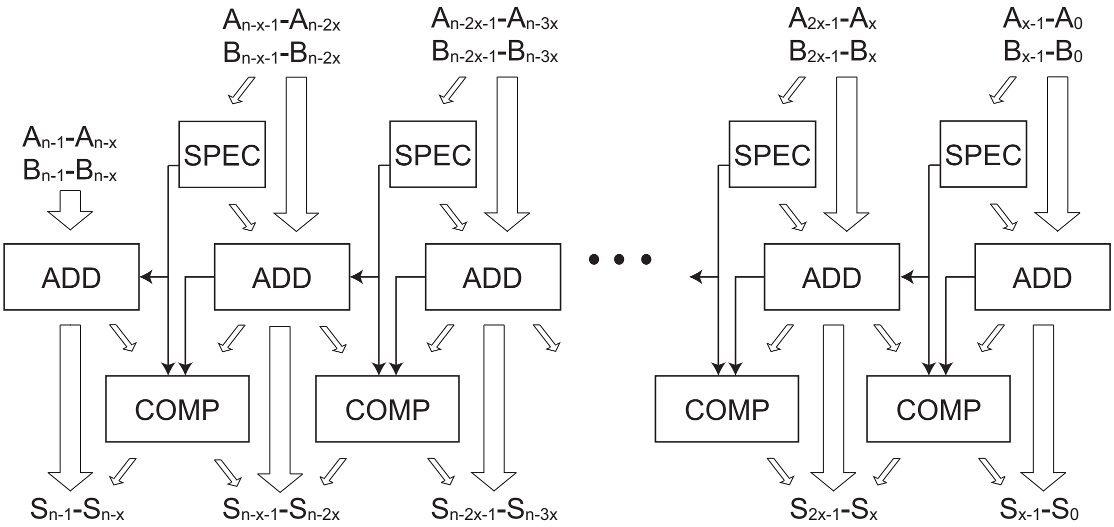


Figure 4.1 – General block diagram of an Inexact Speculative Adder (ISA). Each speculative segment consists of a carry speculator (SPEC), a regular adder (ADD) and an error compensation block (COMP).

4.1 The inexact speculative adder

General Concept

The general block diagram of an Inexact Speculative Adder (ISA) adder is depicted in Fig. 4.1. 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 [9], the probability of fault decreases when increasing the size of this block.
- *ADD* – The sub-adder block calculates local sums from the partial operands and 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 illustrated in Fig. 4.2 is a five 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 subadder 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.

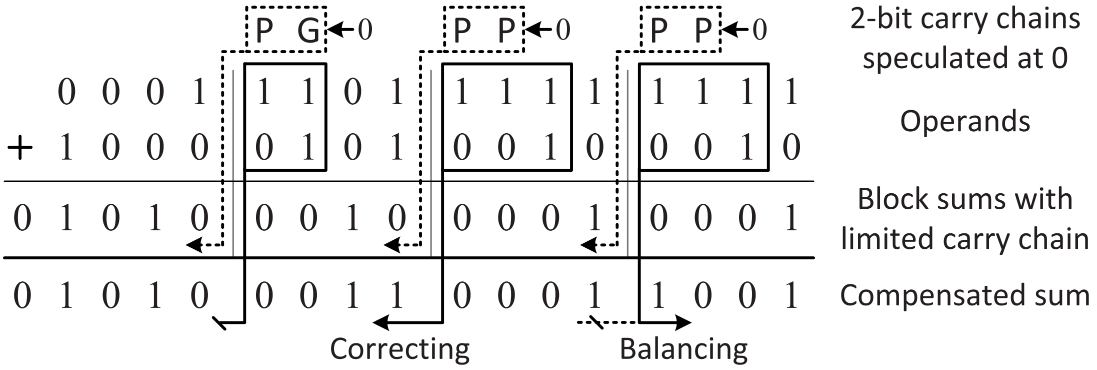


Figure 4.2 – Example of ISA addition arithmetic with 2-bit speculation, 1-bit correction and 1-bit error reduction. Faults only occur in the two right-hand paths. The 1st LSB of the central path can be corrected. The 1st LSB of the right path cannot be corrected, so the 1st MSB of the preceding sum is flipped.

4.1.1 Error Compensation

The COMP's error correction technique, introduced in [1], consists in incrementing or decrementing only a small group of LSBs of the local sum in order to compensate the erroneous speculated carry. In most cases, this can fully resolve carry errors. In the case where those stages are all in propagate modes, correction is impossible as it would lead to an internal overflow. In that case, the uncorrected bits, having a higher significance than the error bit, ensure a low relative error of the result. Using the COMP's error correction technique thus reduces both error rate and relative error. The correction hardware is executed concurrently to the local addition, thus this technique impacts minimally the critical path of the adder.

The COMP's error reduction technique consists in balancing a group of MSBs of the preceding sub-adders in opposite direction than the error. This technique, similarly as in [7], has been intensively employed in literature. But to avoid high relative errors and better control the worst-case error (RE_{MAX}), it relies on large SPEC block directly lying in the critical path of the adder.

4.1.2 Design Strategy

The general design methodology for ISAs is to synthesize and simulate the error characteristics of all possible block sizes for SPEC ADD and COMP for a given timing constraint. By doing so, the designers can be sure to find the best ISA architecture for a given timing and a given error threshold. The main drawback however is that the number of block sizes combination is huge. For a 32-bits ISAs the number of synthesis and error simulations to perform can easily reach hundreds of thousands. In an academic research environment, CAD tools licenses are cheap and several of them can be used for several weeks to try out all the possible ISA combinations. In an industrial environment however, this approach methodology cannot be applied for obvious costs reasons. In this regard, the following paragraphs present a design methodology to quickly obtain satisfying error characteristics and PDAP and energy savings using the ISA.

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.

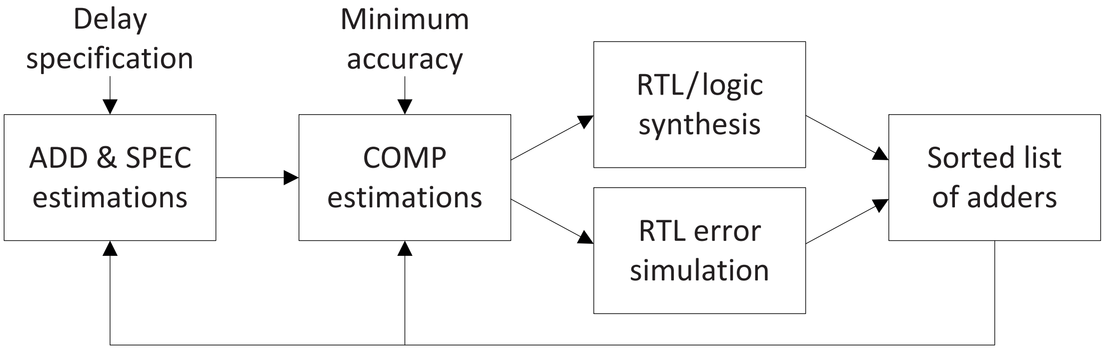


Figure 4.3 – CAD framework for ISA design.

A design methodology through a delay-accuracy approach is presented in Fig. 4.3. The adequate delay tradeoff 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.

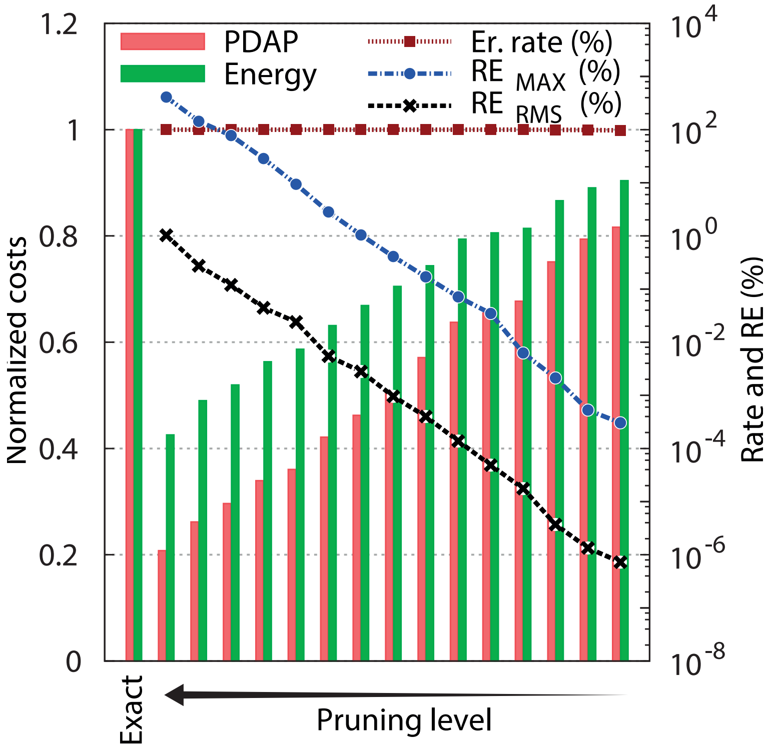
Adders in literature describe particular cases of implementation excessively considering either performances or errors. In the ISA architecture, the speculation overhead can be traded for longer sub-adders while fitting the same delay requirement. It is then possible to use fewer speculative paths and 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[1].

4.2 Comparison of gate-level pruning and ISA

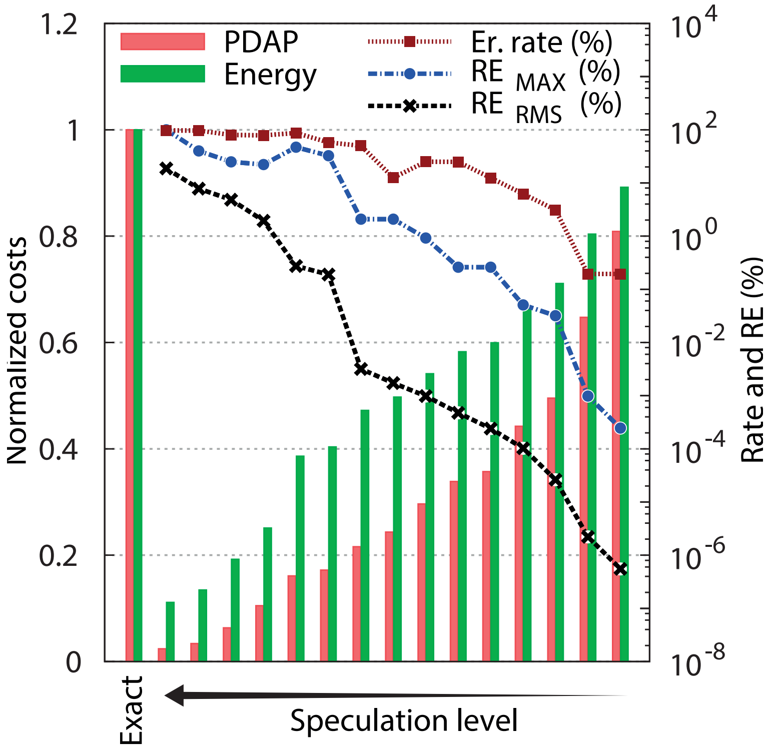
4.2.1 Pruning or Speculation Applied Individually

In order to perform a comparative study, both techniques have been applied on 32-bit adders synthesized in a 65 nm UMC technology library and all the designs have been simulated with a set of five million uniformly distributed random inputs. Comparisons of error characteristics and normalized costs in terms of energy and Power-Delay-Area Product (PDAP) are shown for a selection of pruned and speculative adders synthesized at 3.3 GHz in Fig. 4.4a and 4.4b and synthesized at 1.3 GHz in Fig. 4.5a and 4.5b. For the pruned adders, each couple of bar shows the PDAP and the energy consumption of a pruned design. While the bars on the right of the figure represent adders with a small number of pruned nodes,the bars on the left represent those with a high number of pruned nodes, and therefore significant circuit savings at the cost of relatively high error levels. Similarly, speculative adders with low level of errors and low circuit savings a represented on the right of Fig. 4.4b. Those ISAs achieve low error levels thanks to a low number of carry cuts and large sub-adder blocks. In opposition, the ISAs on the left of the figure have smaller sub-adder blocks and a higher number of carry cuts resulting in higher errors and higher PDAP and energy savings.

Only speculative adders with regular structures have been synthesized (i.e. 2x16, 4x8, 8x4 and 16x2 bit concurrent paths) with diverse error characteristics. For this reason, the displayed characteristics Fig. 4.4b and 4.5b present steps corresponding to changes of structure sizes. Generally, ISA adders built out of small sub-adders show high errors and high savings (on the left of the figures), whereas ISA with large sub-adders are preferred for low errors and lower savings (on the right of figures).

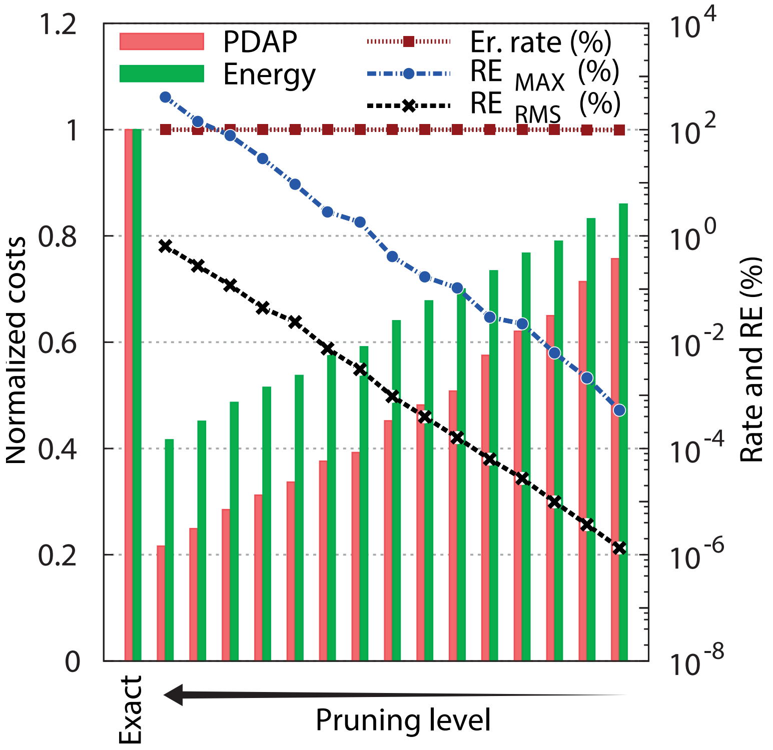


(a) Pruned adders at 3.3 GHz

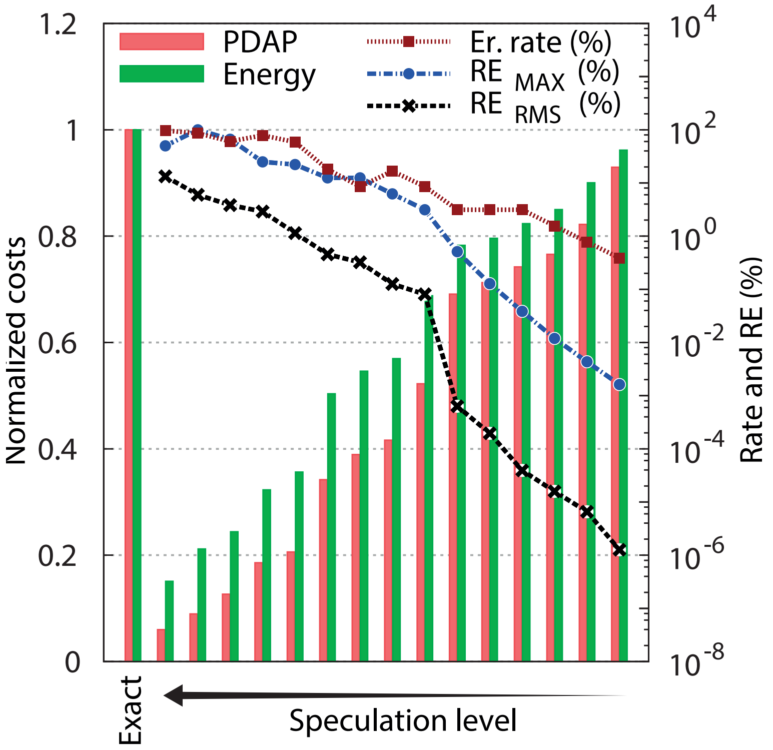


(b) Speculative adders at 3.3 GHz

Figure 4.4 – Error characteristics and normalized cost of 32-bit pruned and speculative adders synthesized at 3.3 GHz



(a) Pruned adders at 1.6 GHz



(b) Speculative adders at 1.6 GHz

Figure 4.5 – Error characteristics and normalized cost of 32-bit pruned and speculative adders synthesized at 1.6 GHz

Fig. 4.4 and 4.5 clearly show that GLP and ISA 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.

For both techniques and frequencies, the RE_{RMS} and the RE_{MAX} grow with an exponential trend versus circuit savings. Only the ISA adders on the left of the figures have a low RE_{MAX} and do not follow the same exponential trend as it is expensive to control when the constraints on the circuit become too high. This explains the gap between RE_{MAX} and RE_{RMS} when RE_{MAX} is lower than 1 %. This gap is then drastically reduced for when RE_{MAX} is higher than 1 % due to the tight constraints on the circuit, which lead to architectures with a higher number of carry cuts.

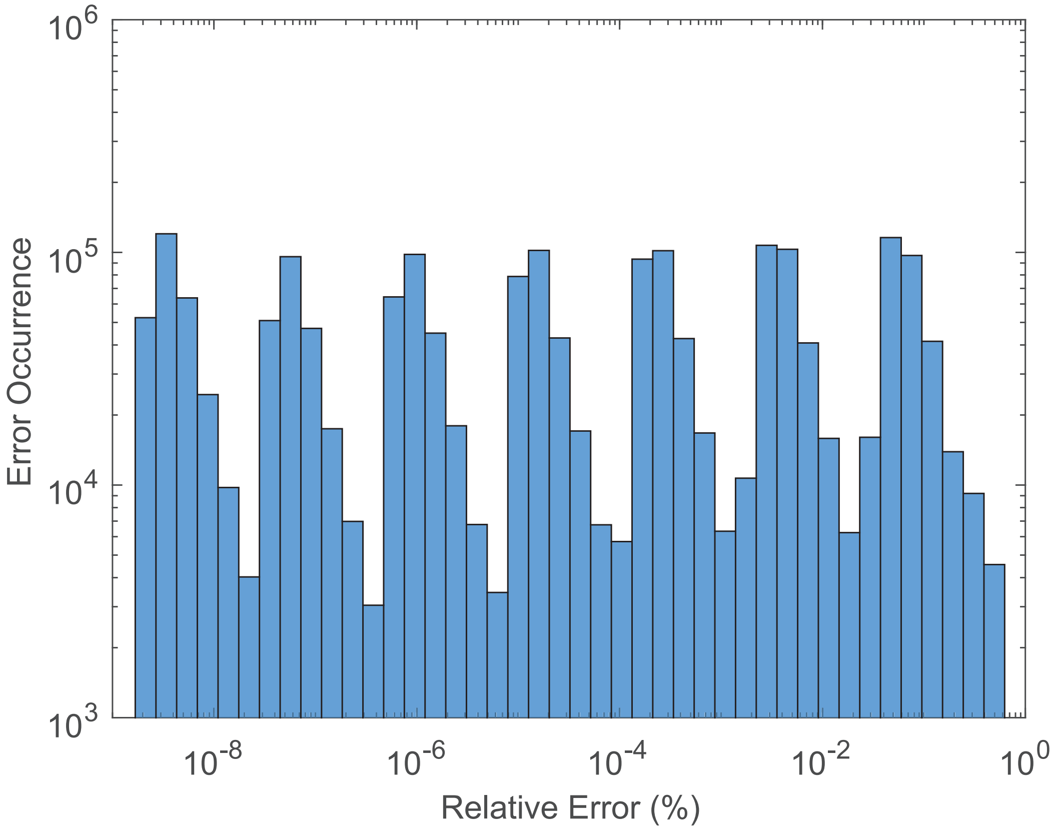


Figure 4.6 – Error distribution of a 32-bit ISA synthesized at 1.25 GHz

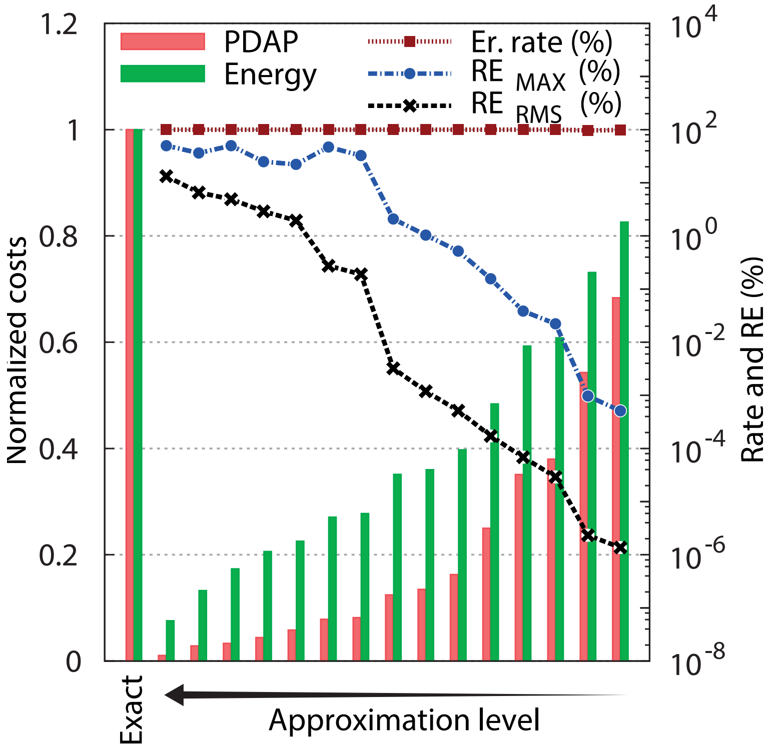
Timing constraint has a significant influence on the results obtained with the two techniques. Fig. 4.4a and Fig. 4.4b show that at high frequency of 3.3 GHz, and for a relative PDAP cost of 0.42, the RE_{MAX} and the RE_{RMS} of the pruned adder are equal to 4 % and 0.008 % respectively. In comparison, the speculative adder having a similar

PDAP has a RE_{MAX} of 10^{-1} % and a RE_{RMS} of 10^{-4} %. This could lead to the conclusion that the speculation technique can achieve similar energy savings than the pruning technique, at a much higher accuracy level. However, Fig. 4.5a and Fig. 4.5b actually depict the opposite trend when using a slightly lower frequency of 1.6 GHz. Hence, the gains obtained through gate-level pruning and circuit speculation are very much dependent on the timing constraints applied to the circuit.

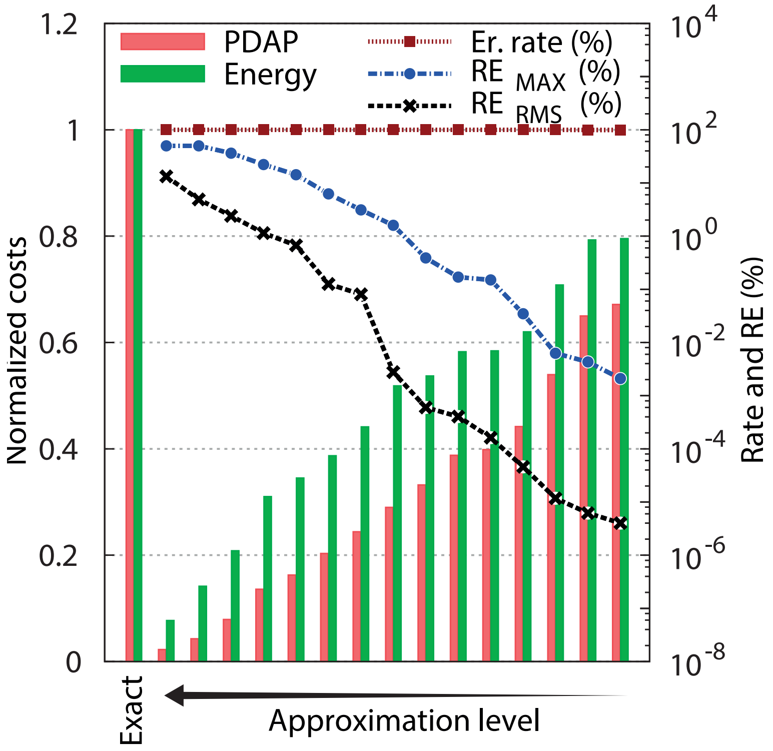
Furthermore, the two techniques produce error of different nature, ISAs tend to produce rare errors with a low magnitude which can appear on significant bits if the carry chain is cut at this position. GLP leads to high error rates, but the errors generally occur on the least significant bit positions. This can also be seen in Fig. 4.6 which plots the error distribution of a 32-bit ISA synthesized at a frequency of 1.25 GHz and which has been simulated with a set of 5 million random uniformly distributed inputs. The error occurrence peaks in multiple relative error positions, which correspond to the bit positions where the carry chain is cut. Here the errors are distributed over the full range (10^{-9} % to 10^{-0} %) whereas for pruning, most of the errors occur in the low relative error range as shown Fig. 3.5. For these reasons, it is worth trying to combine ISA and GLP to get even more circuit savings for similar error levels.

4.3 Mixing of Pruning and Speculation

All the previous sections were discussing the individual use of the pruning and the speculation techniques on 32 bit adders, once at a frequency of 3.3 GHz, and once at 1.6 GHz. Both techniques lead to errors of different nature, it is therefore worth combining the two techniques. Since the pruning methodology is only one additional step in the standard digital flow, the general methodology to combine speculative circuits and pruning is: 1. Synthesize speculative circuits at the desired speed constraint starting from the HDL description. 2. Apply the pruning methodology to the gate-level netlists of the speculative circuits.



(a) Mixed adders at 3.3 GHz



(b) Mixed adders at 1.6 GHz

Figure 4.7 – Error characteristics and normalized cost of 32-bit pruned and speculative adders synthesized at 1.6 GHz

Fig. 4.7a and Fig. 4.7b show the normalized costs as well as the error characteristics of *pruned speculative* (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 of 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. 4.7a and 4.7b have a lower PDAP cost and consume less energy than the pruned or the speculative adders.

Table 4.1 summarises the PDAP reduction obtained through the use of the three techniques, namely pruning, speculation and the mixing of both. It is shown that a *pruned speculative* adder with only 2 % relative error magnitude has a normalised 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 4.1 – Comparison of the three inexact design techniques.

Specifications		Normalised PDAP		
Frequency	RE_{RMS}	Pruning	Mixed	Speculative
3.3 GHz	$3 \cdot 10^{-6}$ %	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

4.4 Comparing approximate adders and truncation

Since truncation is certainly the most simple way to reduce precision in arithmetic circuit, speculative adders need to be compared with truncated adders. Fig. 4.8 compares state-of-the-art 32-bit approximate and truncated adders [2], including the Inexact Speculative Adder (ISA) and pruned adders (GLP) at a frequency of 800 MHz. The

ETBA, ETAII, GCSA [7, 9, 3] and ISA are all speculative adders. The ACA [4] is also a speculative adder but with some additional timing optimizations and the LOA [8] is a truncated adder with OR gates placed on the truncated bits and behaving like 2-bit adders without carry out.

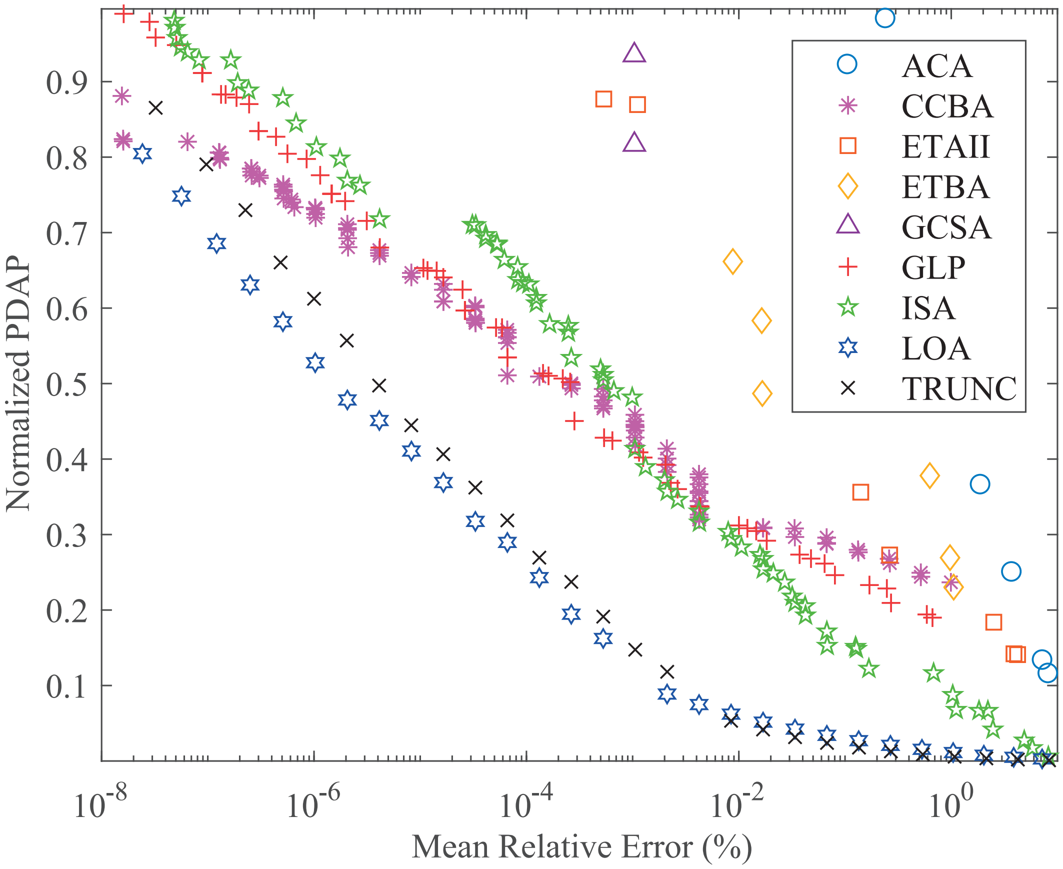


Figure 4.8 – 32-bit state-of-the-art approximate adders compared with truncated adders synthesized at a frequency of 800 MHz

Again, truncation clearly outperforms most of the approximate adders. On the one hand, some architectures such as the ETAII, the ETBA, the GCSA and the ACA lead to very inefficient circuit implementations: for a mean relative error of $10^{-}3$ those approximate adders have a PDAP which is up to 9X higher than the PDAP of truncated adders. On the other hand, the ISA, GLP and CCBA are more efficient and can even beat truncation for low error levels. Only the LOA is slightly more efficient than truncation simply because the LOA is a truncated adder with OR gates placed on the truncated bits, reducing the error with a minimal overhead. The mixed adder is not plotted on this graph, but the data shown Table 4.1 could be extrapolated and would certainly show that mixed adders are more efficient than ISA or GLP adders, but still more expensive than truncated adders.

4.5 Concluding remarks

This chapter presented the inexact speculative adder in which the carry chain is cut to relax the timing constraints. The carry is speculated using speculation blocks and error resulting from the carry cutting are partially compensated using compensation blocks. The ISA and the GLP technique can achieve comparable PDAP savings of up to 80 % for a 1 % RE_{RMS}, but most importantly, ISA and GLP produce errors of different nature thanks to their radically different approximation approach. Thus, it has been shown that combining both techniques by applying gate-level pruning on speculative adders results in even higher PDAP savings of up to 95 % for 2 % RE_{RMS} at 3.3 GHz. Hence, approximate adders should exploit two or even more approximation techniques such as the mixed adder for instance, in order to take advantage of the fact that different approximation techniques lead to errors of different nature, but still lead to a massive PDAP reduction.

Unfortunately, truncation outperforms GLP and ISA and still remains the most efficient approximation technique, especially for high error levels. Only the LOA is slightly more efficient than truncation, but this is because the LOA is a truncated adder with OR gates placed on the truncated bits to limit the error magnitude with a minimal overhead. In this regard, the design procedure to build an efficient approximate adder would be to first use truncation or the LOA, and then apply one or more approximation techniques such as GLP or ISA for fine-tuning. Note that using truncation as a first step also allows to get rid of some registers, and thus lead to system level savings. This is not the case for the LOA as it preserves the LSBs, and therefore does not allow to reduce the number of registers.

References

[1] Vincent Camus, Jeremy Schlachter, and Christian Enz. Energy-efficient inexact speculative adder with high performance and accuracy control. In *Circuits and Systems (ISCAS), 2015 IEEE International Symposium on*, pages 45–48. IEEE, 2015.

[2] Vincent Camus, Jeremy Schlachter, Mattia Cacciotti, and Christian Enz. Timing optimization using artificial false paths and application with the carry cut-back adder. In *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS), submitted*, 2018.

[3] 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.

[4] Andrew B Kahng and Seokhyeong Kang. Accuracy-configurable adder for approx-

imate arithmetic designs. In *Proceedings of the 49th Annual Design Automation Conference*, pages 820–825. ACM, 2012.

[5] Yongtae Kim, Yong Zhang, and Peng 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*, pages 130–137, Nov 2013. doi: 10.1109/ICCAD.2013.6691108.

[6] Tong Liu and Shih-Lien Lu. Performance Improvement with Circuit-level Speculation. In *Microarchitecture, 2000. MICRO-33. Proceedings. 33rd Annual IEEE/ACM International Symposium on*, pages 348–355, 2000. doi: 10.1109/MICRO.2000.898084.

[7] Matthew Weber, Mateja Putic, Hang Zhang, John Lach, and Jiawei Huang. Balancing adder for error tolerant applications. In *Circuits and Systems (ISCAS), 2013 IEEE International Symposium on*, pages 3038–3041. IEEE, 2013.

[8] Zhixi Yang, Ajaypat Jain, Jinghang Liang, Jie Han, and Fabrizio Lombardi. Approximate xor/xnor-based adders for inexact computing. In *Nanotechnology (IEEE-NANO), 2013 13th IEEE Conference on*, pages 690–693. IEEE, 2013.

[9] Ning Zhu, Wang Ling Goh, and Kiat Seng Yeo. An enhanced low-power high-speed adder for error-tolerant application. In *Integrated Circuits, ISIC'09. Proceedings of the 2009 12th International Symposium on*, pages 69–72. IEEE, 2009.

[10] Ning Zhu, Wang-Ling Goh, Gang Wang, and Kiat-Seng Yeo. Enhanced Low-power High-speed Adder for Error-tolerant Application. In *SoC Design Conference (ISOCC), 2010 International*, pages 323–327, Nov 2010. doi: 10.1109/SOCDC.2010.5682905.

5 Approximate applications and accelerators design

5.1 Discrete Cosine Transform

Chapter 3 has shown that the GLP enables large power and area savings when applied to arithmetic circuits. However, one single adder generally only represents a tiny fraction of the area and power consumption of the system it is placed in. For this reason, even 50 % power and area savings achieved on a single adder could turn out to be insignificant at system level, and would not justify the quality loss. Nevertheless, this approach becomes more interesting at the level of a hardware accelerator dedicated to one specific task and which is built out of multiple arithmetic circuits. In this regards, this section analyses how the GLP can be applied simultaneously on several adders and subtractors used to build a Discrete Cosine Transform (DCT), which is one of the most computationally intensive element for many image and video processing compression algorithms such as JPEG or MPEG. This work does not claim or present a novel type of DCT, but it demonstrates how energy-quality tradeoffs can be achieved by applying inexact design techniques, such as GLP, on existing state-of-the-art architectures.

5.1.1 Conventional DCT

DCT algorithms and architectures have been extensively studied in the literature. Even error resilient DCTs have already been proposed [8, 3]. Image encoding algorithms used for instance in JPEG encoding generally compute the DCT per pixel blocks. The following work considers the example of 8x8 pixel blocks DCT, but could be extended to other block sizes and architectures. Efficient implementations are generally based on distributed arithmetic computations [14], and is taken as starting point for the following example, but the proposed methodology could be applied to any existing architecture to trade accuracy of computation against significant area and power savings.

A 2D DCT used in image encoding can be split in two single stage 1D DCTs interleaved

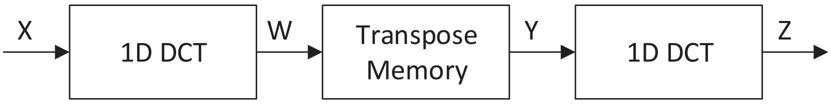


Figure 5.1 – 2D DCT architecture based on 1D stages

with transpose memory as shown in Fig. 5.1. The 8-point 1D-DCT w_k of a data sequence x_i is defined by

$$w_k = \frac{a_k}{2} \sum_{i=0}^{7} x_i \cos\left[\frac{(2i+1)k\pi}{16}\right] \tag{5.1}$$

$$\text{with } a_k = \begin{cases} 1/2, & k = 0 \\ 1, & k = 1...7 \end{cases} .$$

This can also be expressed in its matrix form as

$$W = T \cdot X, \tag{5.2}$$

where T is an 8 x 8 matrix in the case of an 8 point DCT and X and W are row and column vectors. Using the symmetry property of T, (5.2) can be decomposed as follow for even / odd 1D DCT calculations

$$\begin{bmatrix} w_0 \\ w_2 \\ w_4 \\ w_6 \end{bmatrix} = \begin{bmatrix} c_4 & c_4 & c_4 & c_4 \\ c_2 & c_6 & -c_6 & -c_2 \\ c_4 & -c_4 & -c_4 & c_4 \\ c_6 & -c_2 & c_2 & -c_6 \end{bmatrix} \begin{bmatrix} x_0 + x_7 \\ x_1 + x_6 \\ x_2 + x_5 \\ x_3 + x_4 \end{bmatrix} \tag{5.3}$$

$$\begin{bmatrix} w_1 \\ w_3 \\ w_5 \\ w_7 \end{bmatrix} = \begin{bmatrix} c_1 & c_3 & c_5 & c_7 \\ c_3 & -c_7 & -c_1 & -c_5 \\ c_5 & -c_1 & -c_7 & c_3 \\ c_7 & -c_5 & c_3 & -c_1 \end{bmatrix} \begin{bmatrix} x_0 - x_7 \\ x_1 - x_6 \\ x_2 - x_5 \\ x_3 - x_4 \end{bmatrix} \tag{5.4}$$

where $c_k = \cos(\frac{k\pi}{16})$. It can be seen from (5.1) that the DCT is computationally intensive, and requires a large amount of multiplications which are power hungry. Plenty of DCT architectures have been proposed in the literature. However, since the scope of this work is to improve energy-efficiency, a low-power multiplier-less DCT architecture based on row-column parallel distributed arithmetic has been chosen. Fig. 5.2 shows this implementation of the 8 x 8 2D DCT where only 4 adders and 4 subtractors are required to compute the right part of (5.3) and (5.4). The final 1D DCT is obtained by looking-up pre-computed multiply and accumulate (MAC) coefficients stored in a Read-Only Memory (ROM). It can also be seen on Fig. 5.2 that the first stage DCT inputs are on 8 bits since pixels are coded on 8 bits whereas the inputs buses of the second stage are 11 bits large due the carry out of adders and due to the precomputed multiplications.

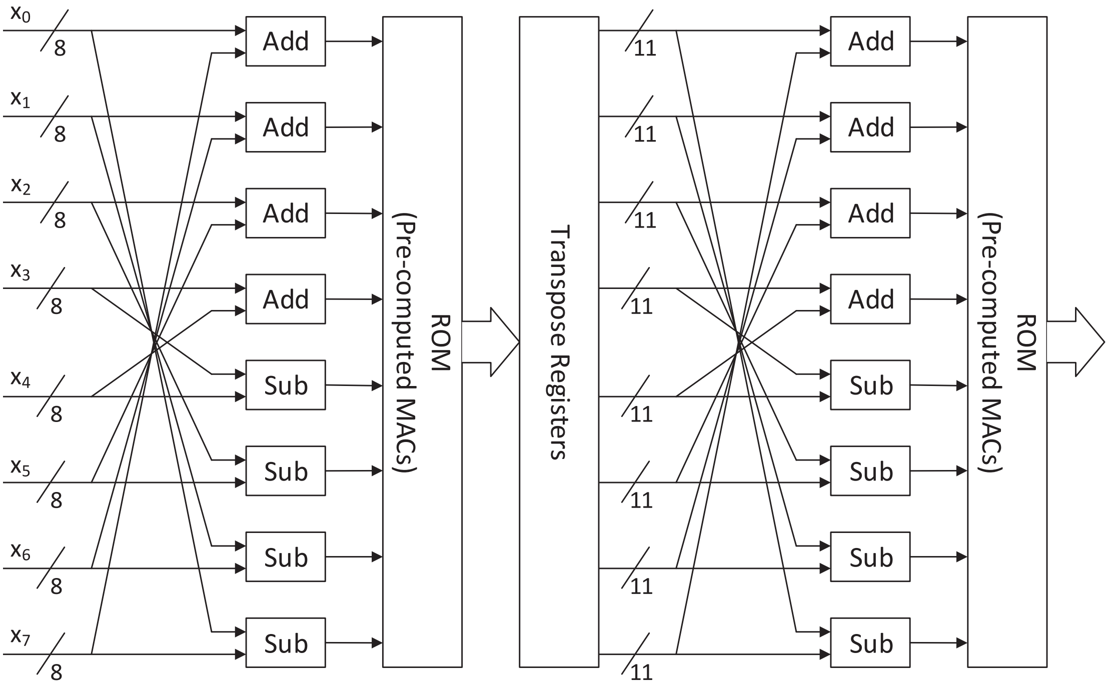


Figure 5.2 – Architecture of the 8 x 8 2D DCT.

5.1.2 Quality testing

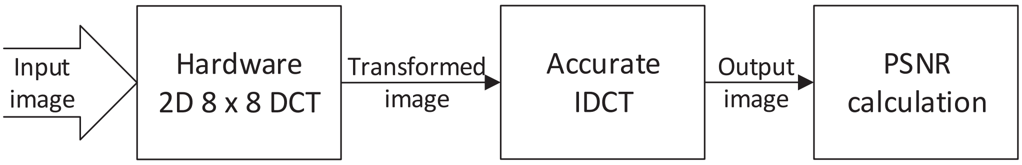


Figure 5.3 – Test setup for quality measurement

Fig. 5.3 sketches the test setup used to characterize the DCT for image processing.

First, the DCT of an image sample is computed with the hardware under test. Image is then reconstructed using a behavioral inverse transform, i.e. with infinite precision. The quality of the reconstructed image compared to the original image is evaluated by calculating the Peak Signal-to-Noise Ratio (PSNR) between the two images as follow:

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{D^2}{\mathrm{MSE}}\right) \tag{5.5}$$

where MSE is the mean squared error between the original and the reconstructed image and D is the maximum possible pixel value, here 255, considering 8-bit pixel representation. With a sample *Lena* picture transformed by the conventional 2D DCT shown in Fig. 5.2, the PSNR is equal to 48 dB. Image quality is limited mainly due to the use of fixed-point arithmetic. As conventional designs are already lossy, it can be acceptable to trade some more accuracy in exchange for power and silicon area savings. The following section presents how this can be done using the gate-level pruning methodology.

5.1.3 Pruning methodology

The 2D DCT described in section 5.1.1 has been synthesized with an industrial 65 nm technology at clock frequency of 1.25 GHz, which is close to the maximum frequency that can be attained with this architecture for this process. The resulting circuit is used as a reference to apply the Gate-Level pruning to each of the 16 adders and subtractors. Seeing that each of these components have slightly different architectures due to differences in timing paths, and considering that the switching activity differs from one to another, pruning is applied individually on each of the 16 operators. Besides, each can have a different impact on the final error bound. It is consequently required to explore the design space to find out the best possible combination of inexact adders in order to minimize the quality loss and maximize the savings. The synthesized adders and subtractors are built out of 45 standard cells in average. It is therefore worth pruning up to 10 nodes for fine-tuning the accuracy. Higher pruning would dramatically degrade the image quality. For 10 levels of pruning considered per adder and subtractor (the exact operator plus 10 pruned ones), there are 11^{16} possible design combinations. For practical reasons such as computing resources, it is clearly not possible to run 11^{16} synthesis and hardware simulations to find out the optimal design.

A good solution to narrow the design space is to apply the same level of pruning p_i to each adder and subtractor inside a given stage i. As the bit-width is the same within a stage, the degradation of arithmetic accuracy is progressive. With this approach, there are $11^2 = 121$ possible combinations left.

Synthesis shows that the area occupied by the 16 adders and subtractors depicted in

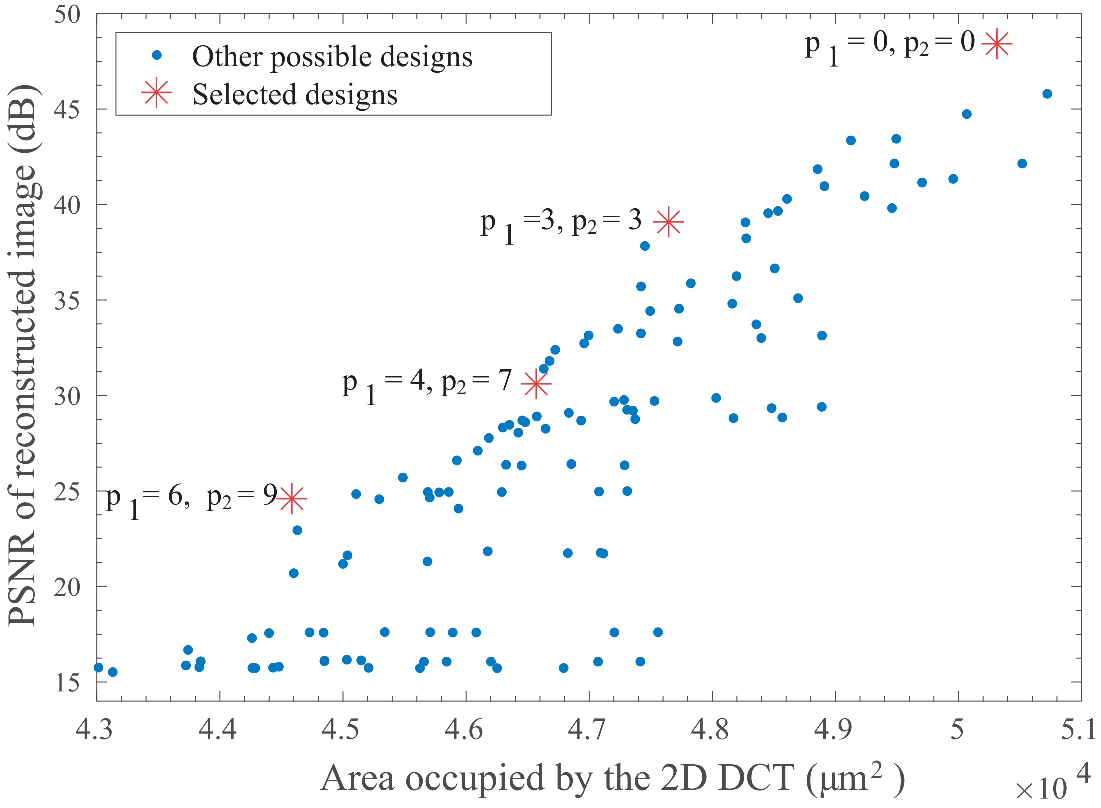


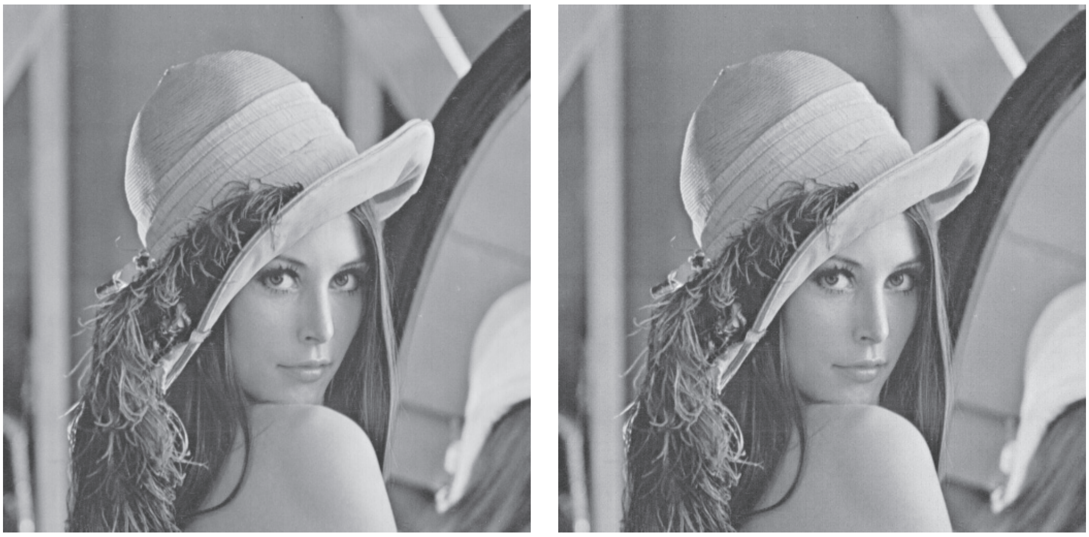
Figure 5.4 – Image quality versus circuit area

Fig. 5.2 represent a small part of the entire conventional 2D DCT: less than 4 % of the total area. Hence, a simple swap between exact and approximate operators would lead to very limited savings. Nevertheless, re-synthesizing the full design with pruned operators eliminates unused ROM and un-necessary registers thanks to logic simplification and constant propagation implemented in the synthesis tool. This results in attractive power and area savings that are presented in the following subsection.

5.1.4 Results

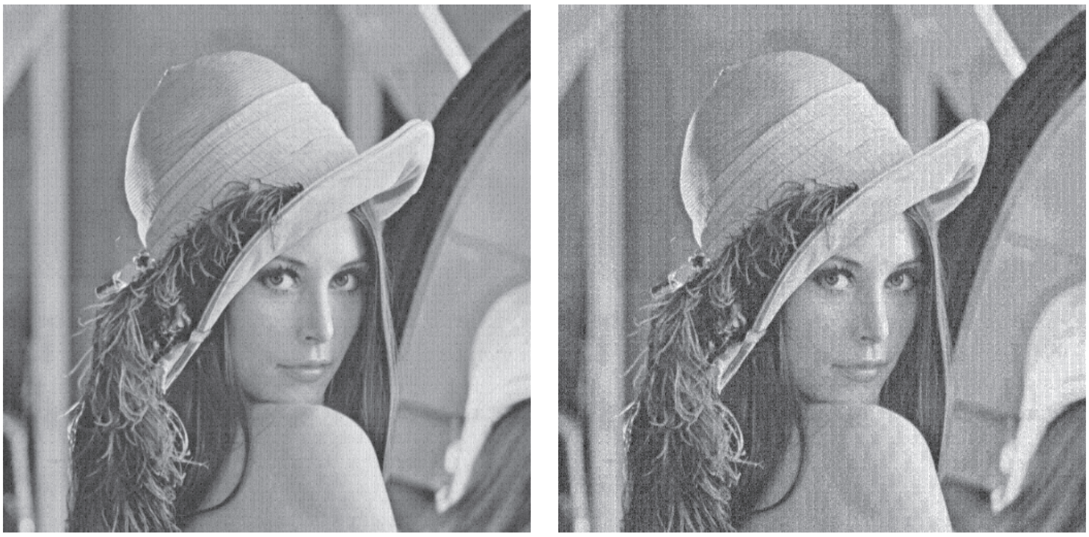
Fig. 5.4 shows the image quality versus area savings for the implemented DCTs. Each point corresponds to a combination (p_1, p_2) in $[0, 10]^2$. This figure highlights the broad diversity of design options offered using this methodology. For a given image quality requirement, pruning of operators in such a complex system allows to precisely match design specifications with an optimal circuit efficiency.

Keeping in mind that the goal of approximate circuits is to trade a little accuracy for the maximum area and power savings, only designs along the upper envelope of the plot in Fig. 5.4 are of interest since they maximize the gains with minimum quality loss.



(a) $p_1 = 0$ $p_2 = 0$ PSNR = 48.4 dB

(b) $p_1 = 3$ $p_2 = 3$ PSNR = 39.1 dB



(c) $p_1 = 4$ $p_2 = 7$ PSNR = 30.6 dB

(d) $p_1 = 6$ $p_2 = 9$ PSNR = 24.6 dB

Figure 5.5 – Pictures of Lena resulting from the test setup using the conventional DCT (a) and the approximate versions (b,c,d). p_i denotes the number of pruned nodes per adder and subtractor in stage i.

Table 5.1 – Power, area and quality of the 4 selected DCTs

Pruning level	PSNR (dB)	Normalized area			Normalized Power
		Arithmetic	Memory	**Total**	
$p_1 = 0$ $p_2 = 0$	48.4	1	1	**1**	**1**
$p_1 = 3$ $p_2 = 3$	39.1	0.94	0.95	**0.94**	**0.96**
$p_1 = 4$ $p_2 = 7$	30.6	0.82	0.93	**0.92**	**0.94**
$p_1 = 6$ $p_2 = 9$	24.6	0.72	0.89	**0.88**	**0.90**

Fig. 5.5 shows reconstructed *Lena* pictures obtained from four selected DCT implementations (the red stars in Fig. 5.4 highlight those designs). Conventional DCT has been used for Fig. 5.5a, while the three others have been obtained using three pruned designs representative of the area-accuracy tradeoff plotted in Fig. 5.4. On the one hand, it is possible to save up to 12 % area at the cost of almost imperceptible errors. On the other hand, for designs achieving the highest area reductions, artefacts start to appear on the edges of the 8x8 pixel blocks.

For the selected designs, power consumption is estimated based on gate-level simulations monitoring switching activity of the *Lena* picture processing. Results are summarized in Table 5.1. Despite adders and substractors represent less than 4 % of the overall DCT area, re-synthesis of the design with pruned operators enables larger savings over the entire system, as explained in Section 5.1.3. For the case ($p_1 = 6$, $p_2 = 9$), the arithmetic area is reduced by 28 % and the arithmetic power consumption is reduced by 46 %. Finally, the re-synthesis of the design with the pruned operators leads to 21 % energy-delay-area savings over the entire DCT. This significant overall saving is obtained thanks to the pruned arithmetic circuits, which set some nodes at constant values, enabling the synthesis tool to further simplify the circuit and memory using constant propagation. Indeed, the nets that are set to a constant value by the pruning tool can be used as starting points to propagate those constant values through the netlist to simplify or remove unnecessary gates or registers.

This section showed how gate-level pruning can be applied on a state-of-the-art discrete cosine transform, which is built out of multiple different arithmetic units and memories. The EDAP gains achieved for the specific adders and subtractors reach 46 % for an image quality loss of 24 dB. Despite the arithmetic circuits occupy less than 4 % of the total DCT area, the re-synthesis of the entire DCT with pruned operators leads to 21 % EDAP savings over the entire accelerator. This significant overall saving is obtained thanks to the pruned arithmetic circuits which sets some nodes at constant values, enabling the synthesis tool to further simplify the circuit and memory. The following section presents how gate-level pruning and inexact speculative adders can be applied to floating-point units and integrated into a small processor.

5.2 Tone-mapping of High Dynamic Range images with approximate floating-point units

The Floating-Point Unit (FPU) is one of the most common building block in any computing system and is used for a huge number of applications. At least one of them is integrated on each Central Processing Unit (CPU) and most of the microcontroller units. Finally, Graphics Processing Units are essentially built out of a huge number of floating-point units working in parallel. The IEEE standard for floating-point arithmetic (IEEE 754) defines the base 2 (binary) format representation of floating-point numbers as:

$$(-1)^S \cdot M \cdot 2^E \tag{5.6}$$

where S is the sign bit, M is the significand also called mantissa, and E is the exponent. The most common representation are:

1. **binary16**, commonly called *half precision* with 1 bit of sign 5 bits of exponent and 10 bits of mantissa.
2. **binary32**, commonly called *single precision* with 1 bit of sign 8 bits of exponent and 23 bits of mantissa.
3. **binary64**, commonly called *double precision* with 1 bit of sign 11 bits of exponent and 52 bits of mantissa.

From eq. 5.6 it can be seen that floating-point unit are more complex than integer or fixed-point adders as the mantissa and the exponent need to be calculated with distinct hardware. Exponents are always encoded on smaller bitwidth than mantissas, the hardware computing the exponent is therefore less constrained and the exponent calculation circuit is smaller than the one used for the mantissa. Moreover, errors that would occur in the exponent would have a dramatic impact. To that extent, this work aims at investigating the benefits of approximate circuits in the mantissa datapath of a FPU. The results will be evaluated with a High-Dynamic-Range image tone-mapping algorithm extensively using floating-point computations for contrast optimization and correction. Floating-Point Units have been implemented in a small multi-core processor to have a realistic view of the possible power savings and their corresponding error during *real-life* operations.

As depicted Fig. 5.6 the chip is based on the basic PULP architecture [2] with 4 Or10n cores, 16 kB of L2 memory, 16 kB of tightly coupled data memory (TCDM) organized

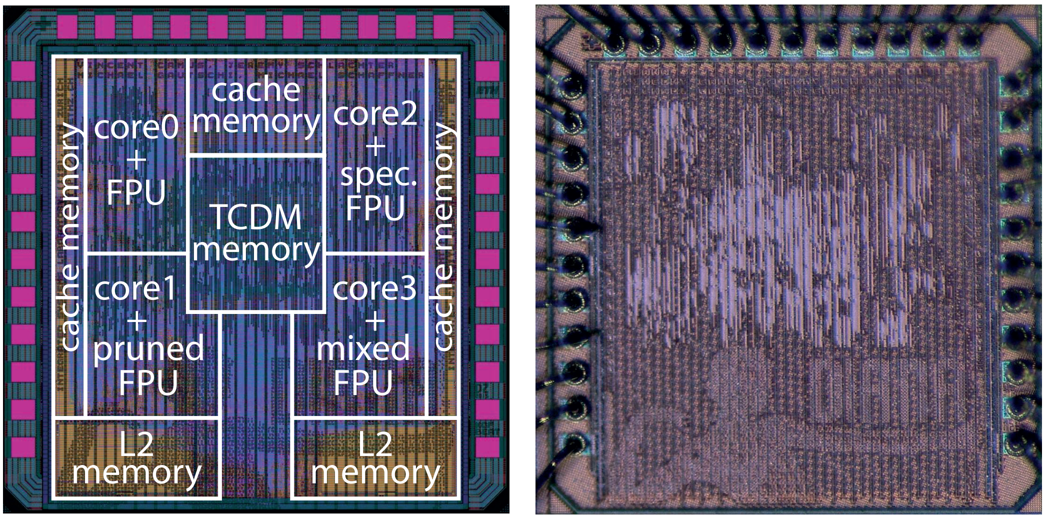


Figure 5.6 – Floorplan and die microphotograph of the chip. Die size is 1.56 mm^2.

into 8 banks and a total of 4 kB of instruction cache. Each of the four cores has a dedicated FPU capable of floating-point additions and multiplications with 2 cycles of latency. One of them is IEEE 754 single precision standard compliant. The three others are approximate variations of it. The chip is fabricated with the UMC 65 nm standard process technology and is designed to run at a maximum frequency of 500 MHz with 1.2 V power supply.

5.2.1 Approximate Floating-Point Units

The 4 FPUs share exactly the same architecture. The only difference between the exact one and the three inexact variations is that the original mantissa adder and mantissa multiplier have been replaced by approximate versions of it. In the FPU 1, gate-level pruning has been used to generate the approximate adder and multiplier. In the FPU 2 speculative adder and a novel speculative multiplier architecture presented in section 5.2.2 have been implemented, and in FPU 3, both techniques, speculation and pruning are combined to obtain even higher energy power and area savings. To ensure a minimal guaranteed precision, and in order to compare the three approximate arithmetic techniques, all approximate FPUs are implemented with at least 10 exact mantissa MSBs.

5.2.2 Inexact Speculative Multiplier

Multiplier circuits have 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), a new approximate multiplier circuit derived from error compensated spec-

Table 5.2 – 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

ulative architectures. 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 with several stages. The ISM is based on a two-stage multiplier architecture. First, a Partial Product Multiplier generates and merges partial products with a compressor tree into two partial sums. Then, an Inexact Speculative Adder (ISA) adds them in a speculative way in the last stage. This approach strongly reduces the overall critical path, and with a retiming step, used for instance in the case of pipelining, it significantly relaxes the timing constraints, leading to smaller overall area and power consumption. Sizing of the different speculative elements of the adder stage directly allows to trade worst-case and average errors in a delay-accuracy approach in the case of unsigned operation. In the case of two's-complement signed multiplication, 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 parameters are selected in the same approach as for unsigned operation. As the mantissa multiplier is in the critical path of the FPU circuit, even the slightest level of approximation can significantly relax the timing constraints. Moreover, the ISM error compensation and the 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'.

5.2.3 Error characterization

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 behaviour. The hardware used for floating-point additions / 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 3.

Table 5.2 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

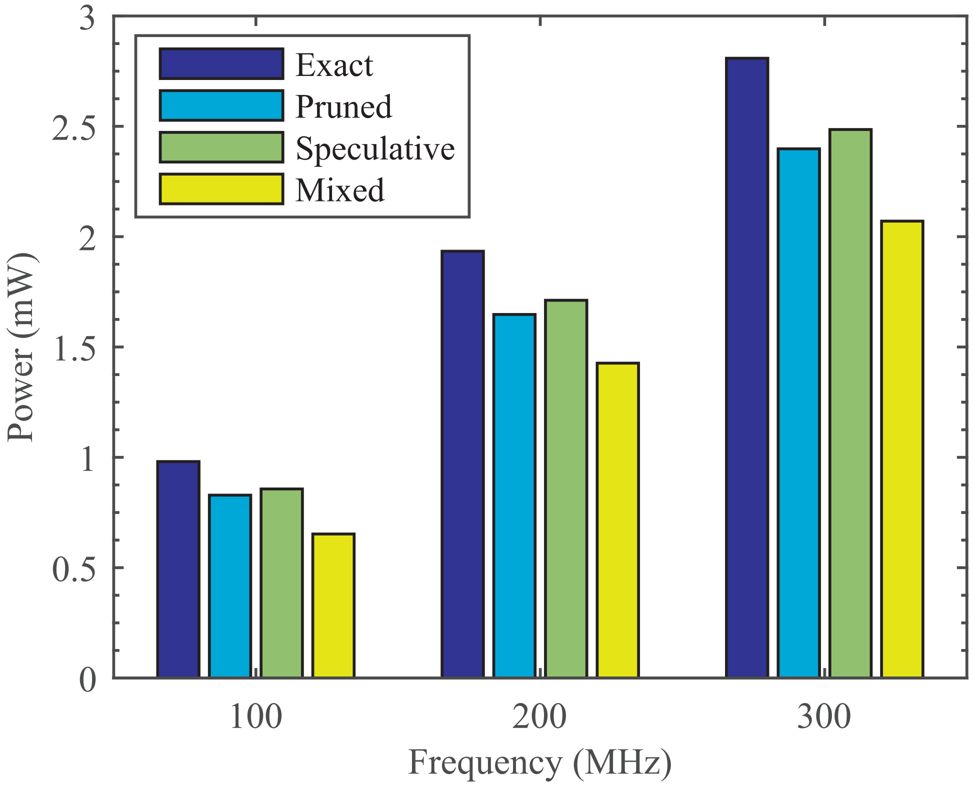


Figure 5.7 – Measured power consumptions of the 4 FPUs for 3 frequencies

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.

5.2.4 Power Measurements

The total power consumption of the chip has been measured by running a vector multiplication and addition benchmark, one core at a time. Then benchmark basically consists in random additions, multiplications and subtractions. By measuring the consumption of the entire chip, there was no noticeable power consumption difference between the exact core and the other 3 cores with approximate FPUs. Indeed, the overhead of the core and more importantly registers and memories are masking the gains obtained on the FPUs. This is expected since each floating-point unit occupies approximately 1 % of the total chip area. To evaluate the benefits of the approximate FPUs, their own consumption have to be measured. In order to be able to extract the consumption of a single FPU without the overhead of the core and memories, a second set of power measurement has been performed by running the same benchmark but with all the assembly floating-point add and multiply instructions replaced by No

Table 5.3 – Power, area and power-area product of the FPUs. Measured at 1.2 V, 300 MHz, room temperature in a UMC 65 nm technology.

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

Operations (NOPs). The power consumption of one FPU is therefore equal to the consumption of the chip while running the full benchmark minus the consumption of the chip while running the benchmark with NOPs. This test has been reproduced over 9 chips and 3 frequencies ranging from 100 MHz to 300 MHz [1].

Measurements results given table 5.3 and Fig. 5.7 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 activity criteria, pruning generally achieves better energy-efficiency than speculation at the cost of larger errors and higher silicon area. Combining pruning and speculation leads to 27 % power, 36 % area and 53% Power-Area Product (PAP) reductions for similar error levels thanks to their radically different approximation approach as discussed in chapter 4. Fig. 5.7 also shows that the relative power savings are independent of the frequency and that the power consumption increases proportionally with the frequency as predicted by the power consumption formula 2.1.

To demonstrate the functionality of such imprecise FPUs, and to compare the end-user impact of circuit speculation and gate-level pruning, HDR images have been tone mapped using non-linear masking [10] with each FPU individually. Unfortunately, this chip is equipped with only 16 kB of L2 memory, 16 kB of TCDM organized into 8 banks and a total of 4 kB of instruction cache, which is too small to execute an HDR tone-mapping algorithm on a descent size image. The largest image that could be tone-mapped with this configuration was in the order of 50 per 50 pixels large. The execution of this algorithm has therefore been simulated on the post-layout of the chip with artificially increased memory.

5.2.5 Tone-mapping Algorithm

HDR images pixel values are stored as single precision numbers and each of the steps involved in the tone mapping described in Fig. 5.8 requires a high number of floating-point addition and multiplication, some of them being looped. The first image

---

[1] Power measurements were not accurate above 300 MHz with the available tool.

normalization step basically consists in normalizing the highest pixel value to 1. This step therefore multiplies each of the pixels by a constant value. Then a 2 dimensional Gaussian is generated and is convoluted with each of the pixels during the tone mapping, involving several exponent and multiply and accumulate operations. This step is used to evaluate the contrast differences between one pixel and its neighbours. If the differences are too low, the contrast is optimized by applying the power function on each of the pixels. Finally, the brightness is adjusted by adding a constant value to each of the pixels. The chosen application is therefore computationally intensive and is a good indicator of the robustness of such approximate FPUs since large errors could have a significant impact on the final image. It further allows to maximise the power savings at the application level considering that most of the computations are floating-point operations.

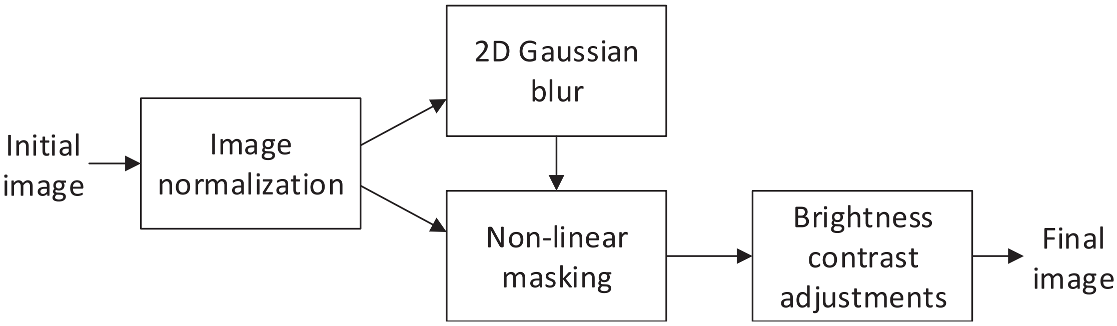
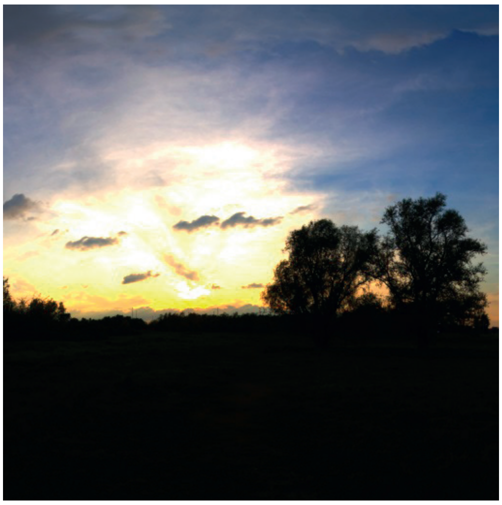


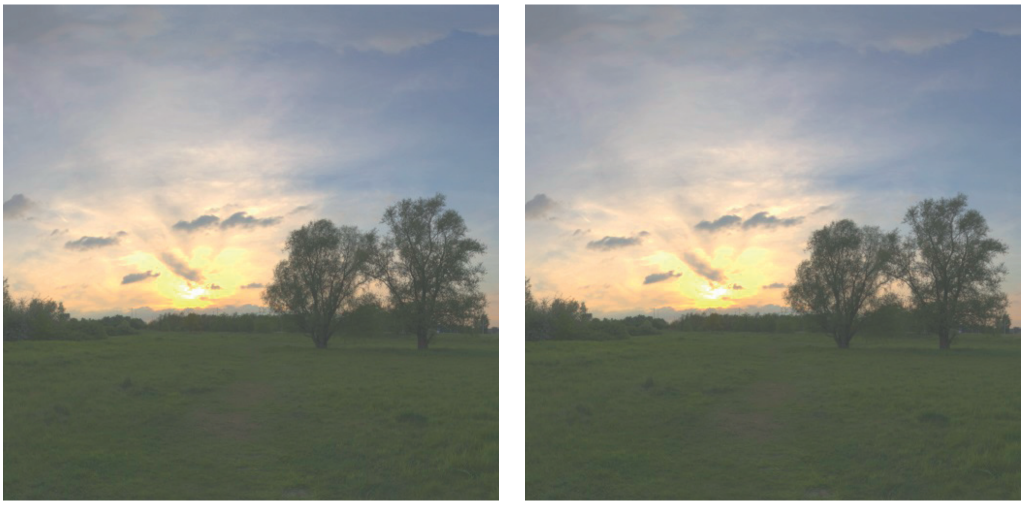
Figure 5.8 – Tone-mapping algorithm.

5.2.6 Results

Fig. 5.9a shows the HDR image before tone mapping. The landscape is not visible at all and the sun is too bright, hiding part of the clouds. Fig. 5.9b shows the image after tone-mapping and contrast-brightness corrections computed with the exact FPU. Thanks to this algorithm, the entire image is now visible and discernible. Fig. 5.9c-e show the tone-mapped image computed by the pruned, speculative and mixed FPU, respectively. Although the image quality might be limited by the printing quality, there is absolutely no difference discernible by the human eye between the pictures processed by the exact FPU and the ones processed by the approximate FPUs. There are however some differences that can be evaluated by computing the Peak Signal to Noise Ratio (PSNR) of each of the images as defined by eq. 5.5. It turns out that the pruned FPU produces the image with the lowest PSNR (76.4dB) while the speculative FPU produces the image with the highest PSNR (127.3 dB) when compared to the image processed with the exact FPU. Those results are consistent since the pruned FPU consumes less and produces larger errors than the speculative one. Finally, the mixed FPU achieves the best performances: it as the lowest power consumption and it produces and image with a PSNR of 90 dB which is in between the two other images generated with approximate FPUs. To give an order of magnitude, the PSNR of images

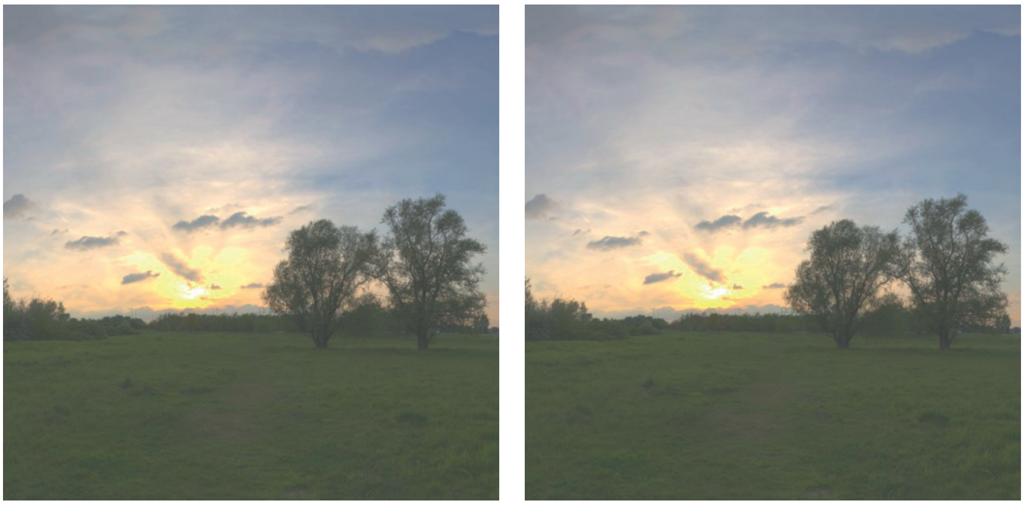


(a) Original image



(b) Exact

(c) Pruned (76.4 dB)



(d) Speculative (127.3 dB)

(e) Mixed (90.0 dB)

Figure 5.9 – Original image (a) and pictures tone-mapped by each of the 4 cores (b-e). PSNR is indicated for images processed by the approximate FPUs. Images size is $512 \cdot 512$ pixels.

after compression such as JPEG encoding are usually in the 50 dB range, explaining why there is no visible difference between all the produced images.

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) indicates the error magnitude of each individual pixel, color by color. It is defined as:

$$PVD = P_{\text{approx}} - P_{\text{exact}} \tag{5.7}$$

where P_{approx} is the pixel value of an image tone mapped by an approximate FPU and P_{exact} is the value of the same pixel in the image processed by the exact FPU. Since pixel data are stored as single-precision FP numbers, PVD is also a FP number indicating the deviation from the exact pixel value.

Fig. 5.10 plots the PVD distribution for each of the approximate tone mapped images. PVD equal to zero are removed from this distribution since they correspond to exact pixel values and therefore do not represent an error. The speculative FPU produces very small error of specific magnitudes due to the carry chain cut at specific bit positions. On the other hand, the error produced by the pruned FPU are spread by two orders of magnitude more than for the speculative FPU, but large errors remain rare. The error distribution of the tone mapped picture processed by the mixed FPU is clearly a combination of the two previous: it is spread like for the pruned FPU and it has a high error count around zero like the speculative FPU.

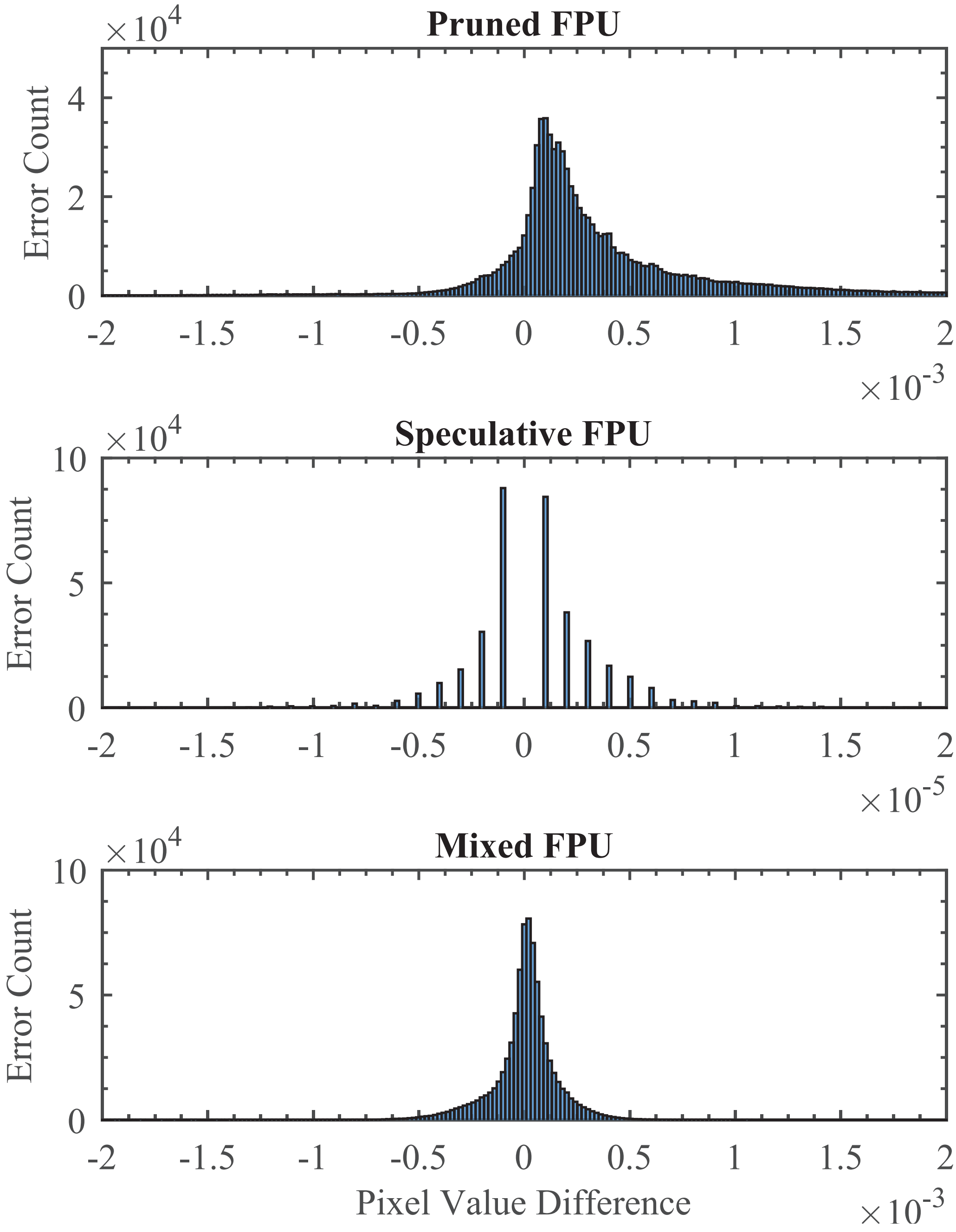


Figure 5.10 – Error distributions of the 3 images tone mapped with the approximate FPUs. X-axis of the speculative and Y-axis of the pruned FPU are scaled differently.

5.2.7 Remarks

By combining Gate-Level Pruning and Inexact Speculative Adder together with a novel Inexact Speculative Multiplier architecture, three single-precision Floating-point Units have been implemented by replacing the mantissa adder and multiplier by approximate versions. The FPUs have been integrated in a 65 nm CMOS process together with a quad-core PULP processor in order to demonstrate their functionality in a computing system. Measurement results show that with a random addition, multiplication and subtraction benchmark, the pruned FPU achieves 15 % power and 11 % area savings, whereas the speculative FPU enables 12 % power and 14 % area savings. Pruning and speculation clearly produce different types of errors and can be combined to achieve 27 % power, 36 % area and 53 % Power-Area Product reductions. The functionality of those approximate FPUs has been tested by executing a floating-point intensive tone-mapping algorithm on High-dynamic Range images. The results show no visible quality loss, however additional image error measurements show that each technique produces a specific error distribution, but all the errors remain small and centered on zero.

Unfortunately, the power consumption of the full chip does not really benefit from the power savings achieved on each of the approximate FPU: those are masked by the overhead of the cores, registers and memories. Indeed, each FPU occupies less than 1 % of the total chip area. It is therefore worth questioning whether complex hardware approximation techniques such as ISA or GLP have a real advantage over truncation. Indeed, as it will be shown in the following section, using truncation on a floating point not only enables power area and delay savings on the floating-point unit itself, but can also be used to reduce the amount of registers and memory which are the main energy consumers in a modern processor.

5.3 Estimating the energy gains in an approximate general-purpose processor with weather modelling as a case study

In opposition to previous sections where the energy-accuracy tradeoff was studied on arithmetic circuits and small hardware accelerators, this section aims at studying how the power consumption of a full processor can be improved by means of injecting inexactness or approximations. As a proof of concept, it is shown how the Intermediate General Circulation Model (IGCM) [1, 6, 7, 12] for weather and climate modelling can benefit from truncated arithmetic circuits and memories in the context of a general-purpose processor.

Table 5.4 – List of operations with their associated relative energy costs in virtual Joules. Energy numbers are normalized to the cost of an integer operation (add, subtract or multiply)

Operation category	Relative Energy (vJ)
Cache access	15
Main memory access	30
Floating-point operation	2
Branches and others	1
Integer operation	**1**

5.3.1 Modelling processor energy consumption

The goal of this work is to build a power model to show how inexact or approximate arithmetic units and memories can improve the overall energy consumption of a processor. To do so, energy across all major activities in both computation and memory systems need to be rigorously modelled. Towards this goal, against each type of activity that is being addressed, an energy cost relative to the least expensive activity is assigned, which, in our case is integer operations pegged at 1 virtual Joule (vJ). Every other activity, therefore, is assigned a relative cost in virtual Joules as shown Table 5.4. Here, rather than showing actual energy numbers which are specific to one processor architecture, virtual Joules are chosen as a generic unit and the energy model presented here is not linked to one specific processor, but is used to get a rough estimate of the relative savings that can be achieve on modern processor architectures that would embed approximate hardware. For this experiment, computational operations are broadly classified into floating-point operations, integer operations, and others that would include branches/control flow, etc. These relative costs have been derived as best estimates obtained by insights from source [9]. It can clearly be seen that main memory accesses are the most energy hungry instructions. It can also be noted that Caches were designed to significantly reduce access times (sometimes up to a factor of 100), but their energy costs are closer to that of the main memory – a factor of two here. The implication, of course, is that any effort to exploit approximation techniques that can be applied to both types of memory accesses will pay rich dividends.

5.3.2 Modelling the energy gains

To model the energy consumed by a program, the number of arithmetic operations and memory accesses are measured, or counted while the program is being executed. This instruction count is denoted by c_i for each category i listed in table 5.4. Let e_i denote the relative energy consumed per operation/access of category i when executed in an exact manner (this is column 2 in Table 5.4. Then, the total relative energy consumed

E_{tot} by the program for exact computations is given by,

$$E_{tot} = \sum_i c_i \cdot e_i \tag{5.8}$$

The gains achieved by approximating some operations can then be measured by counting the fraction f_i of operations / memory accessed that are executed in an approximate manner. Typically, activities that involve data can be made inexact, but all control statements, branches, iterators, etc. must remain exact. Indeed, the energy model presented here supports some operations to be inexact – including inexact memory banks, FPUs and others, while the rest are exact. Let e'_i denote the reduced energy consumed by the approximate of an operation / memory access of the category i. The total energy consumed by the program with part of the instructions being approximate operation E'_{tot} is given by,

$$E'_{tot} = \sum_i c_i \cdot (f_i \cdot e'_i + (1 - f_i) \cdot e_i) \tag{5.9}$$

The relative energy gains between the program executed with approximate instruction versus the program executed with exact instructions only is simply equal to E_{tot}/E'_{tot}. While there are many tools and literature for measuring e_i values, special care needs to be taken to measure e'_i values. Here, inexactness is introduced by truncating the least significant bits. Thus, the energy savings are functions of the number of truncated bits. In the specific case of floating-point operations, any inconsistency in the exponent would result in a huge error distance, thus, truncation is only applied to elements computing the mantissa. To estimate the relative energy savings resulting from truncation, both exact and truncated arithmetic units have been synthesized in the UMC 65 nm process, following the standard digital design flow. Since the estimated energy consumptions are strongly dependent on the technology, the energy cost of each inexact operation is normalized to the cost of an exact operation with 64 bits. Each truncated variant of a design is synthesized under the same delay and area constraints than the exact counterpart.

In the context of memory, the DRAMsim tool [13] has been used to determine energy estimates of the main memory. Again, these numbers are normalized to the cost of a standard 64-bit read or write operation. Energy savings are obtained by reducing the number of bits written or read, per operation. Since we allow the least significant bits to be present in memory, the energy for refreshing remains the same as for exact memory; this was modelled faithfully. Assuming that the cache is built out of SRAM cells, and following [11], the cache energy consumption scales linearly with the width of the word. It should be noticed that overheads involving possible dynamic alteration of width such as power gating are not included in these energy estimates, in this regard, the results presented here should be viewed as optimistic estimates if a dynamically adjustable architecture is to be considered. It should also be noted that the relative savings for memory might however be slightly optimistic since instruction codes memory accesses are not accounted for here.

5.3.3 Case study of IGCM for climate and weather modelling

An atmosphere model

This work studies the use of inexact hardware in the Intermediate General Circulation Model (IGCM) that represents atmospheric dynamics in global simulations [1, 6, 7, 12]. IGCM is used to solve the governing fluid dynamical equations for atmospheric motions in three dimensions on the sphere. In contrast to an operational weather or climate model, it does not include a representation of components such as physical tracers, biosphere, ocean, topography, water vapor or clouds. However, IGCM represents the basic building block of some of the most important weather and climate models (such as IFS and ECHAM) and provides a meaningful testbed for operational models. The model can run in climate configurations for long time intervals (from years to centuries). Here, short-term simulations equivalent to weather forecasts for several days are considered. Numerical simulations of the atmosphere are of crucial importance for reliable forecasts of weather and climate. The quality of these forecasts is dependent on the resolution used, and complexity of the numerical models, which is limited by the computational power of today's supercomputing facilities. By allowing some operations to be approximate, one could potentially free up some computational resources for other operations that could improve the final result of the climate model.

Analysis of the results

Earlier studies [5, 4] have reported that a reduction in precision does not cause a catastrophic reduction of the quality of model simulations for many applications in atmospheric modelling. These studies focused on relating error to the number of bits in the mantissa but did not consider the relationship to energy, which is now pursued here. As long as certain parts of the model are left with high precision (more specifically the dynamics of large-scale pattern and the time-stepping scheme), numerical precision can be reduced heavily with no strong increase in model error. The robustness of atmosphere models in the presence of inexactness can be explained by the inherent uncertainty that is present in model simulations mainly due to physical processes that cannot be fully captured, and the high viscosity which is needed for turbulent closure.

To evaluate the impact of approximations on the energy consumption and on the

Table 5.5 – Normalized energy consumption, global mean error for geopotential height in [m] at day 2 (averaged over 5 forecasts) and normalized operations per virtual Joule (OPVJ) for the different simulations. were presented in [10] and [11].

Model run	Normalized energy	Forecast error at day 2	Normalized operations per virtual Joule
235 km, 64 bits	1	2.3	1.00
260 km, 64 bits	0.82	2.8	1.01
315 km, 64 bits	0.47	4.5	1.2
235 km, 24 bits	0.35	2.3	2.83
235 km, 22 bits	0.32	2.3	3.09
235 km, 20 bits	0.29	2.5	3.41

quality of the result, IGCM model runs are performed in a standard testbed for three dimensional simulations of the atmosphere, the so-called Held-Suarez configuration, with 20 vertical levels. If resolution is increased, grid spacing is decreased and quality of model simulation will be consequently improved. The model error is calculated by comparing runs at coarser resolution to a simulation with much higher resolution with a grid spacing of 125 km. Typically, the different simulations will diverge from the high-resolution simulation with time, due to model errors. A forecast is performed with 235 km resolution in an exact setting using full double precision (64-bit) that serves as a reference or baseline for the model's quality. The computational cost is then reduced by using either lower resolution (260 km and 315 km) together with double precision, or by using the same resolution as the reference run (235 km) but with truncated hardware.

Table 5.5 provides values for the normalized energy demand, the forecast error in geopotential height at day 2 of the forecast and the normalized number of operations per virtual Joule (OPVJ). Geopotential height is a standard meteorological diagnostic that relates pressure to height. The simulations with inexact hardware clearly show a reduced normalized energy demand and an increased number of operations per virtual joule. It is found that all simulations with 235 km resolution clearly show a smaller forecast error in comparison to simulations in double precision at lower resolution (260 km or 315 km), even if floating-point precision is truncated at 20 bits in the floating-point representation (8 bits in the significand). As reiterated in the table, the forecast error with inexact hardware is hardly affected in a significant way in comparison to the double precision simulation. This experiment shows that by using truncated arithmetic, up to 70 % energy can be saved on the IGCM run without significantly affecting the quality of the forecast. Those huge gains are made possible not only by the truncated arithmetic, but mainly by the truncated memories, which consume 15 X to 30 X more than the arithmetic.

5.4 Concluding remarks

This chapter presented how the energy-accuracy tradeoff can be achieved at the level of an entire application. It has been shown that by applying gate-level pruning on a hardware discrete cosine transform – which is the most computationally intensive block used for image and video encoding – up to 12 % area and power can be saved at the cost of a graceful image degradation. Similarly, by applying GLP and ISA on a 32-bit floating-point unit, silicon measurement show up to 27 % power and 36 % area savings. An image tone-mapping application has been used to verify the functionality of those approximate floating-point units leading to no visible quality loss. However, when the full chip is taken into consideration, i.e. the FPUs with the surrounding cores and memories, there are no measurable power savings arising from the approximate FPUs. Indeed, each of the FPU occupies less than 1 % of the total chip area. Most of the area and power is consumed by registers and memories, which leads to question if complex techniques such as GLP or ISA really have an advantage over truncation at the level of a full system or a complete application. The main advantage of ISA and GLP over truncation is that they preserve most of the bits, which means that the least significant bits are generally still computed. This preserves the dynamic range and is great for error behaviour, but it could also be the main drawback when those approximate circuits are placed in systems built with multiple registers and memories since these approximation techniques do not allow to reduce the number of bits that need to be stored. Unquestionably, GLP and ISA enable large savings on the arithmetic circuits, but they lead to very few or almost no savings to the surrounding registers and memories as most of the bits are computed and need to be stored. In opposition, truncation enables large savings on arithmetic circuits and on the surrounding registers and memories as well. Since the least significant bits are truncated, no more memories and registers are required to store them. This explains why the huge savings enabled by ISA and GLP on single adders and multipliers turn out to be insignificant or even imperceptible on a full chip. To overcome this issue, GLP and ISA should be used together with approximate registers and memories on bit positions that are inexact.

To understand how much energy could be saved on a general-purpose processor using truncation or approximate arithmetic circuits together with approximate register and memories, a simplistic energy model has been developed. It has been shown that in the case of the IGCM for weather modelling, up to 70 % energy could be saved over the entire processor by switching from 64-bit precision to 20-bit precision. Unfortunately, this bitwidth is not standard in a processor datapath but it means that bitwidth has a huge impact on the energy consumption of general purpose processors. To further investigate how the energy is distributed in a real processor architecture, the next chapter presents a detailed energy model and shows how energy-accuracy tradeoff can be achieved on commercially available circuits by means of programming techniques.

It should be noted that energy consumption could also be saved without truncated hardware but by emulating truncation using software techniques: by zeroing the LSBs of two operands, one could save on the dynamic energy consumption of arithmetic circuits and registers. Of course, the energy gains resulting from this method would be lower than the ones achievable with real truncated hardware since memory might overwhelm the savings. However, the main advantage is that it would use existing standard hardware and the precision could be tuned during run-time by selecting the number of LSBs to be zeroed, which is not the case for the techniques previously mentioned in this work.

References

[1] M Blackburn. Program description for the multi-level global spectral model. *University of Reading, Dept. of Meteorology, Atmospheric Modelling Group*, 1985.

[2] Francesco Conti, Davide Rossi, Antonio Pullini, Igor Loi, and Luca Benini. Energy-efficient vision on the pulp platform for ultra-low power parallel computing. In *Signal Processing Systems (SiPS), 2014 IEEE Workshop on*, pages 1–6. IEEE, 2014.

[3] Vítor A Coutinho, Renato J Cintra, Fábio M Bayer, Sunera Kulasekera, and Arjuna Madanayake. A multiplierless pruned dct-like transformation for image and video compression that requires ten additions only. *Journal of Real-Time Image Processing*, pages 1–9, 2015.

[4] Peter D Düben and TN Palmer. Benchmark tests for numerical weather forecasts on inexact hardware. *Monthly Weather Review*, 142(10):3809–3829, 2014.

[5] Peter D Düben, Hugh McNamara, and Tim N Palmer. The use of imprecise processing to improve accuracy in weather & climate prediction. *Journal of Computational Physics*, 271:2–18, 2014.

[6] BJ Hoskins and AJ Simmons. A multi-layer spectral model and the semi-implicit method. *Quarterly Journal of the Royal Meteorological Society*, 101(429):637–655, 1975.

[7] IN James and JP Dodd. A simplified global circulation model. *User's manual. Department of Meteorology, University of Reading*, 1993.

[8] Georgios Karakonstantis, Nilanjan Banerjee, and Kaushik Roy. Process-variation resilient and voltage-scalable dct architecture for robust low-power computing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 18(10):1461–1470, 2010.

[9] Sheng Li, Jung Ho Ahn, Richard D Strong, Jay B Brockman, Dean M Tullsen, and Norman P Jouppi. Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Microarchitecture, 2009.*

MICRO-42. 42nd Annual IEEE/ACM International Symposium on, pages 469–480. IEEE, 2009.

[10] Nathan Moroney. Local color correction using non-linear masking. In *Color Imaging Conference (CIC), 8th IS&T/SID*, pages 108–111, Nov 2000.

[11] Takanori Okuma, Yun Cao, Masanori Muroyama, and Hiroto Yasuura. Reducing access energy of on-chip data memory considering active data bitwidth. In *Proceedings of the 2002 international symposium on Low power electronics and design*, pages 88–91. ACM, 2002.

[12] AJ Simmons and BJ Hoskins. A comparison of spectral and finite-difference simulations of a growing baroclinic wave. *Quarterly Journal of the Royal Meteorological Society*, 101(429):551–565, 1975.

[13] David Wang, Brinda Ganesh, Nuengwong Tuaycharoen, Kathleen Baynes, Aamer Jaleel, and Bruce Jacob. Dramsim: a memory system simulator. *ACM SIGARCH Computer Architecture News*, 33(4):100–107, 2005.

[14] Sungwook Yu and EE Swartziander. Dct implementation with distributed arithmetic. *IEEE transactions on Computers*, 50(9):985–991, 2001.

6 Programmable systems

The previous chapters were discussing the use of application specific approximate hardware to achieve significant energy savings by slightly reducing the precision of the computation. The same tradeoff can be achieved on commercially available general purpose processors using programming techniques. For instance, Intel recently introduced some support for half-precision data format into the Advanced Vector Extensions 2 (AVX2) instruction set. Even though half-precision arithmetic operations are not natively supported, meaning that values need to be converted back and forth to single-precision to do a calculation, up to 4 X savings are possible when switching from double-precision to half-precision format [5].

In this regard, this chapter aims at studying the energy per instructions of modern Intel processors in order to understand how data precision affects the overall energy consumption. This chapter is organized as follow: section 6.1 presents the EPIs as well as the measurement methodology. Section 6.2 introduces the Newton application for solving nonlinear equations which is used along this chapter. Section 6.3 shows how EPIs can be used to model the energy consumption of the Newton application. This model is further used to study the energy breakdown and identify the most consuming operations. Finally, Section 6.4 shows that additional energy savings are possible by implementing specialized hardware on programmable logic interconnected with a processor.

6.1 Energy measurements and EPI of commercial processors

In order to develop a power model capable of estimating the energy consumed by a program, the energy cost of each single instruction should be studied. One way to measure the EPI on Intel's processor is to use the Running Average Power Limit (RAPL) performance counter. The RAPL counter is normally used to limit the power consumption of all the cores to the Thermal Design Power (TDP) limit of the package

Algorithm 1 Generic RAPL energy measurement

```
 1: Allocate and fill memory with random values
 2: Initialize RAPL
 3: for $i = 1,\ i < N,\ i{+}{+}$ do
 4:     Increment memory pointer
 5:     Load data to register (optional)
 6:     Inline assembly instruction
 7:     Write result to memory (optional)
 8: end for
 9: Read RAPL
10: Re-run lines 1 to 9 with line 6 commented out
11: The difference between the two energy readings
12: corresponds to N executions of the instruction line 6
```

and it can easily be used to monitor the energy consumption of each of the cores as well as external memory. For this work, the measurements have been performed on a Dell precision T1700 workstation equipped with an Intel core i7 4770 (3.40 GHz), 16 GB of DDR3 RAM and operated by CentOS 7, Linux kernel 4.3. All CPU cores have been set to the minimal P-State except for one. The P-state refers to the frequency and voltage operating point of the core. The distinguished core was set to Turbo Boost, which is the highest P-State: the specified core runs at its maximum frequency and voltage.

Measuring energy consumption with the RAPL is done as follow:

1. Initialize the RAPL counter.

2. Run the code to be measured.

3. Read energy values from the RAPL counter.

This works perfectly for measuring the consumption of a full program. However, the goal of EPI measurement is to monitor one single inline assembly instruction. Since the consumption of a single instruction is generally below the resolution of the RAPL counter, the instruction needs to be looped and the energy consumed by the loop and all the other overhead need to be cancelled out. Algorithm 1 shows the pseudo-code used for measuring EPIs. Lines 1 to 9 of the algorithm are executed two times: once with the instruction to be measured and once without. This allows to cancel out the energy consumed by the loops and memory operations surrounding the instruction of interest. The energy consumed by the instruction is then equal to the difference between the two energy readings divided by the number of executions N. Since there are slight variations between successive energy measurements of the same micro-benchmark, the values presented in Table 6.1 and 6.2 are averaged over 10 measurements.

Table 6.1 – Measured arithmetic EPI

Instruction	Assembly	EPI (nJ)
Scalar add (integers)	ADD	1.46
Scalar add (singles)	ADDSS	1.31
Scalar add (doubles)	ADDSD	1.39
128-bit Vector add (singles)	VADDPS	1.19
256-bit Vector add (singles)	VADDPS	1.11
No OPeration	NOP	0.49
256-bit Vector FMA (singles)	VFMADD231PS	16
256-bit Vector FMA (doubles)	VFMADD231PD	4
256-bit Vector subtract (singles)	VSUBPS	1.94
256-bit Vector subtract (doubles)	VSUBPD	2.02

Table 6.1 shows the EPI for selected arithmetic operations as well as for the No OPeration (NOP). In these experiments, random numbers have been loaded to the registers before executing the arithmetic instruction in order to avoid looped operations on zeros that would lead to lower switching activity and thus lower energy consumption. Most of the arithmetic operations cost about one to two nJ and vector instructions are slightly more expensive than scalar instructions. This is particularly true for vector Fused-Multiply and Add (FMA), which is obviously more complex than a single addition or a single multiplication operation. Surprisingly, a 256-bit vector FMA seems to be four times more expensive when executed on single-precision numbers rather than double-precision numbers.

Table 6.2 shows the EPI for selected data movement operations. Here, a memory load refers to a variable loaded from the memory to one of the processor's register. Similarly, a store operation refers to a variable which is moved from one of the registers to the memory. Each EPI has been measured using two different methods. The first set of measurement was done while ensuring that the data is loaded from the main memory by incrementing the memory pointer between each successive instruction by more than a cache line size. The second measurement has been done by loading data sequentially, letting the hardware prefetcher do his job, therefore drastically reducing the energy consumed. There is indeed a huge consumption difference between *sequential* loads and *non-sequential* loads. Moreover, storing operations seem to be twice as expensive as loads, except for scalars with a sequential access pattern. The cost of sequential memory operations are also directly proportional to the number of bits they treat. For instance, a 256-bit sequential load is twice as expensive as a 128-bit sequential load and 8 times more expensive than a 32-bit integer load. As shown in Table 6.2, many data movement operations share the same assembly instruction. For instance,

Table 6.2 – Measured data movement EPI

Instruction	**Assembly**	**EPI (nJ)**
Main memory integer load (scalar)	MOV	194
Main memory integer store (scalar)	MOV	370
Sequential integer load (scalar)	MOV	5.31
Sequential integer store (scalar)	MOV	19.8
Main memory single load (scalar)	MOV	198
Main memory single store (scalar)	MOVSS	365
Sequential single load (scalar)	MOVSS	17
Sequential single store (scalar)	MOVSS	12.8
Main memory double load (scalar)	MOVSD	199
Main memory double store (scalar)	MOVSD	363
Sequential double load (scalar)	MOVSD	22
Sequential double store (scalar)	MOVSD	18.6
Main memory 128-bit vector load	VMOVDQA	206
Main memory 128-bit vector store	VMOVDQA	395
Sequential 128-bit vector load	VMOVDQA	19
Sequential 128-bit vector store	VMOVDQA	41
Main memory 256-bit vector load	VMOVDQA	205
Main memory 256-bit vector store	VMOVDQA	389
Sequential 256-bit vector load	VMOVDQA	37
Sequential 256-bit vector store	VMOVDQA	93
Integer prefetch from main memory	prefetcht0	163
Sequential integer prefetch	prefetcht0	5.22

VMOVDQA is used for moving 128-bit vectors as well as 256-bit vectors, the difference lies in the registers used with this instruction. 128-bit vectors are stored / loaded to xmm registers while 256-bit vector use the ymm registers.

These two tables clearly show that data movement is the main energy consumer. One single memory transfer not only consumes 5 X to 180 X more energy than a single arithmetic operation, but each arithmetic operations also requires at least two memory transfers, i.e. the two input operands, which certainly makes data movement the main energy consumer with this architecture.

6.2 Energy-precision tradeoff of the Newton method

In [7], Leyffer et. al evaluated the energy-performance tradeoff of the Newton method for solving nonlinear equations with single and double-precision arithmetic. In this work, the same application will be used to evaluate the precision of the EPI based energy model. The Newton application used here is particularly interesting as it shows several similarities with weather modelling applications, but at a smaller scale. The energy model developed here could therefore be used on the weather modelling application to evaluate the possible energy savings. This section briefly describes the Newton method used for this work, but the detailed algorithm and implementation can be found in [7].

The aim of the Newton method is to solve the nonlinear system of equations

$$F(x) = 0, \tag{6.1}$$

where $F : \mathbb{R}^n \to \mathbb{R}^n$ is a twice continuously differentiable function.

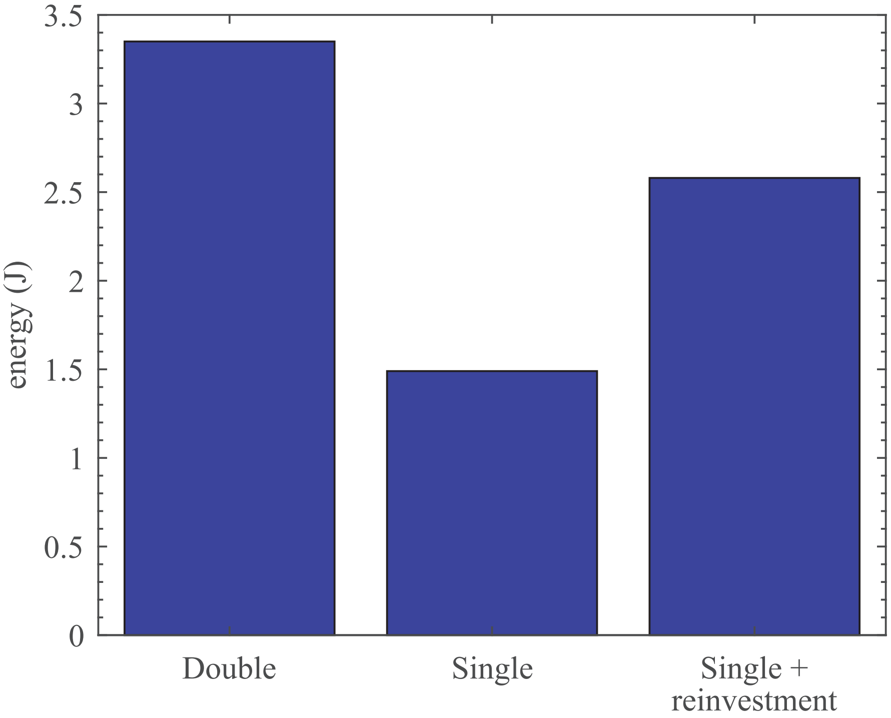
The basic inexact Newton method starts from an initial iterate x^0, and consists of outer and inner iterations. The outer iterations correspond to approximate Newton steps that produce a sequence $\{x^k\}_k$ of iterates. Given x^k, the inner iteration solves the Newton system

$$\nabla F(x^k)s = -F(x^k) \tag{6.2}$$

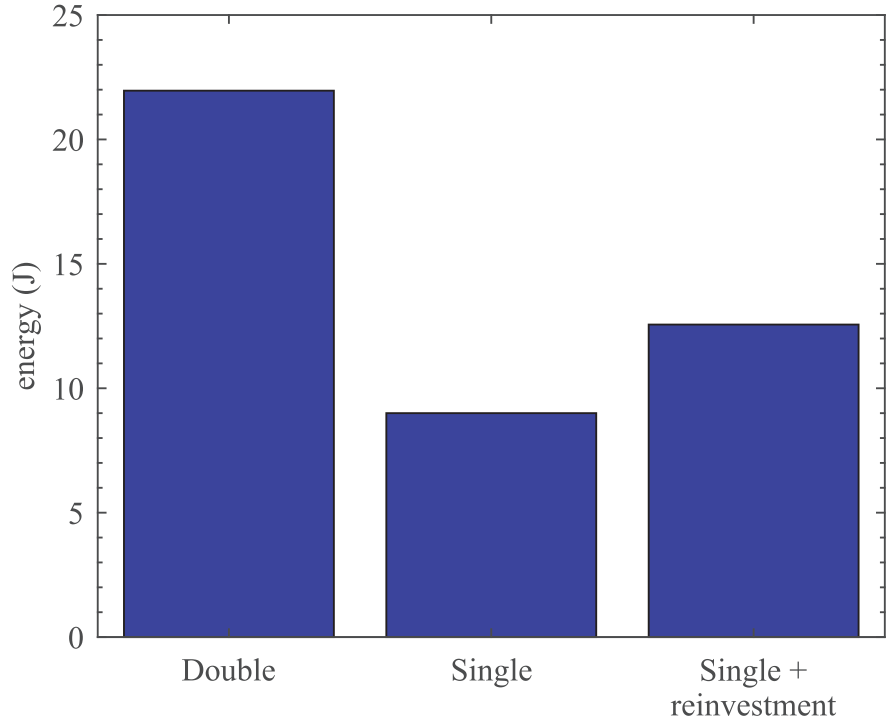
approximately for $s \in \mathbb{R}^n$. The inner iterations terminate on the accuracy of the putative direction s; that is, one stops when s satisfies a relative residual criterion for (6.2),

$$\frac{\|\nabla F(x^k)s + F(x^k)\|}{\|F(x^k)\|} \leq \eta_k, \tag{6.3}$$

where $0 \leq \eta_k < 1$ is a sequence of tolerances that is forced to zero as k increases.



(a) Laplace; convergence accuracy 10^{-6}; post-reinvestment accuracy 10^{-10}



(b) Rosenbrock; convergence accuracy 10^{-5}; post-reinvestment accuracy 10^{-13}

Figure 6.1 – Measured Energy consumption for varying precision with a problem size of 10000

Two sets of test problems of variable dimension are considered. The first problem, Laplace, is a well-conditioned linear system of equations, derived from a central-difference discretization of the Poisson equation. The second problem, Rosenbrock, is nonlinear and notoriously ill conditioned, and was chosen to provide a more strenuous test for the low-precision implementation.

Laplace

The first system of equations is given by,

$$\begin{aligned} F_1(x) &= b_1 + 4x_1 - x_2 && (6.4)\\ F_i(x) &= b_i - x_{i-1} + 4x_i - x_{i+1}, \qquad i = 2, \dots, n-1 && (6.5)\\ F_n(x) &= b_n - x_{n-1} + 4x_n, && (6.6) \end{aligned}$$

where $b_1 = 1.0$, $b_i = -2.0$ for $i = 2, \dots, n-1$, and $b_n = 4.0$.

Chained Rosenbrock

The second, nonlinear system of equations is derived from the chained Rosenbrock function

$$\sum_{i=1}^{n-1} \left(a(1-x_i)^2 + 100 \left(x_{i+1} - x_i^2 \right)^2 \right), \tag{6.7}$$

whose first-order system of equations are given by,

$$\begin{aligned} F_1(x) &= 2a(x_1 - 1) - 400x_i \left(x_2 - x_1^2 \right) && (6.8)\\ F_i(x) &= 200 \left(x_i - x_{i-1}^2 \right) + 2a(x_i - 1) - 400x_i \left(x_{i+1} - x_i^2 \right), \quad i = 2, ..., n-1 && (6.9)\\ F_n(x) &= 200 \left(x_n - x_{n-1}^2 \right). && (6.10) \end{aligned}$$

The parameter $a > 0$ controls the conditioning of the problem. Here, the standard value $a = 1$ is used.

It has been shown in [7] that up to half of the energy consumption can be saved by switching from double precision to single precision with the same convergence accuracy. In order to improve the final convergence accuracy, this saved energy can then be reinvested by switching back to double precision once the single-precision version reaches its maximum accuracy. With this approach, measurements have shown up to 4 orders of magnitude convergence accuracy improvement for Laplace and up to

Table 6.3 – EPIs for simplified model

Instruction	EPI (nJ)
Vector add	1.11
Vector fused multiplied and add	4
Vector subtract	2
L1 hit	3
L2 or L3 hit	5
Memory load / store	80

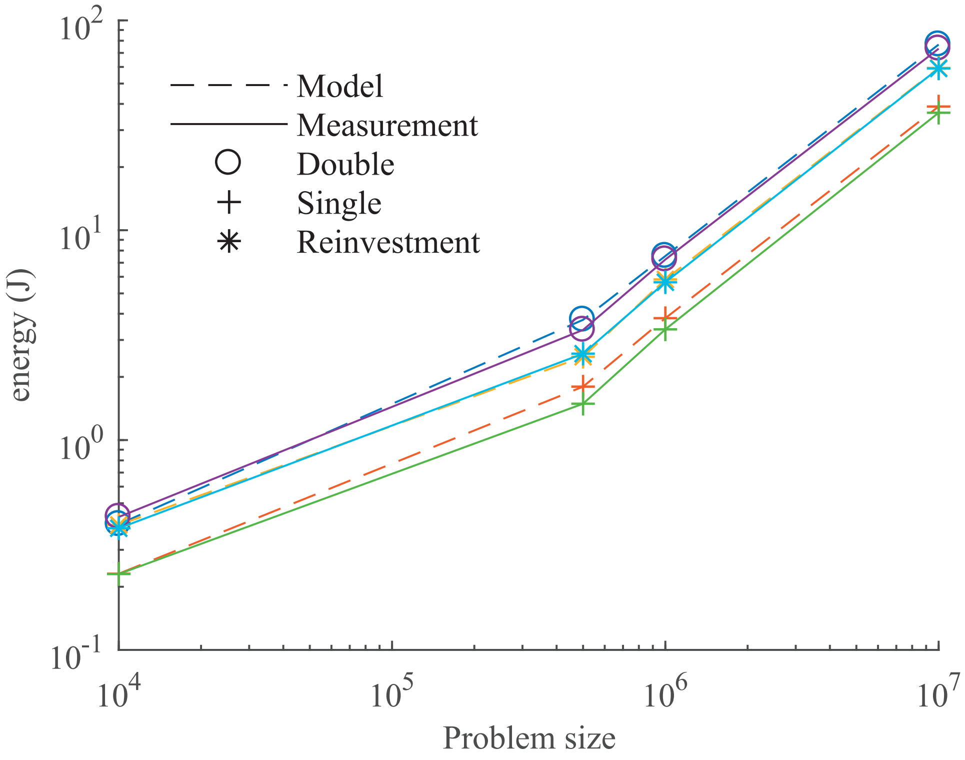
8 orders of magnitude convergence accuracy improvement for Rosenbrock.

6.2.1 Measurements with performance counters

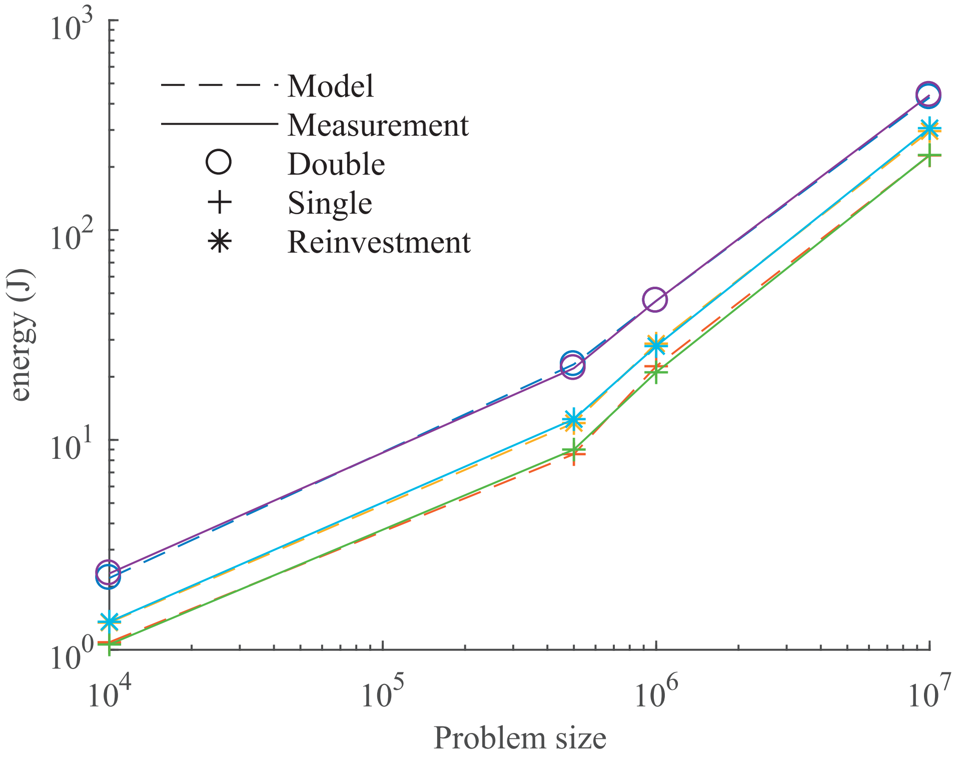
In this work, the same experiments have been performed using the same Newton method. The Fortran code has been compiled with Intel's Ifort compiler with vectorization option enabled to achieve a maximal efficiency. Energy measurements have been performed using the RAPL counter. As shown in Fig. 6.1, similar energy numbers and accuracy improvement factors have been obtained. Fig. 6.1a shows the energy measurements for the Laplace applications. The energy consumption is reduced by a factor 2 by switching from double to single precision with a convergence accuracy of 10^{-6}. After reinvestment, a final convergence accuracy of 10^{-10} is obtained by switching back to double precision and continuing the calculations until the initial energy budget is reached. Here, single precision with reinvestment consumes slightly less than the original double-precision energy budget. Indeed improving the final convergence accuracy by another order of magnitude with the reinvestment method would exceed the original energy budget. Similarly for the Rosenbrock application shown Fig. 6.1a, the same energy reduction is achieved by switching from double to single precision with a convergence accuracy of 10^{-5}, and after reinvestment, a convergence accuracy of 10^{-13} is reached consuming only 60 % of the initial energy budget.

6.3 EPI-based energy model

It has been demonstrated that significant energy savings can be achieved on commercially available processors by simply switching from double-precision to single-precision data type. The aim of this work is to create a simple EPI based energy model to get an energy breakdown and understand which instructions consume the most, and how reduced precision can decrease the total energy consumption.



(a) Laplace



(b) Rosenbrock

Figure 6.2 – Modelled and measured energy consumption versus problem size

The EPIs have been measured in order to build an energy model and understand which type of instruction dominates the energy consumption. An exhaustive model would require to get the EPI data for all possible instructions and then count the occurrence of all those instructions for the program under analysis. The energy consumed would therefore be equal to the sum of EPIs multiplied by their instruction count as shown below

$$E = \sum EPI_i \cdot IC_i. \tag{6.11}$$

Such an exhaustive model would be difficult to build since it would require a microbenchmark to measure each single instruction, but it also requires to count the occurrence of each instruction for a given program. For the sake of simplicity, the model developed in this work is only based on the instructions shown in Table 6.3. Cache accesses costs have not been measured but have been derived with the best insights from [6, 9]. Cache hits and misses have been counted using the cachegrind profiler [1]. Since cachegrind does not distinctly count L3 hits and misses, L2 and L3 cache accesses costs had to be averaged as shown in Table 6.3. Similarly, to avoid complexifying the energy model, hardware prefetches are not counted and all the main memory load and store operations are averaged to 80 nJ. Moreover, it is assumed that instruction cache and data cache operations cost the same amount of energy.

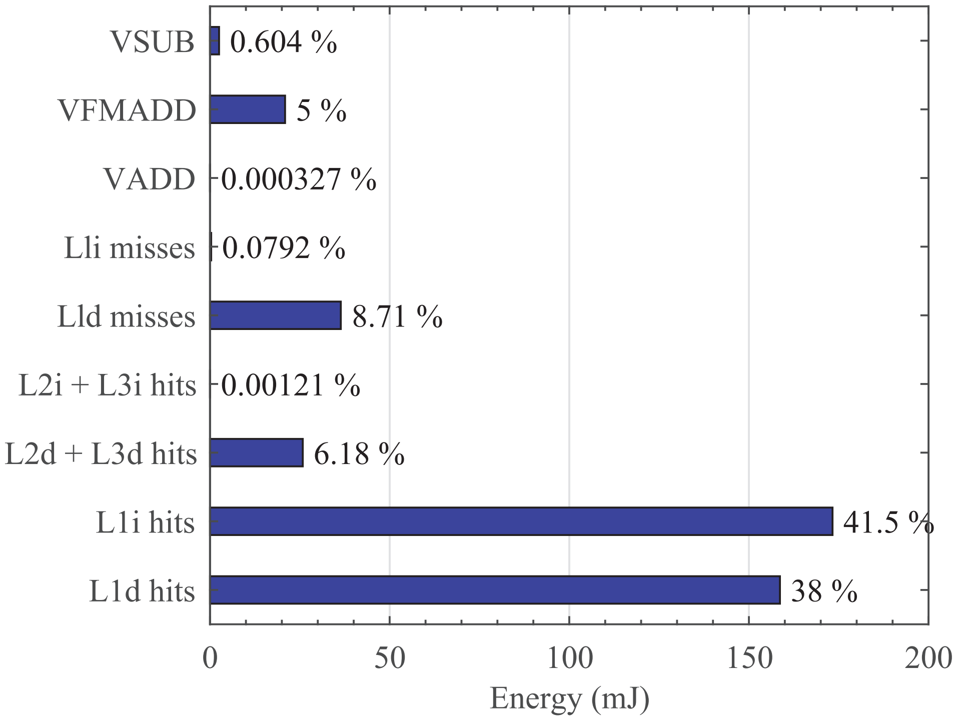
Since the Laplace and Rosenbrock application are floating-point intensive and heavily vectorized, vector additions, vector multiplications and vector subtractions have been counted using Intel's Software Development Emulator (SDE) [2].

6.3.1 Accuracy of the energy model

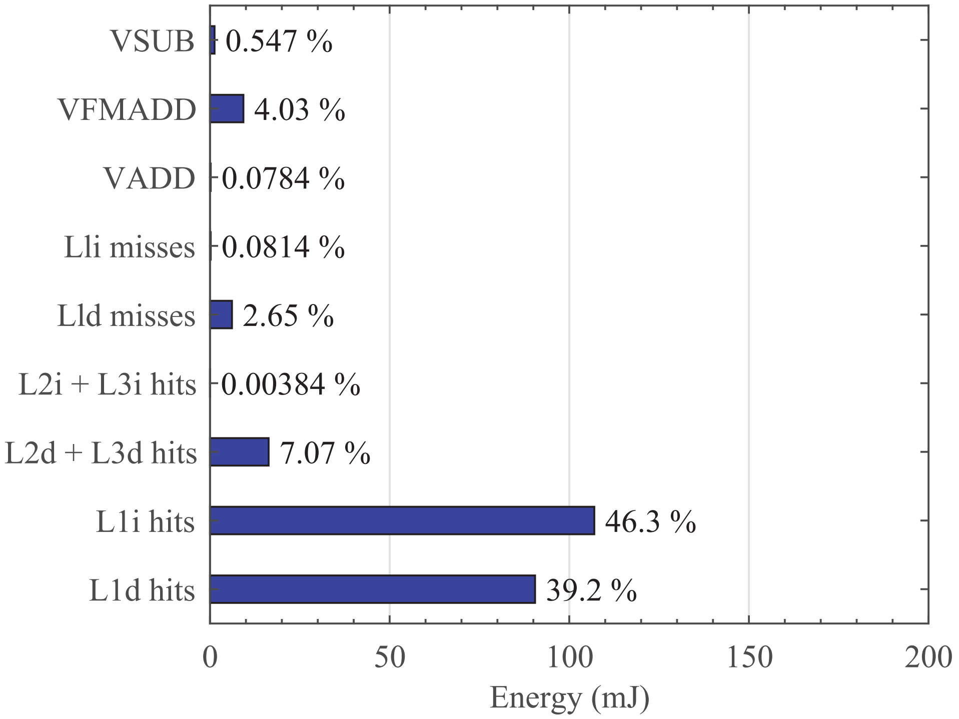
Fig. 6.2a and 6.2b show the energy consumption of the Laplace and the Rosenbrock application with a variable problem size. Dashed lines represent the energy consumption estimated by the EPI based model whereas full lines represent the measurement obtained with the RAPL counter. For all problem sizes and precision modes, i.e. double, single and single precision with reinvestment, the model fits the RAPL measurements with less than 10 % error.

6.3.2 Energy breakdown

As a precise energy model has been built, it can now be exploited to study the energy breakdown and understand which instruction are the most expensive over the entire program execution. Fig. 6.3 shows this energy breakdown for the Laplace problem with a problem size of 10000 for double precision (Fig. 6.3a), single precision (Fig. 6.3b) and single precision with reinvestment (Fig. 6.3c). The arithmetic instructions reported here are vector subtractions (VSUB), additions (VADD) and fused multiply and add (VF-

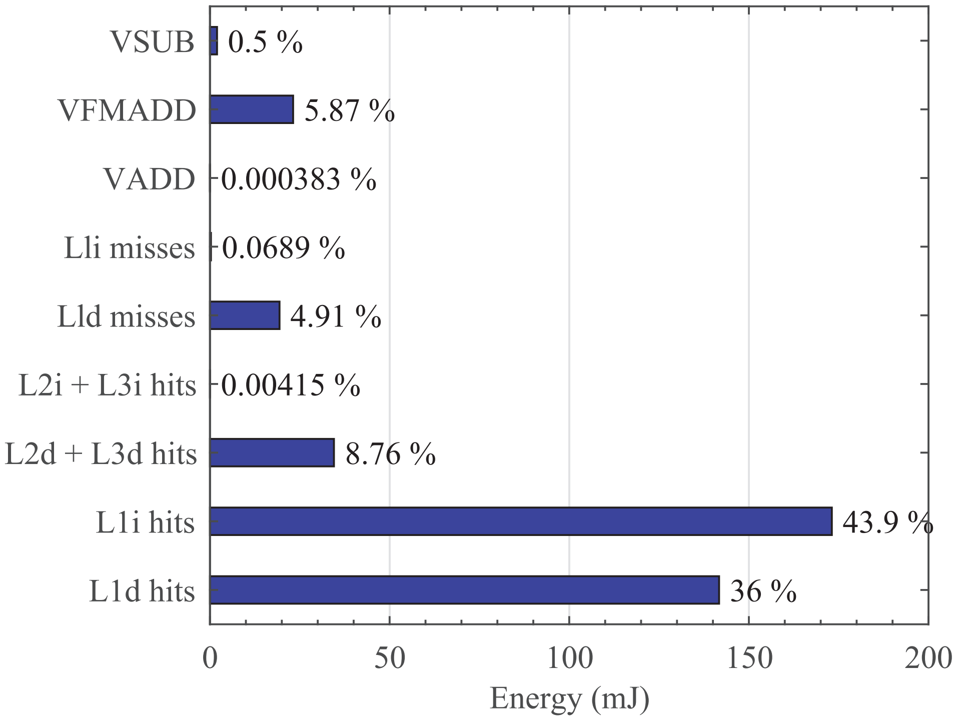


(a) Double precision



(b) Single precision

Figure 6.3 – Energy breakdown of the laplace problem with a problem size of 10000



(c) Reinvestment (single and double precision)

Figure 6.3 – Energy breakdown of the laplace problem with a problem size of 10000

MADD). Main memory accesses correspond to the sum of Last level instruction misses and Last level data misses (Lli and Lld misses). Finally, cache hits are represented by L1, L2 and L3 instruction and data hits.

For all the different precision modes, L1 instruction hit is the most consuming operation with around 44 % of the total energy consumed. Similarly, L1 data hit also takes 40 % of the total energy consumption independently of the precision. This means that around 80 % of the energy is consumed by L1 cache hits only. The remaining energy is mainly consumed by L2-L3 data cache hits and vector floating-point operations. The latter only consume less than 6 % of the energy, which is rather surprising for a computationally intensive application.

These bar charts are also interesting to study the effect of precision reduction on the energy consumption. When switching from double precision Fig. 6.3b, to single precision Fig. 6.3a, the consumption of almost each instructions is reduced by a factor 1.6 X to 2 X. The explanation to this is rather simple; with a 256-bit vector processing unit such as the one used in this processor, 4 double-precision or 8 single-precision values can be processed in one single instruction. Hence, reducing precision allows to reduce the number of instructions as well as the data movements by about a factor 2. As a consequence, the energy consumed by instruction cache hits as well as data cache hits is naturally reduced. Moreover, precision reduction also improves cache behaviour by decreasing main memory accesses. Indeed, a lower bitwidth allows for more values to be stored in the cache. For this reason, the 40 mJ consumed by Lld misses in double precision goes down to 10 mJ in single precision.

After the reinvestment phase (Fig. 6.3c), the energy breakdown is almost similar to the double-precision breakdown except that there are less main memory accesses and more L2 and L3 hits. Indeed, the program starts by loading single-precision values to cache, and once that the maximum convergence precision for single precision is reached, those values are converted to double precision locally and do not require any main memory access.

It should be noted that for this type of architecture, approximate arithmetic units are not appropriate unless they exploit bitwidth reduction such as truncated adders and multipliers. Indeed, even with an approximate floating-point unit saving 30 % energy [4], the overall savings might not be perceptible considering that those would be effective on the 6 % energy consumed by arithmetic operations only. For this reason, truncation certainly remains the most simple and most efficient approximate hardware design technique as demonstrated in [3]. It does not only bring savings on the arithmetic operations, but also to the memory transfers.

6.4 Increasing energy efficiency with Programmable Logic

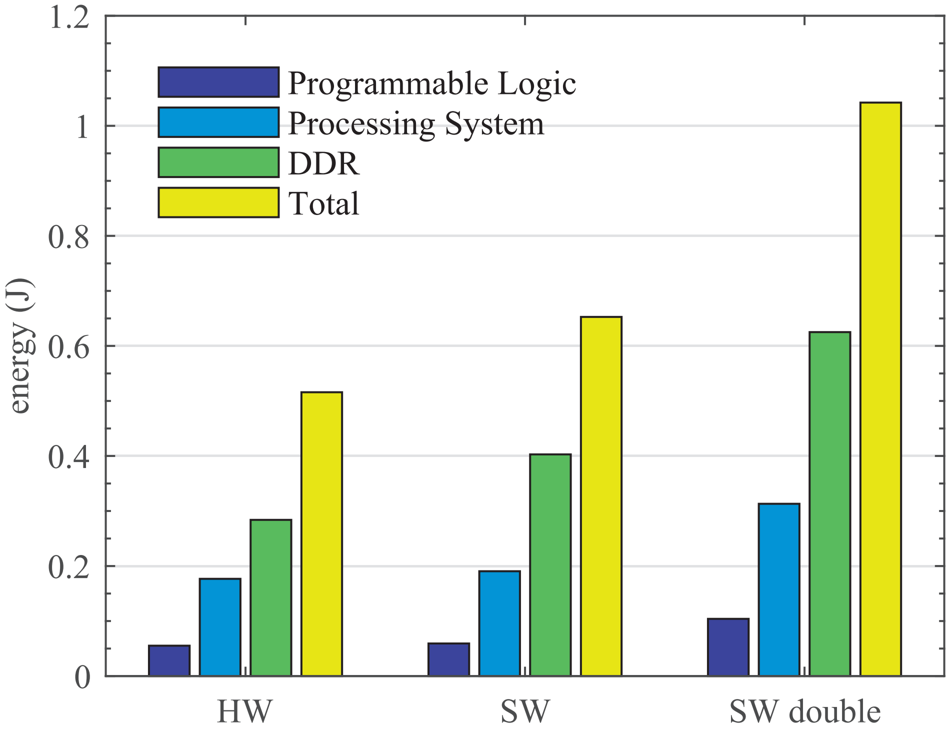
Another way to improve energy efficiency is to use specialized hardware. Unfortunately, general purpose CPUs cannot be equipped with dedicated accelerators covering all the possible applications. Moreover, all these applications might require different precision modes, and designing efficient circuits with reconfigurable precision is very challenging. One way to overcome this hurdle is to associate those CPUs with Field Programmable Gate Arrays (FPGA) in order to have reconfigurable application-oriented hardware accelerators. This was first demonstrated in the Microsoft Catapult project, which consisted in building a data center made of CPUs interconnected with FPGAs for increasing speed and energy efficiency of Bing search applications [8].

The Newton method has been implemented on a Xilinx Zynq-7000 SoC. This chip implements a dual ARM Cortex-A9 core together with programmable logic which makes it a perfect candidate for hardware-software co-design. The board used in this experiment is a Xilinx ZC702 which features 1GB of DDR3 memory and TI power controllers capable of measuring the power consumed by the different elements on the board.

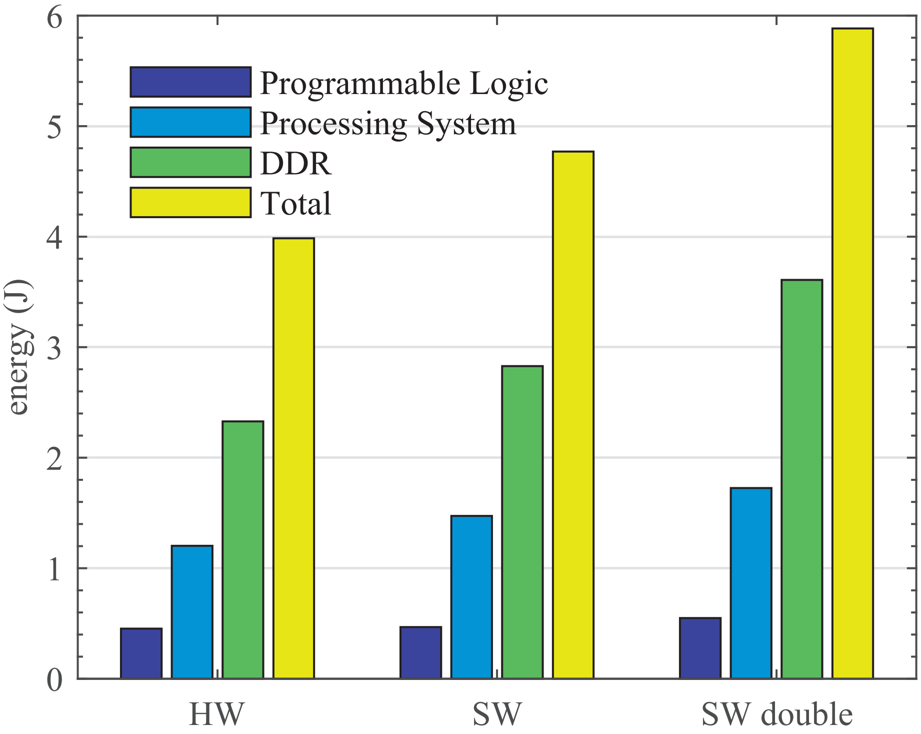
The Newton Fortran code has been translated to C code in order to use Xilinx's SDSoC tool to profile the application and automatically generate hardware accelerators using high-level synthesis. The function evaluating the residual Eq. 6.3 has been identified as the most time consuming and has therefore been transferred to programmable logic.

The power consumption of the DDR memory, the processing system (dual core ARM processor) and the programmable logic have been sampled during the program's execution using the on-board TI power controllers. These power numbers have been averaged and multiplied by the execution time to get the energy consumption. Since SDSoC does not support double precision for hardware acceleration, only single-precision problems have been accelerated.

Fig. 6.4a and 6.4b show the energy consumed on the ZC702 board for the Laplace and for the Rosenbrock problems, respectively. The energy consumption is reported for software double precision, software single precision, meaning that the full code is running on the ARM processor, and for hardware single precision for which the evaluation of the residual Eq. 6.3 is done on the programmable logic with an accelerator. For both, Laplace and Rosenbrock, around 60 % energy is saved by switching from double-precision to single-precision software. Moving the evaluation of the residual allows for an additional 20 % energy savings. Similarly to Intel's processor, DDR memory accesses seem to dominate the energy consumption.



(a) Laplace



(b) Rosenbrock

Figure 6.4 – Energy consumed by the Newton applications on the Xilinx ZC702 board with a problem size of 100000. Convergence accuracy 10^{-5}

6.5 Concluding remarks

This chapter studied the EPIs on an Intel core i7 4770 processors. It has been shown that data movement instructions consume 5 X to 180 X more energy than any vector or scalar arithmetic instruction. The EPIs have been used to build an energy model and study the effect of precision reduction on a Newton method for solving nonlinear equations. The energy breakdown shows that about 80 % is consumed by data movements, mainly L1 cache hits, while arithmetic instructions consume only 6 % of the total energy. Solving nonlinear equations might not be representative of all typical applications, however, other applications have shown similar energy distributions, or even higher fraction of the energy consumed by data movements, particularly for image and video processing. To that extent, approximate adders are certainly not ideal candidates to improve the overall energy consumption of such systems, unless they exploit bitwidth reduction. It is however shown that FPGAs can be associated to a general purpose processor in order to improve the efficiency by implementing application specific hardware accelerators. With the single-precision Newton method used in this chapter, it has been shown that 20 % energy can be saved by moving part of the algorithm to the programmable logic on the Xilinx Zynq platform. Future work should study if a similar improvement can be achieved by associating a PCI Express FPGA to an Intel processor, which has an architecture similar to the ones that are used in data centres for high performance applications.

References

[1] Cachegrind: a cache and branch-prediction profiler. URL http://valgrind.org/docs/manual/cg-manual.html.

[2] Intel software development emulator. URL https://software.intel.com/en-us/articles/intel-software-development-emulator.

[3] Benjamin Barrois, Olivier Sentieys, and Daniel Menard. The hidden cost of functional approximation against careful data sizing–a case study. In *EEE/ACM Design Automation and Test in Europe (DATE)*, 2017.

[4] Vincent Camus, Jeremy Schlachter, Christian Enz, Michael Gautschi, and Frank K Gurkaynak. Approximate 32-bit floating-point unit design with 53% power-area product reduction. In *European Solid-State Circuits Conference, ESSCIRC Conference 2016: 42nd*, pages 465–468. IEEE, 2016.

[5] Mike Fagan, Jérémy Schlachter, Kazutomo Yoshii, Sven Leyffer, Krishna Palem, Marc Snir, Stefan Wild, and Christian Enz. Overcoming the Power Wall by Exploiting Inexactness and Emerging COTS Architectural. In *29th IEEE INTERNATIONAL SYSTEM-ON-CHIP CONFERENCE*, 2016.

[6] Victor Lee. Instruction Based Energy Model. URL http://www.prism.gatech.edu/~{}gtg417r/micro47/downloads/Section6.pdf.

[7] Sven Leyffer, Stefan M Wild, Mike Fagan, Marc Snir, Krishna Palem, Kazutomo Yoshii, and Hal Finkel. Doing moore with less–leapfrogging moore's law with inexactness for supercomputing. *arXiv preprint arXiv:1610.02606*, 2016.

[8] Andrew Putnam, Adrian M Caulfield, Eric S Chung, Derek Chiou, Kypros Constantinides, John Demme, Hadi Esmaeilzadeh, Jeremy Fowers, Gopi Prashanth Gopal, Jan Gray, et al. A reconfigurable fabric for accelerating large-scale datacenter services. In *Computer Architecture (ISCA), 2014 ACM/IEEE 41st International Symposium on*, pages 13–24. IEEE, 2014.

[9] Yakun Sophia Shao and David Brooks. Energy characterization and instruction-level energy model of intel's xeon phi processor. In *Proceedings of the 2013 International Symposium on Low Power Electronics and Design*, pages 389–394. IEEE Press, 2013.

7 Conclusion

7.0.1 Personal contributions

The first two chapters of this thesis presented two techniques to trade accuracy of computation against significant energy delay and area savings. The first technique, gate-level pruning, consists in removing selected gates from any conventional circuit using the significance times activity product criterion. A fully automatized CAD tool has been developed and integrated in a standard digital flow. The GLP methodology has been applied on several arithmetic circuits such as adders and multipliers synthesized at different frequencies. The gains that can be achieved are strongly dependent of the frequency as different timing constraints lead the synthesis tool to generate different circuit architectures. It has been shown that up to 78 % PDAP reduction can be achieved at 10 % mean relative error for 32-bit adders. Unfortunately, at such high error levels, truncation clearly outperforms GLP. However, pruning remains interesting at low error levels as it offers a much larger design space than truncation. It is therefore recommended to first use truncation, and then GLP for fine-tuning.

The second technique is called Inexact Speculative Adder and consists in cutting the carry chain in multiple position on a long adder circuit. The error can be controlled by using and varying the size of the speculation and compensation blocks. This technique leads to similar results than GLP in terms of PDAP reduction, but the error produced by these approximate adders are of different nature than errors resulting from GLP. Combining ISA and GLP therefore leads to even higher PDAP savings: up to 86 % PDAP reduction for similar error levels (1 % $\mathrm{RE}_{\mathrm{RMS}}$).

Chapter 5 then showed how approximate circuit techniques can be applied at the level of a hardware accelerator or a full application. GLP has been applied on a state of the art Discrete Cosine Transform for image encoding, which is made of several adders, subtractors, register and memories. Despite arithmetic circuit occupy less than 4 % of the total DCT area, the re-synthesis of the entire DCT with pruned operators enables up to 12 % area and 10 % power savings of the entire system at the cost of 24 dB loss in the image PSNR. Gate-Level Pruning and Inexact Speculative adders and multipliers have also been combined together and implemented in the mantissa datapath of three approximate single-precision FPUs. The 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 errors of different nature, pruning and speculation can be combined to achieve 27 % power, 36 % area and 53 % power-area product reductions. The use of those FPUs has been validated by running a floating-point-intensive tone-mapping algorithm on high-dynamic range 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 on zero. Unfortunately, the power consumption of the full chip does not really benefit from the power savings achieved on each of the approximate FPU: those are masked by the overhead of the cores, registers and memories. Indeed, each FPU occupies less than 1 % of the total chip area. It is therefore worth questioning whether complex hardware approximation techniques such as ISA or GLP have a real advantage over truncation. Indeed, by building a simplistic power model, it has been shown that up to 70 % energy could be saved at a full processor scale for a weather modelling application by using truncated arithmetic circuits , registers and memory. This result has been obtained with a non-standard floating-point format. Since different application may require different number format and bitwidth, building a general-purpose processor that would efficiently run those different applications with different wordlengths would be very challenging, if not impossible.

Chapter 6 investigates which are the most consuming elements in a computing system, the energy consumptions of the main instructions in an Intel processor have been measured and an energy model has been built. It has been shown that data movement instructions consume 5X to 180X more energy than any vector or scalar arithmetic instruction. Moreover, for a Newton application, about 80 % of the total energy is consumed by data movements, mainly L1 cache hits, while arithmetic instructions consume only around 6 % of the total energy. Any approximation technique that does not alleviate memory accesses would therefore only bring marginal gains. In this regard, it is shown that by switching from double-precision to single-precision data type, the energy consumption of all the instructions are reduced by a factor 1.6 X to 2 X. Moreover, the total number of instructions is reduced thanks to vectorization capabilities of modern processors, leading to 60 % energy saving over the program's execution. Even if those savings are significant, they could still be further improved by using application specific hardware accelerators. Unfortunately, a general-purpose processor cannot be equipped with an infinite number of hardware accelerators to cover all the possible applications. As a way to overcome this issue, hardware accelerators can be implemented on programmable logic (FPGA) interconnected to general-purpose processors. As a proof of concept, the Newton application has been implemented on the Xilinx Zynq System on Chip, which is made of a dual-core Arm processor interconnected with programmable logic. By moving the most power hungry functions to the FPGA, an additional 20 % energy can be saved without reducing the precision at all.

7.0.2 Final conclusion and future work

The energy-efficiency of computing systems can be strongly improved by means of approximation techniques ranging from hardware design to software development. Those techniques can lead to a significant reduction off the total energy consumption, but only if the number of bits are reduced. This can be achieved for instance by means of bitwidth reduction on the hardware side, or data-type change on the software side. Indeed, any approximation technique that does not alleviate memory and register transfers will only bring marginal savings at a system scale since memory and registers are dominant in terms of area and energy consumption, whereas arithmetic circuit usually only occupy a small fraction of a chip. This work mostly focused on adders, but multipliers are also very good candidates for approximations since they generally occupy a much larger silicon area (typically, up to a factor 10 for a 32-bit wordlength) and consume more energy. Intuitively, one could think that recent machine learning application which require a huge amount of arithmetic operations would not suffer from the same problem. However, Google's Tensor Processing Unit (TPU), which is a chip devoted to machine learning and used in data centers, seems to demonstrate the opposite as it is still memory limited [1].

The general methodology to design efficiently approximate hardware therefore consists in applying truncation as a first step to find the optimal bitwidth, and then use one or more advanced approximation techniques such as GLP and ISA for fine-tuning. This approach leads to a drastic reduction of the energy consumption, but suffers from the fact that the precision of the circuit is set at the design time and cannot be changed over the lifetime of the chip. On the other hand, programming techniques offer much more flexibility, even allowing to tune the precision on the fly in different functions of a given program.

General purpose processors are great in the sense that they can run any application, but this also means that they are not optimized for any specific application. To overcome this, hardware accelerators can be integrated with CPUs to perform some specific task with a higher energy efficiency and within a shorter runtime. However, silicon area is limited and a single CPU cannot embed hardware accelerators covering all the possible applications. In this regard, an FPGA is a perfect companion for a CPU as it can be used to implement reconfigurable hardware accelerators to fit all kind of application needs. This thesis presented preliminary results in this direction by solving nonlinear equations on a Xilinx Zynq SoC. While this platform is well suited for IoT and industry 4.0 applications, future work should study how Intel processors and FPGAs with run-time precision tuning behave for large-scale high-performance applications such as climate modelling for instance.

References

[1] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. *arXiv preprint arXiv:1704.04760*, 2017.

References

[1] Cachegrind: a cache and branch-prediction profiler. URL http://valgrind.org/docs/manual/cg-manual.html.

[2] Intel software development emulator. URL https://software.intel.com/en-us/articles/intel-software-development-emulator.

[3] Benjamin Barrois, Olivier Sentieys, and Daniel Menard. The hidden cost of functional approximation against careful data sizing–a case study. In *EEE/ACM Design Automation and Test in Europe (DATE)*, 2017.

[4] P. Bauer, A. Thorpe, and G. Brunet. The quiet revolution of numerical weather prediction. *Nature*, pages 47–55, September 15 2015.

[5] M Blackburn. Program description for the multi-level global spectral model. *University of Reading, Dept. of Meteorology, Atmospheric Modelling Group*, 1985.

[6] Shekhar Borkar. Designing reliable systems from unreliable components: the challenges of transistor variability and degradation. volume 25, pages 10–16. IEEE, 2005.

[7] Vincent Camus, Jeremy Schlachter, and Christian Enz. Energy-efficient inexact speculative adder with high performance and accuracy control. In *Circuits and Systems (ISCAS), 2015 IEEE International Symposium on*, pages 45–48. IEEE, 2015.

[8] Vincent Camus, Jeremy Schlachter, and Christian Enz. A low-power carry cut-back approximate adder with fixed-point implementation and floating-point precision. In *Proceedings of the 53rd Annual Design Automation Conference*, page 127. ACM, 2016.

[9] Vincent Camus, Jeremy Schlachter, Christian Enz, Michael Gautschi, and Frank K Gurkaynak. Approximate 32-bit floating-point unit design with 53% power-area product reduction. In *European Solid-State Circuits Conference, ESSCIRC Conference 2016: 42nd*, pages 465–468. IEEE, 2016.

[10] Vincent Camus, Jeremy Schlachter, Mattia Cacciotti, and Christian Enz. Timing optimization using artificial false paths and application with the carry cut-back adder. In *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS), submitted*, 2018.

[11] Lakshmi N Chakrapani, Pinar Korkmaz, Bilge ES Akgul, and Krishna V Palem. Probabilistic system-on-a-chip architectures. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 12(3):29, 2007.

[12] Lakshmi NB Chakrapani, Kirthi Krishna Muntimadugu, Avinash Lingamneni, Jason George, and Krishna V Palem. Highly energy and performance efficient

embedded computing through approximately correct arithmetic: A mathematical foundation and preliminary experimental validation. In *Proceedings of the 2008 international conference on Compilers, architectures and synthesis for embedded systems*, pages 187–196. ACM, 2008.

[13] Ik Joon Chang, Debabrata Mohapatra, and Kaushik Roy. A priority-based 6t/8t hybrid sram architecture for aggressive voltage scaling in video applications. *IEEE transactions on circuits and systems for video technology*, 21(2):101–112, 2011.

[14] S Cheemalavagu, Pinar Korkmaz, and Krishna V Palem. Ultra low-energy computing via probabilistic algorithms and devices: Cmos device primitives and the energy-probability relationship. In *Proc. of The 2004 International Conference on Solid State Devices and Materials*, pages 402–403, 2004.

[15] Suresh Cheemalavagu, Pinar Korkmaz, Krishna V Palem, Bilge ES Akgul, and Lakshmi N Chakrapani. A probabilistic cmos switch and its realization by exploiting noise. In *IFIP International Conference on VLSI*, pages 535–541, 2005.

[16] Francesco Conti, Davide Rossi, Antonio Pullini, Igor Loi, and Luca Benini. Energy-efficient vision on the pulp platform for ultra-low power parallel computing. In *Signal Processing Systems (SiPS), 2014 IEEE Workshop on*, pages 1–6. IEEE, 2014.

[17] Vítor A Coutinho, Renato J Cintra, Fábio M Bayer, Sunera Kulasekera, and Arjuna Madanayake. A multiplierless pruned dct-like transformation for image and video compression that requires ten additions only. *Journal of Real-Time Image Processing*, pages 1–9, 2015.

[18] S. Das, C. Tokunaga, S. Pant, Wei-Hsiang Ma, S. Kalaiselvan, K. Lai, D.M. Bull, and D.T. Blaauw. Razorii: In situ error detection and correction for pvt and ser tolerance. *Solid-State Circuits, IEEE Journal of*, 44(1):32–48, Jan 2009. ISSN 0018-9200. doi: 10.1109/JSSC.2008.2007145.

[19] Peter Düben, Jeremy Schlachter, Parishkrati, Sreelatha Yenugula, John Augustine, Christian Enz, K. Palem, and T. N. Palmer. Opportunities for energy efficient computing: A study of inexact general purpose processors for high-performance and big-data applications. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, DATE '15, pages 764–769, San Jose, CA, USA, 2015. EDA Consortium. ISBN 978-3-9815370-4-8. URL http://dl.acm.org/citation.cfm?id=2755753.2755927.

[20] Peter D Düben and TN Palmer. Benchmark tests for numerical weather forecasts on inexact hardware. *Monthly Weather Review*, 142(10):3809–3829, 2014.

[21] Peter D Düben, Jaume Joven, Avinash Lingamneni, Hugh McNamara, Giovanni De Micheli, Krishna V Palem, and TN Palmer. On the use of inexact, pruned hardware in atmospheric modelling. *Phil. Trans. R. Soc. A*, 372(2018):20130276, 2014.

[22] Peter D Düben, Hugh McNamara, and Tim N Palmer. The use of imprecise processing to improve accuracy in weather & climate prediction. *Journal of Computational Physics*, 271:2–18, 2014.

[23] Dan Ernst, Nam Sung Kim, Shidhartha Das, Sanjay Pant, Rajeev Rao, Toan Pham, Conrad Ziesler, David Blaauw, Todd Austin, et al. Razor: A low-power pipeline based on circuit-level timing speculation. In *Microarchitecture, 2003. MICRO-36. 36th Annual IEEE/ACM International Symposium on*, pages 7–18. IEEE, 2003.

[24] Mike Fagan, Jérémy Schlachter, Kazutomo Yoshii, Sven Leyffer, Krishna Palem, Marc Snir, Stefan Wild, and Christian Enz. Overcoming the Power Wall by Exploiting Inexactness and Emerging COTS Architectural. In *29th IEEE INTERNATIONAL SYSTEM-ON-CHIP CONFERENCE*, 2016.

[25] Jason George, Bo Marr, Bilge ES Akgul, and Krishna V Palem. Probabilistic arithmetic and energy efficient embedded signal processing. In *Proceedings of the 2006 international conference on Compilers, architecture and synthesis for embedded systems*, pages 158–168. ACM, 2006.

[26] Swaroop Ghosh, Swarup Bhunia, and Kaushik Roy. Crista: A new paradigm for low-power, variation-tolerant, and adaptive circuit synthesis using critical path isolation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(11):1947–1956, 2007.

[27] Vaibhav Gupta, Debabrata Mohapatra, Sang Phill Park, Anand Raghunathan, and Kaushik Roy. Impact: imprecise adders for low-power approximate computing. In *Proceedings of the 17th IEEE/ACM international symposium on Low-power electronics and design*, pages 409–414. IEEE Press, 2011.

[28] Vaibhav Gupta, Debabrata Mohapatra, Anand Raghunathan, and Kaushik Roy. Low-power digital signal processing using approximate adders. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(1):124–137, 2013.

[29] Vaibhav Gupta, Debabrata Mohapatra, Anand Raghunathan, and Kaushik Roy. Low-power digital signal processing using approximate adders. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 32(1):124–137, 2013.

[30] Soheil Hashemi, R Bahar, and Sherief Reda. Drum: A dynamic range unbiased multiplier for approximate applications. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 418–425. IEEE Press, 2015.

[31] M. Horowitz, E. Alon, D. Patil, S. Naffziger, R. Kumar, and K. Bernstein. Scaling, Power, and the Future of CMOS. In *Electron Devices Meeting, 2005. IEDM Technical Digest. IEEE International*, pages 7 pp.–15, Dec 2005. doi: 10.1109/IEDM.2005.1609253.

[32] BJ Hoskins and AJ Simmons. A multi-layer spectral model and the semi-implicit method. *Quarterly Journal of the Royal Meteorological Society*, 101(429):637–655, 1975.

[33] 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.

[34] IN James and JP Dodd. A simplified global circulation model. *User's manual. Department of Meteorology, University of Reading*, 1993.

[35] Simona Jankowski, James Covello, Heather Bellini, Joe Ritchie, and Daniela Costa. The internet of things: Making sense of the next mega-trend. *Goldman Sachs*, 2014.

[36] Andrew B Kahng and Seokhyeong Kang. Accuracy-configurable adder for approximate arithmetic designs. In *Proceedings of the 49th Annual Design Automation Conference*, pages 820–825. ACM, 2012.

[37] Georgios Karakonstantis and Kaushik Roy. Voltage over-scaling: A cross-layer design perspective for energy efficient systems. In *Circuit Theory and Design (ECCTD), 2011 20th European Conference on*, pages 548–551. IEEE, 2011.

[38] Georgios Karakonstantis, Nilanjan Banerjee, and Kaushik Roy. Process-variation resilient and voltage-scalable dct architecture for robust low-power computing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 18(10):1461–1470, 2010.

[39] Georgios Karakonstantis, Christoph Roth, Christian Benkeser, and Andreas Burg. On the exploitation of the inherent error resilience of wireless systems under unreliable silicon. In *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, pages 510–515. Ieee, 2012.

[40] Yongtae Kim, Yong Zhang, and Peng 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*, pages 130–137, Nov 2013. doi: 10.1109/ICCAD.2013.6691108.

[41] C. M. Kirsch and H. Payer. Incorrect Systems: It's not the Problem, It's the Solution. In *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, pages 913–917, June 2012.

[42] Pinar Korkmaz, Bilge ES Akgul, Krishna V Palem, and Lakshmi N Chakrapani. Advocating noise as an agent for ultra-low energy computing: Probabilistic complementary metal–oxide–semiconductor devices and their characteristics. *Japanese journal of applied physics*, 45(4S):3307, 2006.

[43] Philipp Klaus Krause and Ilia Polian. Adaptive voltage over-scaling for resilient applications. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*, pages 1–6. IEEE, 2011.

[44] Parag Kulkarni, Puneet Gupta, and Milos Ercegovac. Trading accuracy for power with an underdesigned multiplier architecture. In *VLSI Design (VLSI Design), 2011 24th International Conference on*, pages 346–351. IEEE, 2011.

[45] Victor Lee. Instruction Based Energy Model. URL http://www.prism.gatech.edu/~{}gtg417r/micro47/downloads/Section6.pdf.

[46] Sven Leyffer, Stefan M Wild, Mike Fagan, Marc Snir, Krishna Palem, Kazutomo Yoshii, and Hal Finkel. Doing moore with less–leapfrogging moore's law with inexactness for supercomputing. *arXiv preprint arXiv:1610.02606*, 2016.

[47] Sheng Li, Jung Ho Ahn, Richard D Strong, Jay B Brockman, Dean M Tullsen, and Norman P Jouppi. Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, pages 469–480. IEEE, 2009.

[48] A Lingamneni, C. Enz, J. L Nagel, K. Palem, and C. Piguet. Energy parsimonious circuit design through probabilistic pruning. In *Design, Automation Test in Europe Conference (DATE), 2011*, pages 1–6, March 2011. doi: 10.1109/DATE.2011.5763130.

[49] Avinash Lingamneni, Christian Enz, Krishna Palem, and Christian Piguet. Parsimonious circuits for error-tolerant applications through probabilistic logic minimization. In *Integrated Circuit and System Design. Power and Timing Modeling, Optimization, and Simulation*, pages 204–213. Springer, 2011.

[50] Avinash Lingamneni, Kirthi Krishna Muntimadugu, Christian Enz, Richard M. Karp, Krishna V. Palem, and Christian Piguet. Algorithmic methodologies for ultra-efficient inexact architectures for sustaining technology scaling. In *Proceedings of the 9th Conference on Computing Frontiers*, CF '12, pages 3–12, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1215-8. doi: 10.1145/2212908.2212912. URL http://doi.acm.org/10.1145/2212908.2212912.

[51] Avinash Lingamneni, Christian Enz, Krishna Palem, and Christian Piguet. Synthesizing parsimonious inexact circuits through probabilistic design techniques. *ACM Transactions on Embedded Computing Systems (TECS)*, 12(2s):93, 2013.

[52] Tong Liu and Shih-Lien Lu. Performance Improvement with Circuit-level Speculation. In *Microarchitecture, 2000. MICRO-33. Proceedings. 33rd Annual IEEE/ACM International Symposium on*, pages 348–355, 2000. doi: 10.1109/MICRO.2000.898084.

[53] Weiqiang Liu, Linbin Chen, Chenghua Wang, Máire O'Neill, and Fabrizio Lombardi. Design and analysis of inexact floating-point adders. *IEEE Transactions on Computers*, 65(1):308–314, 2016.

[54] Hamid Reza Mahdiani, Ali Ahmadi, Sied Mehdi Fakhraie, and Caro Lucas. Bio-inspired imprecise computational blocks for efficient vlsi implementation of soft-computing applications. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 57(4):850–862, 2010.

[55] Amir Momeni, Jie Han, Paolo Montuschi, and Fabrizio Lombardi. Design and analysis of approximate compressors for multiplication. *Computers, IEEE Transactions on*, 64(4):984–994, 2015.

[56] Nathan Moroney. Local color correction using non-linear masking. In *Color Imaging Conference (CIC), 8th IS&T/SID*, pages 108–111, Nov 2000.

[57] Takanori Okuma, Yun Cao, Masanori Muroyama, and Hiroto Yasuura. Reducing access energy of on-chip data memory considering active data bitwidth. In *Proceedings of the 2002 international symposium on Low power electronics and design*, pages 88–91. ACM, 2002.

[58] K. Palem, A. Lingamneni, C. Enz, and C. Piguet. Why design reliable chips when faulty ones are even better. In *ESSCIRC (ESSCIRC), 2013 Proceedings of the*, pages 255–258, Sept 2013. doi: 10.1109/ESSCIRC.2013.6649121.

[59] K. V. Palem. Energy aware computing through probabilistic switching: a study of limits. *IEEE Transactions on Computers*, 54(9):1123–1137, Sept 2005. ISSN 0018-9340. doi: 10.1109/TC.2005.145.

[60] Krishna V Palem. Computational proof as experiment: Probabilistic algorithms from a thermodynamic perspective. In *Verification: Theory and Practice*, pages 524–547. Springer, 2003.

[61] Krishna V Palem. Energy aware computing through probabilistic switching: A study of limits. *IEEE Transactions on Computers*, 54(9):1123–1137, 2005.

[62] Krishna V Palem, Lakshmi NB Chakrapani, Zvi M Kedem, Avinash Lingamneni, and Kirthi Krishna Muntimadugu. Sustaining moore's law in embedded computing through probabilistic and approximate design: retrospects and prospects. In *Proceedings of the 2009 international conference on Compilers, architecture, and synthesis for embedded systems*, pages 1–10. ACM, 2009.

[63] Andrew Putnam, Adrian M Caulfield, Eric S Chung, Derek Chiou, Kypros Constantinides, John Demme, Hadi Esmaeilzadeh, Jeremy Fowers, Gopi Prashanth Gopal, Jan Gray, et al. A reconfigurable fabric for accelerating large-scale datacenter services. In *Computer Architecture (ISCA), 2014 ACM/IEEE 41st International Symposium on*, pages 13–24. IEEE, 2014.

[64] J. Schlachter, V. Camus, C. Enz, and K.V. Palem. Automatic generation of inexact digital circuits by gate-level pruning. In *Circuits and Systems (ISCAS), 2015 IEEE International Symposium on*, pages 173–176, May 2015. doi: 10.1109/ISCAS.2015.7168598.

[65] Jeremy Schlachter, Vincent Camus, and Christian 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*, pages 476–480. IEEE, 2015.

[66] Yakun Sophia Shao and David Brooks. Energy characterization and instruction-level energy model of intel's xeon phi processor. In *Proceedings of the 2013 International Symposium on Low Power Electronics and Design*, pages 389–394. IEEE Press, 2013.

[67] AJ Simmons and BJ Hoskins. A comparison of spectral and finite-difference simulations of a growing baroclinic wave. *Quarterly Journal of the Royal Meteorological Society*, 101(429):551–565, 1975.

[68] Tom Simonite. Intel puts the brakes on moore s law. *MIT Technology Review (Mar. 2016)*, 2016.

[69] David Wang, Brinda Ganesh, Nuengwong Tuaycharoen, Kathleen Baynes, Aamer Jaleel, and Bruce Jacob. Dramsim: a memory system simulator. *ACM SIGARCH Computer Architecture News*, 33(4):100–107, 2005.

[70] Matthew Weber, Mateja Putic, Hang Zhang, John Lach, and Jiawei Huang. Balancing adder for error tolerant applications. In *Circuits and Systems (ISCAS), 2013 IEEE International Symposium on*, pages 3038–3041. IEEE, 2013.

[71] Zhixi Yang, Ajaypat Jain, Jinghang Liang, Jie Han, and Fabrizio Lombardi. Approximate xor/xnor-based adders for inexact computing. In *Nanotechnology (IEEE-NANO), 2013 13th IEEE Conference on*, pages 690–693. IEEE, 2013.

[72] Sungwook Yu and EE Swartziander. Dct implementation with distributed arithmetic. *IEEE transactions on Computers*, 50(9):985–991, 2001.

[73] Ning Zhu, Wang Ling Goh, and Kiat Seng Yeo. An enhanced low-power high-speed adder for error-tolerant application. In *Integrated Circuits, ISIC'09. Proceedings of the 2009 12th International Symposium on*, pages 69–72. IEEE, 2009.

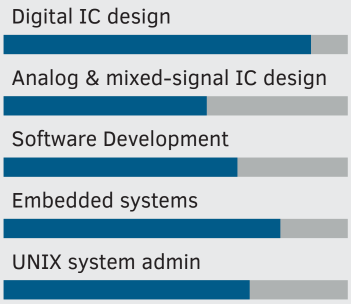
[74] Ning Zhu, Wang-Ling Goh, Gang Wang, and Kiat-Seng Yeo. Enhanced Low-power High-speed Adder for Error-tolerant Application. In *SoC Design Conference (ISOCC), 2010 International*, pages 323–327, Nov 2010. doi: 10.1109/SOCDC.2010.5682905.

Jérémy Schlachter

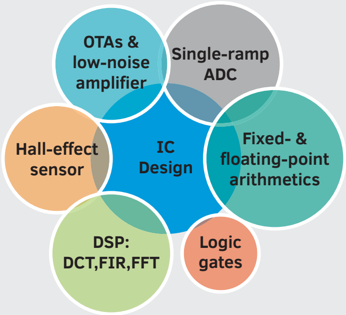
IC Design Engineer

ℹ French, born 1990
🌐 linkedin.com/in/jschlachter/
@ jeremy.schlachter68@gmail.com

Skills



Designed



Reference

Prof. Christian Enz
Thesis director, EPFL
christian.enz@epfl.ch

Hobbies

Mountain biking, alpine skiing, hiking, squash, basketball
Homebrewing
Traveling

Education

2013-2017 **PhD, Approximate Digital Circuit Design** EPFL
Integrated Circuit Laboratory, Neuchâtel

2011-2013 **MSc, Micro and Nanoelectronics** University of Strasbourg
Strasbourg, France

2008-2011 **BSc, Physics and Electrical Engineering** University of Strasbourg
Strasbourg, France

Industry-oriented trainings

Imec	Advanced Digital Physical Implementation Flow (feat. Low-Power)
Mead	Practical Aspects in Mixed-Signal ICs, High-Performance Data Converters, Low-Power Analog IC Design

Research

2013-2017 **Research Assistant** EPFL
Thesis topic: inexact and approximate digital circuit design
- Developed a Gate-Level Pruning CAD tool to trade accuracy of computation for significant area, delay and energy savings
- Designed approximate arithmetic circuits (adders, multipliers and floating-point units)
- Designed energy-efficient hardware accelerators (DCT, FFT, FIR)
- Tapped out 2 multi-core processors with approximate accelerators
- Participated in writing a successful project proposal, 1 patent and 10 peer-reviewed publications among which 2 top conferences
- Managed IT: administration of workstations and servers, installation of EDA tools and design kits

Oct-Nov '16 **Visiting Scholar** Rice University, Houton TX
Investigated techniques for improving the energy-quality behavior of applications executed on commercial processors and FPGA:
- Exploited vectorization capabilities of modern processors
- Tuned precision to improve data movement and cache behavior
- Power and energy measurements
- Developed a power model to identify main energy consumers

Experience

Feb-Aug '13 **IC design intern** Lemoptix (acquired by Intel)
Studied the eye safety aspects of a laser-based pico-projector:
- Identification of the requirements for laser class 1 certification
- Design of a rail-to-rail comparator in 180nm to detect a MEMS micro-mirror failure

2005-2012 **Factory worker (summer jobs)** Endress + Hauser, Reinach BL
Participated to the circuits production of instrumentation tools:
- Operated SMD pick-and-place production chains
- Assembled discrete components on PCBs for wave soldering

Competences

Technical Languages

HDL	VHDL, VHDL-AMS, Verilog, Verilog-A, System Verilog, UVM (notions)
Computer	Perl, Tcl, Matlab, C, Assembly, Makefile, Bash, Csh, Python, LaTeX

Software

EDA	Synopsys Design Compiler, Cadence tools, Mentor Modelsim, Spice
FPGA	Xilinx Vivado and SDSoC, Altera Quartus

Languages

French	Mother tongue
English	Fluent – Cambridge First Certificate in English
German	Fluent – Zentrale Mittelstufenprüfung - Goethe Institut

Publications

Patents

- Jeremy Schlachter and Nicolas Abelé. *A method and device for projecting an image with improved safety*. WO Patent App. PCT/EP2013/063,419. Dec. 2014. url: `www.google.ch/patents/WO2014206465A1`.

- Vincent Camus, Jeremy Schlachter, and Christian Enz. *System and Method for Optimization of Digital Circuits with Timing and Behavior Co-Designed by Introduction and Exploitation of False Paths*. Patent pending, filing date May 20, 2016. May 2016.

Journal article

- Jeremy Schlachter, Vincent Camus, Krihsna Palem, and Christian Enz. “Design and Applications of Approximate Circuits by Gate-Level Pruning”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* (2017).

Conference proceedings

- Jeremy Schlachter, Mike Fagan, Krishna Palem, and Christian Enz. “A Study on the Energy-Precision Tradeoffs on Commercially Available Processors and SoCs with an EPI Based Energy Model”. In: *2017 30th IEEE International System-on-Chip Conference (SOCC) (SOCC 2017)*. Munich, Germany, Sept. 2017.

- Jeremy Schlachter, Vincent Camus, and Christian Enz. “Design of energy-efficient discrete cosine transform using pruned arithmetic circuits”. In: *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on Circuits and Systems*. IEEE. 2016, pp. 341–344.

- Vincent Camus, Jeremy Schlachter, Christian Enz, Michael Gautschi, and Frank K Gurkaynak. “Approximate 32-bit floating-point unit design with 53% power-area product reduction”. In: *European Solid-State Circuits Conference, ESSCIRC Conference 2016: 42nd*. IEEE. 2016, pp. 465–468.

- Mike Fagan, Jeremy Schlachter, Kazutomo Yoshii, Sven Leyffer, Krishna Palem, Marc Snir, Stefan Wild, and Christian Enz. “Overcoming the Power Wall by Exploiting Inexactness and Emerging COTS Architectural”. In: *29th IEEE International System-on-Chip Conference*. 2016.

- Vincent Camus, Jeremy Schlachter, and Christian Enz. “A low-power carry cut-back approximate adder with fixed-point implementation and floating-point precision”. In: *Proceedings of the 53rd Annual Design Automation Conference*. ACM. 2016, p. 127.

- Jeremy Schlachter, Vincent Camus, and Christian Enz. “Near/sub-threshold circuits and approximate computing: The perfect combination for ultra-low-power systems”. In: *VLSI (ISVLSI), 2015 IEEE Computer Society Annual Symposium on Very Large Scale Integration*. IEEE. 2015, pp. 476–480.

- Vincent Camus, Jeremy Schlachter, and Christian Enz. “Energy-efficient digital design through inexact and approximate arithmetic circuits”. In: *New Circuits and Systems Conference (NEWCAS), 2015 IEEE 13th International New Circuits and Systems Conference*. IEEE. 2015, pp. 1–4.

- Jeremy Schlachter, Vincent Camus, Christian Enz, and Krishna V Palem. “Automatic generation of inexact digital circuits by gate-level pruning”. In: *Circuits and Systems (ISCAS), 2015 IEEE International Symposium on Circuits and Systems*. IEEE. 2015, pp. 173–176.

- Shyamsundar Venkataraman, Akash Kumar, Jeremy Schlachter, and Christian Enz. “Designing inexact systems efficiently using elimination heuristics”. In: *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*. EDA Consortium. 2015, pp. 758–763.

- Peter Düben, Jeremy Schlachter, Sreelatha Yenugula, John Augustine, Christian Enz, K Palem, TN Palmer, et al. “Opportunities for energy efficient computing: A study of inexact general purpose processors for high-performance and big-data applications”. In: *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*. EDA Consortium. 2015, pp. 764–769.