Analyse d'erreurs pour les systèmes utilisant des calculs approximés

Justine Bonnot

THESE DE DOCTORAT DE

L'INSA RENNES
COMUE UNIVERSITE BRETAGNE LOIRE

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Signal, Image, Vision*

Par

Justine BONNOT

Error Analysis for Approximate Computing Systems

Thèse présentée et soutenue à l'INSA Rennes, le 09 Octobre 2019
Unité de recherche : IETR
Thèse N° : 19ISAR 14/D19 - 14

Rapporteurs avant soutenance :

Alberto BOSIO	Professeur d'Université, Ecole Centrale de Lyon
Christophe JEGO	Professeur d'Université, Bordeaux INP

Composition du Jury :

Olivier SENTIEYS	Professeur d'Université, Université de Rennes 1, Président du Jury
Alberto BOSIO	Professeur d'Université, Ecole Centrale de Lyon, Rapporteur
Christophe JEGO	Professeur d'Université, Bordeaux INP, Rapporteur
Fabienne JEZEQUEL	Maître de conférences, HDR, Université Panthéon-Assas
Thibault HILAIRE	Maître de conférences, Sorbonne Université
Karol DESNOS	Maître de conférences, INSA Rennes
Daniel MENARD	Professeur d'Université, INSA Rennes Directeur de thèse

Intitulé de la thèse :

Error Analysis for Approximate Computing Systems

Justine BONNOT

En partenariat avec :

Contents

Acknowledgements

I would first like to thank my advisors Dr. Karol Desnos and Pr. Daniel Menard for their help and support during these three years of thesis. Karol, thank you for your help on technical and redactional details. I have really appreciated your help during the first months working on my beloved stereo matching algorithm ! Daniel, who had to work with me for a little longer, a huge thanks for your trust in me and my work, since my ≪ Parcours recherche ≫. It was really positive to have someone always behind me in the hardest moments of the PhD. Besides, all our mathematical discussions were really a pleasure, and thank you for letting me lead my PhD as I was willing to.

I also want to thank Pr Alberto Bosio and Pr. Christophe Jégo for having accepted to review this thesis. I hope it was the best-seller to read during the summer! Thanks also to Pr. Olivier Sentieys for presiding the jury, as well as to Fabienne Jézequel and Thibault Hilaire for being part of the jury of my PhD. The questions and discussions we had on this subject were really relevant and brought out new perspectives on this work.

It was also a pleasure to work with all my collegues from the VAADER team of IETR, especially the collegues that shared my office during these 3 years, as Judicaël Menant, Alexandre Mercat or Florian Lemarchand. Our musical Friday were always a pleasure and you were also there when I had bad times during my PhD. Thanks to the IETR Vaader team for being great co-workers. Also, a huge thank to Jocelyne Tremier, Corinne Calo, Aurore Gouin and Justine Gromaire for their administrative (and friendly) support. This thesis also benefited from many discussions with the team of the ANR project in which it was a real pleasure to work.

Thanks to my friends and family who always truly believed in my scientific skills despite an undeniable gift for singing. A special thank to my uncle Sylvain who, I am sure, was very proud from high above. He always encouraged me to follow my dreams. Thanks to my best friends Edgar and Lady for their (too early) morning wake ups. Thanks to Depeche Mode and especially Dave Gahan (and sorry Daniel and Karol for my fan attitude) for having unwillingly encouraged me during the hard part of writing this thesis, with your amazing songs.

Lastly, but far from least important, I would like to thank Julien, my husband, for his love, trust and support he gave me during these three years. I am really lucky to have you in my life, and hope you will be there for a lot of new challenges to come in the future.

To end with, enjoy the reading of this thesis, and most of all, enjoy the silence !

Part I

State of the Art and Motivations

CHAPTER 1

Approximate Computing: Trading-off Quality for Performance

"Le mieux est l'ennemi du bien"

Voltaire

1.1 Semiconductors and computing systems trends: towards approximate computing

The fierce competition to design faster, cheaper and more energy-efficient electronic systems is not only economical but also an urging answer in the need to save the available energy resources. Indeed, according to the Semiconductor Industry Association and Semiconductor Research Corporation, the total energy required by computing systems will exceed the estimated world's energy production by 2040, if no significant improvement is obtained in terms of energy-aware computing systems [AC15].

A boom in the computing capacities

From 1965 to 2012, the computing capacity offered by computing systems, along with the accuracy, have increased exponentially. According to Moore's law [Moo06], stated less than a decade after the invention of the silicon integrated circuit in 1958 by Jack Kilby, the number of transistors on a chip was expected to double every 2 years, which was closer to every 18 months in reality, consequently increasing the processing speed while decreasing the price of a chip. Gordon Moore made indeed quite a remarkable prediction since, as presented in Figure 1.1 the number of transistors on an integrated circuit closely followed this trend from year 1970 to 2016. Besides, the cost reduction for the chip manufacturer as well as chip user has been made possible thanks to new chemical processes to manufacture integrated circuits while optimizing the cost. Using optical lithography instead of hand painting, by the mid-1960s, the transistors packaging was more expensive than the cost of manufacturing the transistor. Analyzed by Hutcheson [Hut05], the economic version of Moore's law states that the cost per transistor is halved every 18 months. From 30 transistors on the first silicon integrated circuits, to 10 billion transistors today, the global chip production has boomed, thanks to technical factors such as the reduction in feature size, an increased yield and an increased packing density.

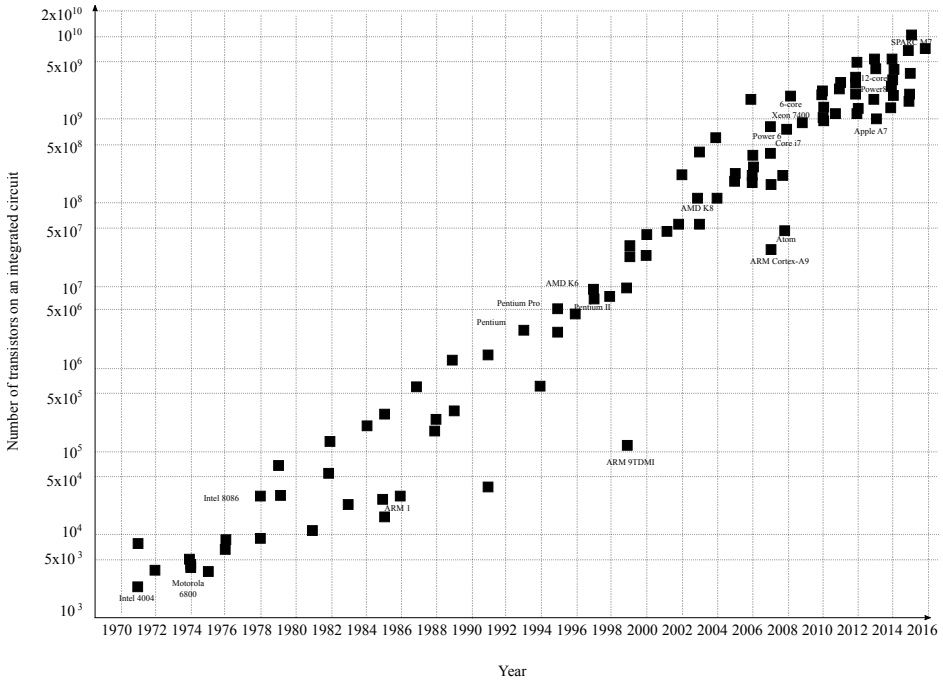


Figure 1.1 – *The facts behind Moore's law [Int].*

The ease in accessing to powerful chips induced a growth in the consumer demand for computing systems and powerful software. Nevertheless, as shown in Figure 1.1, the end of Moore's law [Ard02] has been observed since 2012. The scaling trends of CMOS technology impede to respect the predicted yields and performances. For instance, 45-nm Central Processing Units (CPUs) were expected to achieve a clock frequency of 10 GHz while in practice only achieving 3 to 4 GHz [SK00]. As Markov [Mar14] presented, computing efficiency is inherently limited in a fundamental, material, device, circuit and system/software nature. For instance, on-chip interconnect highly limits the performance of a chip. Transistors are themselves limited by the width of the dielectric gate, which has reached the size of a few atoms. Along with the increase in computation speed, the energy required for computing has also increased according to the Heisenberg uncertainty principle [BHL07] and its management becomes major concern. The Heisenberg uncertainty principle, in its energy-time form, expresses the operational limitations due to quantum mechanics as:

$$\Delta t \Delta E \geq \frac{h}{2} \tag{1.1}$$

where Δt and ΔE represents the time and energy, respectively, and h Planck's constant. That is to say, that to compute faster, more energy is required, since if Δt decreases, then ΔE has to increase for the inequality to remain true.

The Dennard's scaling prediction, that stated that the power consumption of semiconductor integrated circuits is kept constant while increasing their density, broke down in 2006.

An unreliable future hardware

Following the end of Moore's law and Dennard's scaling, a paradox has been set up: Very Large Scale Integration (VLSI) designers need to reduce their cost of chip manufacturing

to answer the growing demand, modifying their design margins but produce always more error-prone technologies. Indeed, according to Moore's law, the density of transistors on a chip increases while the number of faults in a chip increases with the density of transistors. In the search for efficiency, the constantly reducing size of manufactured chips goes along with the reduction of the operating voltage that induces a higher fault vulnerability to electrical noise as explained in [SWK+05]. Electrical noise that can also be called interference, may be an undesirable electrical signal that interferes with the original electrical signal.

Because of the fault vulnerability of currently manufactured chips, so as to ensure a strict accuracy on a result, a redundancy in the computational modules is required. This process increases all the more the energy required. The future hardware being unreliable, the process of discarding a chip because of an imperfection appears to be another part of the current paradox.

A massive quantity of data to process

The volume of information to process has also boomed over the past few years: "in 2010, during two days, the same volume of information is produced that it has been in two billion years up to 2003" (Eric Schmidt [Sie10]). Data centers are expected to deal with 175 zettabytes of data by 2025 [Ins19]. This growing volume of data to process goes along with an exponential growth in demand for storage and computing.

Nevertheless, the volume of data produced is growing twice faster than the growth of bandwidth. New ways of computing are required to deal with this massive stream of information to process. For instance, the world's largest radio telescope, the Square Kilometer Array (SKA) [DHSL09], whose construction is planned to begin in 2020, will be composed of hundred of thousands of receiving antennas. In the first phase of the SKA project, 160 terabytes of raw data per second will be generated and will have to be analyzed. Another example of this exponentially growing volume of produced data is Internet of Things (IoT) which connects together billions of computing devices. Work in energy-efficient computing is strongly needed, at all levels of computing, whether it be on sensors or on ultra low-power processors with energy harvesting.

At the same time, according to Bell's law, every decade, new classes of computing devices are getting 100 times smaller. Computing devices are required to deal with massive streams of information while shrinking in terms of size. In this context, to face with the growing volume of data to process, or the size of the chips embedded in the daily used systems, near sensor computing ([LHG+16]) has been proposed. Instead of centralizing the computations, only the meta-data are transmitted to the central processing part, while the computations are done near the sensors. LikamWa et al. [LHG+16] have proposed near sensor computing in the domain of continuous mobile vision, to overcome the energy bottleneck, with RedEye. In RedEye, the processing part has been moved into the analog domain to reduce the workload of the analog readout, the data movements and consequently, the energy required. However, near sensor computing is also subject to numerous constraints such as the operating voltage, or the bandwidth capacities. Compromises have to be made. The energy consumption of current systems is still rapidly growing as they need to process always more information.

New techniques are then searched for meeting real-time and energy constraints when designing embedded systems, and save resources during the implementation phase. In this context, Approximate Computing (AC) is one of the main approaches for post-Moore's Law computing and is under consideration in this thesis. AC uses the numerical accuracy of an application as a new tunable parameter to design more efficient systems in terms of

area, energy or speed. It exploits the error resilience of numerous applications in order to save energy or accelerate processing. AC benefits from the intrinsic error resilience of algorithms in signal, image, video processing, artificial intelligence or data mining fields.

1.2 Energy-aware computing

Several types of energy-aware computing have been proposed since the 1960s. Since power consumption has become one of the main limiting factors to increase the performance of applications, new computing methods have been proposed to optimize energy/power consumption.

1.2.1 Stochastic computing

Stochastic computing has been proposed in the 1960s [Gai69, AH13]. Stochastic computing processes data as probabilities represented by bit-streams. The probability indicates the number of $1s$ in the bit-stream without having an incidence on its position. A bit-stream is then called a stochastic number. For instance, to represent $x = 0.75$, numerous bit-streams can be used depending on the length of the bit-streams used, since x represents the probability of having a 1 in the bit-stream:

$$x = 0.75 = \begin{cases} (1,1,1,0) \\ (1,0,1,1) \\ (1,1,1,1,0,0,1,1) \end{cases}$$

Treating information as bit-streams allows using less costly, complex and energy-consuming units. Indeed, a multiply unit can be implemented with a **AND** gate.

For example, instead of processing the multiplication of $x = \frac{4}{8}$ and $y = \frac{6}{8}$, they can be represented as follows:

$$x = \frac{4}{8} = \begin{cases} (0,1,1,0,1,0,1,0) \\ (0,1,0,1,1,1,0,0) \end{cases}$$

$$y = \frac{6}{8} = \begin{cases} (1,0,1,1,1,0,1,1) \\ (1,1,1,0,1,0,1,1) \end{cases}$$

The multiplication of x and y will then be processed with an **AND** gate, giving for the first representations of x and y the result $(0,0,1,0,1,0,1,0) = \frac{3}{8}$, while for the second representations of x and y the result $(0,1,0,0,1,0,0,0) = \frac{2}{8}$.

Nevertheless, as shown in the example, depending on the stochastic number used to represent the values (x, y), the accuracy at the output of the operation is modified. To generate the stochastic number representing a data x, pseudo-random numbers generators are used. The difficulty, when generating stochastic numbers to represent probabilities, is that each bit in the bit-stream possesses the same weight, thus several stochastic numbers can represent the same probability. To compute with fair stochastic numbers, they have to be non-correlated and random enough. Correlation between stochastic numbers may generate errors during the computation. To derive error statistics on stochastic computing, the techniques of statistical inference, estimation and detection, are used. The precision of a stochastic number is represented by the number of bits used to encode the data. One of the major problems of stochastic computing is that the length of the bit-streams used to process the data increase exponentially with the accuracy required. Consequently, stochastic computing is generally slow for obtaining a relatively small precision. Despite the drawbacks of stochastic computing, the current production of unrealiable hardware

circuits, as presented in Section 1.1 pushed researchers to reconsider the computation with probabilities.

1.2.2 Quantum computing

Quantum computing is presented [Gru99] as an opening on the different forms of energy-aware computing. Quantum computing aims at reproducing the ability of subatomic particles, for instance hadrons, which can exist in more than a state at a time. Quantum computers can then be more efficient at solving particular problems than classical computers, as for instance, factoring integers or solving discrete logarithms [Sho99]. Indeed, contrary to classical computing in which the information is carried out by bits that can only exist in two distinct states, 0 or 1, in quantum computing, information is carried out by qubits, the quantum generalization of a bit. Two fundamental laws are driving quantum computing.

1. Superposition: Qubits can exist in $|0>$-state, $|1>$-state or in a linear combination of both expressed as $|\phi>= a \cdot |0> + b \cdot |1>$ where (a, b) are complex numbers.

2. Entanglement: A correlation exists between the individually random behavior of two qubits.

Qubits are a superposition of the different values and can exist in more than 2 states. A qubit can be positioned anywhere on a sphere of radius 1. The probability that the qubit is in state $|0>$ is equal to $|a|^2$, and in state $|1>$ to $|b|^2$. Consequenly, computing with qubits allows to store more information with less energy. However, the stability of qubits is still an issue, as well as cooling considerations. In 2016, IBM released a 5-qubit computer available online and programmable in Python for researchers, and has released more recently a 20-qubit quantum computer.

Nevertheless, quantum computing requires new software systems to be viable.

1.2.3 Probabilistic computing

Unlike stochastic or quantum computing, **probabilistic computing** presented by Palem et al. [Pal03] is a technique to estimate the energy savings obtained with an accuracy trade-off. Probabilistic computing links the energy consumed by a computation with its probability of being accurate. Linking computations with the notion of probability makes sense since the behavior of errors is best described with a probabilistic behavior. Palem considers that energy savings can be computed with thermodynamics laws. A bit of information equal to 0 or 1 is then called a value and is considered as a thermodynamics microstate that can be measured. Boltzmann's equation derives the energy that can be saved for a computation with a probability p of being accurate, as being $k \cdot T\ln(\frac{1}{p})$ Joules. Probabilistic computing has been strongly influenced by the Automata Theory [VN56] of Von Neumann in which he states that "error should be treated by thermodynamics methods".

1.2.4 Approximate computing

Approximate computing is the energy-aware computing technique considered in this thesis. AC is inspired by the error conception of Von Neumann, in which the tolerance to imprecision of an application is exploited. Indeed, in 1956, John Von Neumann [VN56] has studied how to constrain an error induced when building an automaton using unrealiable

components. Instead of dealing with errors with the will-power to completely eliminate them, he showed that under certain conditions, the accuracy of an operation realized by an automaton can be controled and framed. Besides, as long as this error was bounded, the automaton was able to give acceptable results despite the presence of erroneous values. According to John Von Neumann, "error is viewed as [...] an essential part of the process under consideration". Error is then treated as a required information to take into account throughout the computation process. Processors implementing AC techniques are trying to mimic the computation as done by the brain. For instance, when people are asked to quickly evaluate a percentage with mental calculus, they are working with approximations to obtain a close idea of the answer. The brain possesses approximately 10^{10} neurons, each has about 10^4 connections and requires 25 Watts of continuous power to be able to compute 25 Peta Operations per seconds. Surely the best processor ever built! In the last two decades, research on the AC paradigm has been very active. Lots of approximation techniques have been proposed, from the circuit to system level. An overview of these techniques is presented in Chapter 2. Nevertheless, a lot of work is still left to do to analyze accurately these techniques in terms of induced errors and behavior within an application, so as to make it easier for application designers to use them.

1.3 Approximate computing

1.3.1 The error resilience property

Applying AC techniques to an application requires a good knowledge of its behavior when errors occur. The error resilience property of an application is the key-point in applying AC: used in domains such as video or image processing, data mining or recognition applications. AC exploits the fact that the precision offered by an application is generally greater than the precision required. Numerous applications may output acceptable results while differing from the exact expected result, as explained in [KIK+98] for video coding algorithms. The property of error resilience, for an application, is the ability for the application, to produce acceptable results despite the fact that some computations are skipped or approximated and thus inexact. This property may be induced by several factors:

- The **end-user**: the need of the end-user or its perception may limit the need for accuracy on the output. For instance, if the end-user of the output of the application is a neural network, it can compensate the errors induced by the use of approximation. Another example, in video processing where the end-user is the human user, the human visual perception limits greatly the accuracy needed. Finally, for applications as in data mining, no golden result exist.

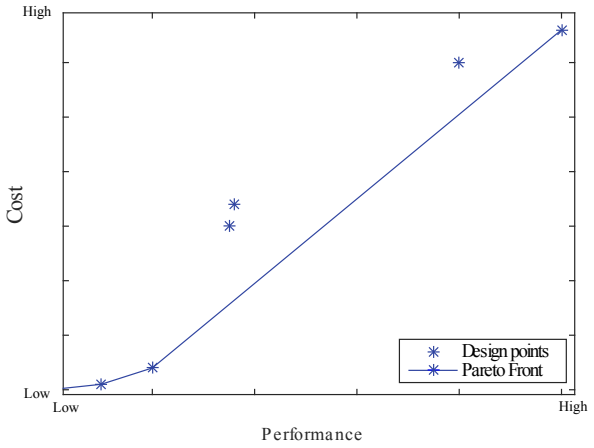
- The **input**: the input data of the application may be redundant or noisy, inherently inducing noise in the computations. Real-world data can generally be processed with approximations.

- The **computations**: the computations in the application may be statistic, probabilistic or recursive. Besides, iterative computations can compensate the errors caused by the use of an approximate module.

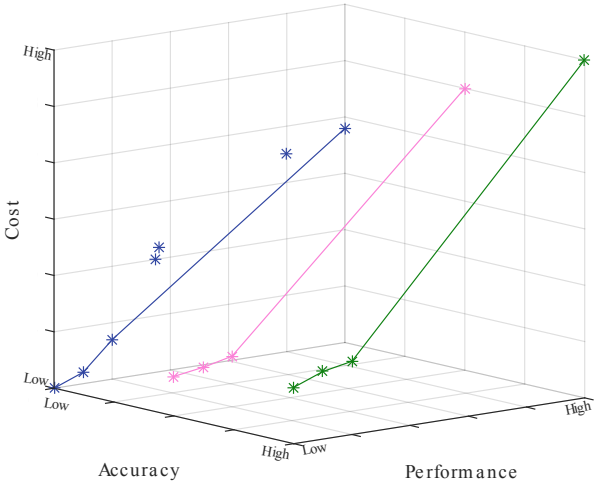
It is to be noted though that some applications are not error resilient. For instance, in control-based applications, a small error in the computations may lead to a wrong decision and in the end to a non-acceptable error at the output of the application. In addition, in

critical systems, the application has to be predictible, that is to say, for a given input, the result has to be known.

According to error resilient applications, the error between the reference application and its approximate version can be used as a margin to trade-off the performance in terms of computation time or energy consumption, and the implementation cost in terms of memory requirements or complexity as such. AC offers a new degree of freedom to design an application as illustrated in Figure 1.2.



(a) Design of an application without AC.



(b) Design of an application with AC.

Figure 1.2 – *New degree of freedom when designing an application with AC techniques.*

Nevertheless, the property of error-resilience for an application is not sufficient to apply AC techniques. Programmers have to clearly identify parts of the application that tolerate AC as well as parts where strict accuracy is required. For instance, an application with non-linear operations may not tolerate approximations near these operations. Approximations having an impact on the control flow or memory accesses may lead to disastrous behaviors as Yetim et al. [YMM13] presented with an approximation leading to a segmentation fault. A sensitivity analysis can be led to annotate the algorithm indicating the less sensitive

parts, as presented in [CCRR13]. Chippa et al. [CCRR13] proposed the Application Resilience Characterization framework to identify the error-resilient parts of applications, and characterize the error-resilient parts depending on varied AC techniques.

Furthermore, to choose an approximation strategy, a Design Space Exploration (DSE) is generally performed to find the configuration with respect to the constraints of the platform on which the algorithm is embedded. Once an approximation strategy has been chosen, the impact of the approximation on the application quality metric has to be quantified. The application quality metric, whose measurement depends on the application, quantifies the output quality of the application. For instance, for a signal processing application, the application quality metric can be the signal-to-noise ratio (SNR). For an image processing application, the application quality metric can be the Structural Similarity Index [WBSS04].

1.3.2 Different levels of approximate computing techniques

AC can be applied at different levels, as presented in Figure 1.3.

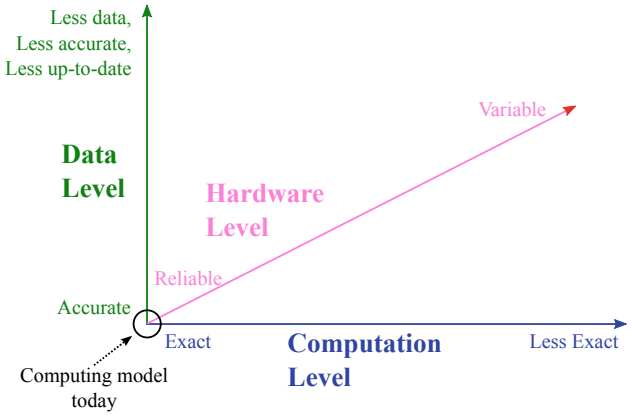


Figure 1.3 – *Different levels of approximation.*

Firstly, the approximation can be made on the **Data Level**. The number of data to process can be decreased leading to less up-to-date data or a smaller volume of data to process. Data can also be stored with a reduced precision, saving bits and thus energy, or can even be compressed. In parallel programs, the synchronization between different threads can be relaxed so as to use less up-to-date data. The different approximation techniques on data are presented in Section 2.2.

At the **Computation Level**, the algorithm itself is adjusted so as to trade off the accuracy. Computation or memory accesses skipping can be implemented. In this case, some computing-blocks are not applied. Functional approximation can also be used, approximating sophisticated mathematical functions by polynomials, or by Look-Up Table (LUT) for instance. The different approximation techniques at the computation level are presented in Section 2.3.

Finally, approximations can be done at the **Hardware Level**. In that case, alternative hardware implementations of a circuit less energy-consuming can be found, as the numerous alternative implementations of exact arithmetic operators. A circuit can also be approximated in terms of its functionning parameters: modifying the operating voltage or frequency of a circuit has an impact on the result quality and the energy consumption.

To end with, unreliable storage can be used to store error-resilient data. The different approximation techniques at the hardware level are presented in Section 2.4.

Beyond the three distinct axes of AC, Hedge et al. [HS01] have proposed a technique to identify approximable parts at each levels of design abstraction in an application. The recognition, mining and synthesis (RMS) applications are targeted with this cross-layer method, since they possess several interesting properties for AC: the statistical and aggregative algorithms average down or out the errors, and the iterative algorithms have the self-healing property.

From the programmer point of view, there remains a long way to go from the exact implementation of an application to an approximated version of the application that can be used. This is the reason why whole methodologies have been created to apply AC techniques to a whole application: from the subdivision of the application into blocks, to the computation of the energy savings, real recipes have been proposed to help programmers to use efficiently these techniques.

Identifying the level on which applying approximations is a real current challenge. Indeed, approximations may be applied from the transistor level up to the processor itself.

1.4 Scope of the thesis and contributions

To implement approximations in an application, the functionality of the application has to be guaranteed and the induced errors at the output of the application have to be framed and quantified. Tools are needed to quickly explore the different approximation perspectives and their impact on the application output quality. The difficulty to study the impact of approximations on an algorithm, is that AC algorithms are generally more intricate in terms of hardware implementation or memory accesses than their accurate implementation. The simulation of AC algorithms is more complicated and the adaptation effort to use them generally requires an entire team of system and hardware engineers. The main challenge when including AC in an application is to evaluate the impact of the approximation on the quality of service (QoS) at the output of the application. Potential approximations have to be analyzed and fine tuned to choose the best approximations with respect to the different constraints on the application implementation. The Approximation Design Space Exploration (ADSE) is large and requires a fast simulation to evaluate approximation impact on QoS. The selection of the set to simulate is also a critical point in testing the approximation perspective.

In this thesis, the objective is to propose new techniques to adress the problem of error characterization and propagation in AC systems. Methods to characterize the errors induced by the approximations are presented, as well as methods to link the induced errors to the output quality of the application. The main contributions of this thesis are:

1. A generic simulation-based framework to statistically characterize the error induced by inexact operators. This method is based on inferential statistics and extreme value theory to derive a subset to simulate according to user-defined confidence requirements. This contribution has been published in [BCDM18] and [BCDM19].

2. A fast simulator of inexact operators has been developped. The proposed simulator allows to quickly measure the impact of approximation on the application QoS. This contribution has been published in [BDM18c] and [BDM18a].

3. A new method for fixed-point noise power evaluation is proposed. The error induced by fixed-point coding is statistically characterized, allowing to compute the noise

power with an adaptive and reduced number of simulations. This contribution has been published in [BDM19a].

4. A new method based on inferential statistics for fixed-point refinement has been developed. The number of simulations is adapted depending on the required accuracy on the estimation of the noise power. This contribution has been presented as a poster in DAC19 and has led to patent number FR1903747 [BMD19].

5. A method for accuracy or quality metric inference using kriging, a geostatistical method, has been developed. The proposed method allows to reduce the number of simulation-based evaluations of the considered metric. This contribution has not been published yet.

Another contribution of this thesis has been to get involved in scientific popularization. AC has been presented into several conferences during events as the "Festival des Sciences", "En Direct du Labo" or "A la découverte de la recherche". Three articles have been written for scientific media for the general public, and are listed below:

1. A general presentation of the problematic of energy savings in IoT as well as the potential solutions offered by AC is presented in [Bon18a].

2. A short presentation of the problematic of my thesis is described in the article [Bon18b].

3. An overview of the use of IoT in medicine for rhythmology is presented in the article [Bon].

Finally, this thesis has been elected "Coup de coeur du public" at the semi-final in Rennes of the competition "Ma Thèse en 180s".

1.5 Outline of the thesis

This thesis is organized in two parts: Part I introduces the concept of AC and presents an overview of AC techniques in **Chapter 2**. Then the different error metrics proposed to characterize the errors induced by approximations are presented, as well as the models to link errors with application QoS in **Chapter 3**. Part II formally introduces and evaluates the contributions of this thesis. **Chapter 4** presents an algorithmic-level approximation technique and its behavior towards two different error metrics. Chapter 4 exposes the motivations of the thesis. **Chapter 5** presents the proposed method for the characterization of the error profile of inexact operators. **Chapter 6** details the proposed simulators to be able to quickly simulate an inexact operator and measure its impact on the output quality metric. In **Chapter 7**, a simulation-based model for the accuracy evaluation of systems converted in finite precision using inferential statistics is presented. This model is used in an optimization algorithm for the fixed-point refinement of the word-lengths of the data in an application converted in finite precision. The proposed method is based on the previously proposed statistical error model of the noise power, and uses inferential statistics to greatly reduce the number of simulations required. Seeking to go further in the improvement of the time to evaluate the finite precision effect, **Chapter 8** explains how simulations can be avoided by estimating the accuracy metric at the output of an application during fixed-point refinement using geostatistical methods as kriging. Finally, **Chapter 9** concludes this thesis and proposes potential research directions for future research.

CHAPTER 2

Background: Approximation Techniques

"Science does not claim at establishing immuable truths and eternal dogmas; its aim is to approach the truth by successive approximations, without claiming that at any stage final and complete accuracy has been achieved"

Bertrand Russell

2.1 Introduction

To optimize criteria as energy, latency and area, a recent approach is to trade-off the output application quality for the cost of the designed system as presented in [BMS18]. In this context, numerous approaches have been proposed in Approximate Computing (AC). An overview of these approaches along the three approximation axes is presented. First, the data-driven approximations are presented in Section 2.2. Then, the approximations done at the software level on the computations, are presented in Section 2.3. Finally, approximations done on the hardware structure are presented in Section 2.4. AC does not use any assumption on the stochastic nature of the underlying process of the application contrary to stochastic computing, but uses the statistical properties of the data or the algorithm. Along with these approximation techniques, methods to analyze the profile of the errors generated by the induced approximations have also been proposed, and are presented in Chapter 3.

2.2 Approximate computing techniques at the data level

The statistical properties of data can be used to implement several AC techniques. Data-driven AC techniques benefit from the reduction of the volume of data to process or the representation of the intern variables of an application that can be more energy-efficient. Different strategies are proposed to process data-driven approximations and are presented in Figure 2.1.

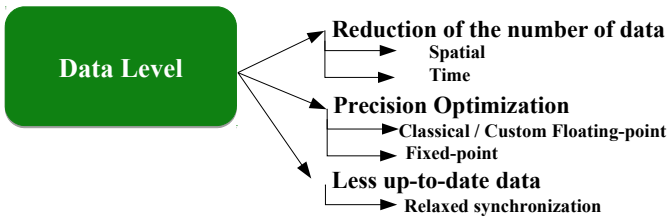
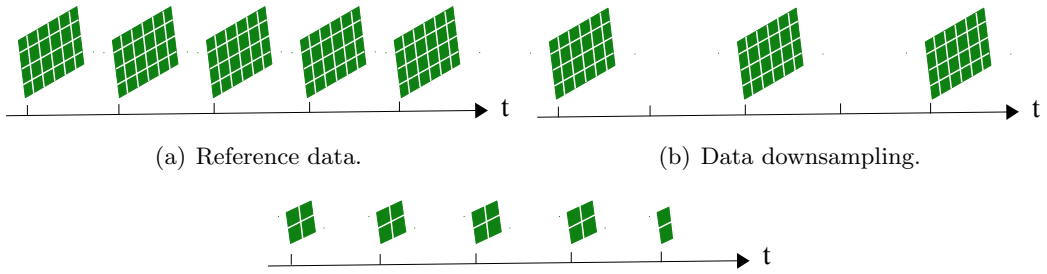


Figure 2.1 – *Approximate computing techniques: at the data level.*

2.2.1 Reduction of the amount of data to process

One may use a reduction in the amount of data to process as presented in Figure 2.2. This reduction can be done in the spatial or in the time domain. Both techniques are massively used by application designers in signal, image or video processing to meet real time and complexity constraints. Concerning the time domain, reducing the sampling rate of a signal to process allows decimating it. The reduction of the sampling rate reduces the bit rate of the process which impacts the quality of the signal to process. In Figure 2.2(b), data downsampling is done by doing the computations with a clock frequency lower than the clock frequency without data downsampling presented in Figure 2.2(a), which implies that data are present less often. When it comes to the spatial domain, the signal to process can be downscaled (fitting a signal to a lower resolution) or upscaled (fitting a signal to a higher resolution). In Figure 2.2(c), data downscaling is done by doing the computations on input data of smaller sizes compared with the original input data presented in Figure 2.2(a).



(a) Reference data.

(b) Data downsampling.

(c) Data downscaling.

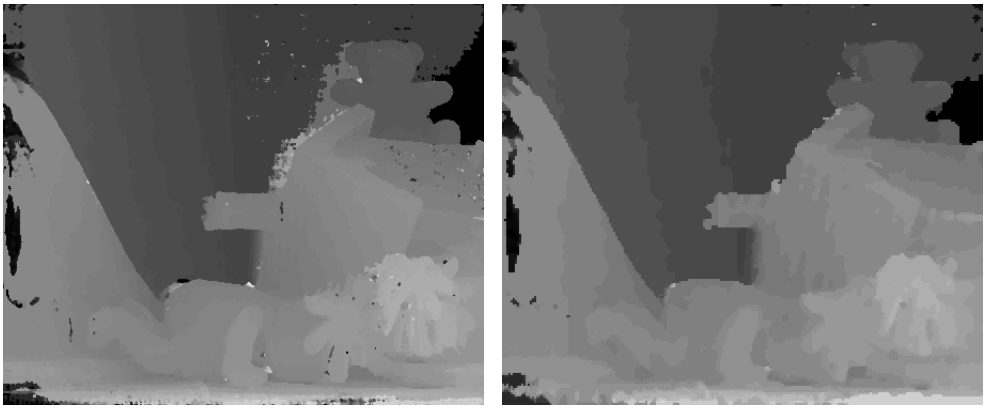
Figure 2.2 – *Methods for reducing the amount of data to process.*

In [ACN14], downsampling is proposed as a reduction of the amount of data to process in the **time** domain. The proposed AC method is illustrated on a timing synchronization problem for Wideband-Code Division Multiple Access (W-CDMA) systems. The timing synchronization is originally done with the computation of a matched filter. The input samples of the matched filter are decimated by a factor D. Consequently, one among D Multiply-accumulate (MAC) operations needed to filter the input signal are pruned. This technique reduces the overall complexity of the algorithm while keeping good performances with a dynamic adjustment of the approximation given the obtained signal-to-noise ratio (SNR).

This technique has two advantages:

- The overall computational load of the algorithm is reduced.
- Further approximation techniques can be implemented, as reducing the actual sampling rate of the system by a factor $\frac{1}{D}$ to save even more energy or increasing the factor D.

In the **spatial** domain, downscaling the data to process greatly reduces the volume of computations. For instance, it can be applied to an image processing application as the stereo matching algorithm [BDM18b]. In this case, the input signal composed of two rectified images, is fitted to a lower resolution, dividing the size of the input images by 4 to half the complexity of the studied algorithm. Nevertheless, in this case, important characteristics are lost as presented in Figures 2.3. Compared to the reference output shown in Figure 2.3(a) obtained with the input images in full resolution, the output of the stereo matching algorithm with the downsampled images 2.3(b), suffers from blurred contours. Similar techniques can be used in video processing to reduce the computational complexity.



(a) Reference output. (b) Output with downsampled input data.

Figure 2.3 – *Impact of downscaling the input data on a stereovision application: the case of stereo matching.*

2.2.2 Precision optimization

Designing an algorithm intended for embedded platforms is usually made using tools giving high precision results, as for instance double floating-point algorithms written with high-level programming languages (Matlab, Python, C/C++, ...). Nevertheless, the complexity of these algorithms limits their embeddability. Indeed, if no special care is taken during the implementation phase, the high precision of the produced results is often obtained at the expense of a high overhead in terms of latency, memory storage and energy consumption. Real-time and power consumption constraints driving the design of embedded systems increase the need to adapt algorithms before embedding them. Currently, playing with data representation has been shown to be one of the most efficient ways to efficiently trade-off quality for performance [BSM17].

Commonly, **floating-point** arithmetic is used for its simplicity of application development. Offering a high dynamic range as well as a high precision in the computations, this type of arithmetic does not need any manual conversion during computations since it is already done by the hardware. Besides, floating-point arithmetic offers the same accuracy to encode very small and very large numbers. According to the IEEE-754 standard for floating-point arithmetic [ZCA+08], floating-point numbers are represented with three parameters, namely the sign s, the exponent e, and the mantissa m as:

$$x = (-1)^s \times m \times 2^e \tag{2.1}$$

where m is a M-bit number representing the significant figures of the number x, and e an integer giving the position of the radix-point. According to IEEE-754 standard, single-precision floating-point numbers are represented with $M = 24$ and $e = 8$ and double precision floating-point numbers are represented with $M = 53$ and $e = 11$. Nevertheless, when implementing an application using floating-point arithmetic, the ease of use is to the detriment of area, latency or energy consumption. Barrois [Bar17] measured the overhead of using floating-point numbers instead of integers on a simple addition using floating-point and integers types from MentorGraphics. These results are presented in Table 2.1. The overhead, in terms of area, to implement a 32-bit floating-point addition compared to an integer addition is 3.5 times larger, while being 3.9 times larger to implement a 64-bit floating-point addition compared to an integer addition. In terms of total power required to process the addition, for 32-bit operations the overhead is 12 times larger while for 64-bit operations the overhead is 15.7 times larger. Finally, in terms of delay, the overhead on the critical path for 32-bit operations is 27.3 times larger while being 30 times larger for 64-bit operations.

	Area (μm^2)	**Total power (μW)**	**Critical path (ns)**
32-bit float ADD	653	0.439	2.42
64-bit float ADD	1453	1.120	4.02
32-bit integer ADD	189	0.037	1.06
64-bit integer ADD	373	0.071	2.10
32-bit float MPY	1543	0.894	2.09
64-bit float MPY	6464	6.560	4.70
32-bit integer MPY	2289	0.065	2.38
64-bit integer MPY	8841	0.184	4.52

Table 2.1 – *Overhead of floating-point operation compared to integer operation, extracted from [Bar17].*

When comparing floating-point and integer multiplication, the area for implementing floating-point multiplication is lower than for integer multiplication. Indeed, the multiplication is done only on the mantissa, consequently on 24 bits for single precision and 53 bits for double precision floating-point contrary to 32 or 64 bits for integers. Nevertheless, the floating-point multiplication is still more costly in terms of energy consumption.

Custom floating-point data-types have been proposed to trade accuracy for performance and can be preferred for an implementation on Field Programmable Gate Arrays (FPGAs) for instance, as presented by Mishra et al. [MS13]. In this case, the mantissa and exponent word-lengths are reduced or increased beyond the classical floating-point formats. The area and computation time can then be reduced or increased compared to a classical use of single or double precision floating-point. For instance, half-precision floating-point data-types are represented with 1 sign bit, 5 bits to encode the exponent and 10 bits to encode the mantissa [VDZ08]. Nevertheless, when customizing the word-lengths of the exponent and mantissa, a whole study has to be made to check the validity of the word-lengths of the data in the targeted application.

Representing real numbers as integers, **fixed-point** arithmetic can be considered. A number x encoded with fixed-point arithmetic is represented with three parameters, namely the sign s, the number of bits m to encode the integer part and the number of

bits n to encode the fractional part. m represents the distance expressed in number of bits between the position of the radix-point and the Most Significant Bit (MSB) while n represents the distance expressed in number of bits between the position of the radix-point and the Least Significant Bits (LSB). A number x is represented in fixed-point coding as x_Q:

$$x_Q =< x \times 2^n > \times 2^{-n} \tag{2.2}$$

where the operator $< . >$ represents the rounding mode. Despite being harder to use since fixed-point arithmetic requires determining for each variable the integer and fractionnal part word-lengths, and offering a lower dynamic range and limited precision, fixed-point Digital Signal Processors (DSPs) are cheaper than floating-point DSPs and their implementation is much faster on a General Purpose Processor (GPP). Implementing an application with fixed-point arithmetic may be useful to optimize word-lengths of the internal variables in the application, thus optimizing the dynamic power consumption. The dynamic power P_{dyn} is expressed in Equation 2.3 depending on the activity factor α, the global circuit equivalent capacitance C_g, the clock frequency and the supply voltage V_{dd}.

$$P_{dyn} \approx \alpha \cdot C_g \cdot f_{CLK} \cdot V_{dd}^2 \tag{2.3}$$

According to Equation 2.3, decreasing the activity factor which represents the fraction of the circuit that is switching, or the global circuit equivalent capacitance decreases the dynamic power and may then decrease the energy consumption of the circuit. Consequently, optimizing the word-lengths of the intern variables in an application may decrease its energy consumption.

Dynamic precision scaling is proposed in [PCR10] for applications implementing a Discrete Cosine Transform (DCT). According to the problem of massive data to process, numerous data compression techniques have been implemented, and among them DCT. The output of the targeted applications being an image, the perceived quality can be traded for the energy consumption of the algorithm. Benefiting from the fact that the high-frequency coefficients of a DCT have a lower impact on the output image quality, different input bit-width are used for the computation of the different coefficients. An algorithm is then proposed to select the best bit-width allocation to maximize the energy savings while minimizing the quality loss, offering power savings from 36 to 75%. The operand bit-width is then modified at run-time using the fact that some coefficients become negligtible after a certain number of iterations and do no more have an impact on the computation thanks to the energy compaction property of the DCT.

Clareda et al. [CGS15] implemented a modification at run-time of the word-lengths of the internal variables in an Orthogonal Frequency-Division Multiplexing (OFDM) receiver. Indeed, the required accuracy of the wireless receiver strongly depends on the channel conditions. To save energy, the bit-width is adapted depending on the channel conditions. The proposed method managed to save up to 63% of the dynamic energy consumption while offering acceptable accuracy. Nguyen et al. [NMS09] implemented dynamic precision scaling in the rake receiver of a W-CDMA receiver and managed to save between 25% and 40% of energy.

2.2.3 Less up-to-date data

The last level of action for implementing AC on the data is to work with less up-to-date data. This technique is proposed in [RSNP12] for parallel programs. The motivation of this technique is to reduce the overhead due to the synchronization between threads in

parallel programs. For instance, the authors indicate that on an IBM Power 7, the K-Means algorithm with a large input set and targetting to find 8 clusters spends 90% of the computation time in synchronization. Nevertheless, the synchronization in parallel programs is needed to ensure the following conditions:

- Data structures as linked lists are not manipulated simultaneously by several threads
- The different threads reach several points during the execution in a predictible way
- The threads work with consistent values even when working with shared variables

In the proposed method, the two last conditions are relaxed. Nevertheless, to implement this technique, a quality criterion must be predefined. Then a whole study of the considered algorithm allows the programmer choosing the relaxation points. To end with, the proposed technique allows choosing between the exact and approximate version of the code depending on the comparison of the obtained result and the quality criterion. The proposed relaxed synchronization method offers important savings in terms of computation time and thus energy consumption while guaranteeing an acceptable result.

Meng et al. [MCR09] propose using best-effort computing for parallel programming to relax dependencies between computations.

Another way to use less up-to-date data is to reduce the latency and energy of memory accesses. Miguel et al. [MBJ14] propose approximating the values inducing cache misses when accessed. In the proposed method, when a cache miss occurs, instead of accessing the higher memory levels (main memory or higher cache level), a hardware mechanism is used to estimate the accessed memory value. This mechanism is used to prevent the processor from waiting for the retrieved data. Nevertheless, load value approximation can only be used on applications with a high locality in memory values, with no cache misses.

2.3 Approximate computing techniques at the computation level

The objective when implementing an AC technique at the computation level, is to reduce the processing complexity. When reducing the processing complexity, the processing time is also reduced, which leads to a possible opportunity to go into deep sleep mode and consume less energy with dynamic power management. Two directions can be taken to reduce the processing complexity, both requiring algorithmic modifications. Computations can be skipped or sophisticated processing can be modified and approximated by simpler computations as presented in Figure 2.4. Before implementing an approximation at the computation level, the error-tolerant computations have to be identified. As explained by Nogues et al. [NMP16], the computations to approximate are chosen if they are error-tolerant as well as costly in terms of computation time or energy consumption.

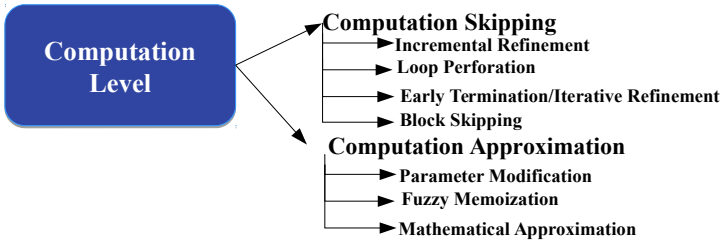


Figure 2.4 – *Approximate computing techniques: at the computation level.*

2.3.1 Computation skipping

Computation skipping consists in not executing parts of the computations to reduce the processing complexity. The selection of the skipped computations can be done at two granularity levels.

For fine-grained computation skipping, **incremental refinement** has been proposed by Ludwig et al. [LNC96] for reducing the energy consumption of digital filters. In the proposed approach, the filter order is dynamically adjusted to reduce the energy consumption which is proportional to the filter order. Ludwig et al. express the average power consumption in the implemented digital filter as:

$$P = \sum_i N_i \cdot C_i \cdot V_{dd}^2 \cdot f \tag{2.4}$$

where N_i represents the number of operations of type i performed per sample, which can be multiplications, additions, memory storages etc., C_i is the average capacitance switched per operation of type i, V_{dd} is the operating supply voltage and f the sample frequency. The authors proposed to reduce the number of operations of type i performed per sample by dynamically adjusting the filter order, which leads to a direct reduction of the average power consumption. By analyzing the input signal to filter, the higher taps in the filter can be powered-off.

Loop perforation can be used to reduce the volume of processing in an application. Sidiroglou et al. [SDMHR11] proposed to identify critical and tunable loops in an application, to reduce the executed iterations in tunable loops. In this case, a subset of the iterations of a loop is executed, which generally results in an output differing from the output of the accurate application. By applying loop perforation on several applications, the authors managed to reduce the execution time by seven while keeping the difference at the output lower than 10%. Besides, the authors have identified several computational kernels that support well perforation, as the computation of a sum, the *argmin* operation or Monte-Carlo simulation. This technique has been applied to applications from the PARSEC 1.0 benchmark suite [BKSL08] so as to cover a wide range of application domains as finance, media processing or data mining for instance.

Vassiliadis et al. [VPC+15] proposed an approximation technique similar to loop perforation but based on user information on the critical nature of parts of an application, as well as a percentage of tasks to approximate.

Loop perforation can be done dynamically, to adjust the number of iterations perforated at run-time depending on the obtained output quality or energy requirements, but the selection of the iterations to perfore induces a run-time overhead. Selective dynamic loop perforation is proposed in [LPM18], to skip a subset of the instructions in an iteration, but an overhead appears due to the selection of the iteration. In this case, loops are automatically transformed to be able to skip instructions in chosen iterations. Selective dynamic loop perforation achieves an average speedup of 2× compared to classical loop perforation technique while inducing the same amount of error.

Early termination or **iterative refinement** is a method that ends a computation process before reaching its end. Generally, the considered computation is an iterative process that converges towards a value. To save time, the iterative process is stopped before it reaches full convergence. Several stopping conditions can be set. In some algorithms, the number of iterations has a direct link with the accuracy and can be computed and fixed in advance. For instance, for the Cordic algorithm [MVJ+09], the higher the number of iterations, the more accurate the estimation of the searched cosine and sine functions will be. The stopping condition can be defined when the improvement between several

successive iterations falls below a fixed threshold. Nevertheless, to define a stopping criterion directly linked with the output quality, one may know the reference output which is generally not the case. To answer this problem, the framework ApproxIt [ZYYX14] has been proposed. It targets only iterative method and implements a quality estimator to be able to select the best approximation strategy for the next iteration at runtime.

The differences between loop perforation and iterative refinement are presented in Figure 2.5. The different iterations of the loop are figured with i_k. In the accurate implementation of the loop, the iterations from i_0 to i_n are successively executed. In the version of loop perforation LP_1, iterations of the loop are periodically skipped. In the version of loop perforation LP_2, iterations of the loop are skipped with no particular period. Finally, for incremental refinement, from a certain iteration, in the proposed example, i_2, the iterations of the loop do no more need to be executed: the loop stops.

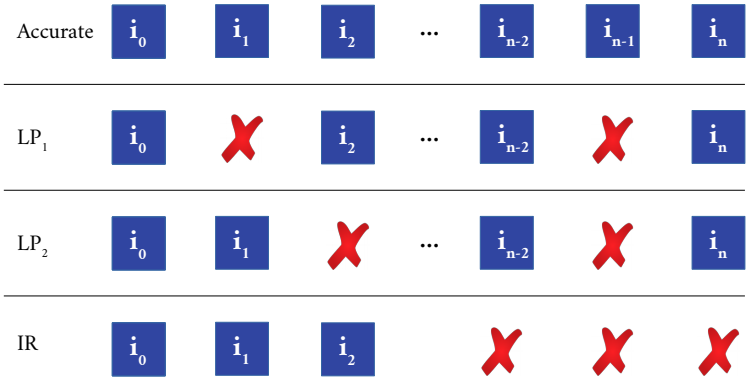


Figure 2.5 – *Different approximations on a loop: Loop Perforation (LP) and Iterative Refinement (IR).*

Finally, a coase-grained algorithmic approximation can be implemented, namely **block skipping**. In this case, a whole block is not executed to gain performance. Block skipping allows applying AC at the level of the whole application. Nogues et al. [NMP16] proposed to decompose the application to approximate, in this case, a software High Efficiency Video Coding (HEVC) decoder, into processing blocks to apply AC on chosen blocks. Once the application has been decomposed into blocks, a profiling step allows classifying the different blocks and analyzing which one are the most computationally intensive. Indeed, depending on the class of a block, it can be approximated or not. For instance, if the processing block affects the application control flow, then no approximation can be applied on this block. This block is said to produce control-oriented data. Approximable blocks are signal-oriented blocks with domain conservation. These blocks are said to produce signal-oriented data. Besides, approximation can be done mainly if such blocks preserve the signal domain. The preservation of the signal domains means that the nature of the input of the block is similar to the nature of the output of the block. In the software HEVC decoder, the most computationally intensive blocks are the motion compensation filters. To reduce the energy consumption of the decoder, block skipping is implemented. To trade-off the energy consumption and the application quality of service (QoS), a parameter called *skip control* defines the frequency of block skipping. According to the skip control parameter, the filters are activated (or not). The frequency of block skipping is set as a percentage of pixel blocks where filters are skipped. The chosen blocks can then be skipped permanently or regularly. Block skipping in the HEVC decoder allows saving up to 40% of the energy consumption.

2.3.2 Computation approximation

When implementing computation approximation, sophisticated processing blocks are replaced by less complex ones. To guarantee that the processing by both blocks will lead to a similar result, mathematical equivalence between both blocks is required. For instance, replacing a division by a constant by the multiplication by the inverse of the constant are two blocks mathematically equivalent. Nevertheless, mathematical equivalence can be relaxed to achieve further complexity reduction.

Parameter modification can be applied to reduce the computational complexity in blocks such as filters or transforms for instance. In this case, the algorithm is preserved but a parameter is modified to reduce the volume of computations. For instance, a filter with similar specifications may be designed for different filter orders, thus affecting the number of computations in the filter. The parameter modification induces a modification of the algorithm. Nogues et al. [NMP16] proposed to tune the number of taps of the motion compensation filters according to a parameter called the *approximation level control*.

Fuzzy memoization can be used. Memoization is the principle of saving the result of the execution of a function or computation in memory, so as to use it for future executions. This technique has first been proposed to reduce the execution time for a computation already done. The main motivation to implement memoization techniques is to remove redundancy due to the repetition of instruction-level inputs, as well as repetition of input data within high-level computations. The principle of memoization is particularly useful for costly operations as multiplications or divisions or for complex processing composed of several elementary operations. When an instruction has to be executed, for instance, x Op $y = z$, the input operands x and y are used to access the Look-Up Table (LUT) storing the previously executed instructions, called the reuse table. If the operation has already been executed, this is a hit, and the result is extracted from the reuse table. If this is not the case, this is a miss. The instruction is executed and its result is stored in the reuse table. The principle of memoization is presented in Figure 2.6. As Arjun et al. explained in [SSRS15], to apply memoization in a program, two conditions have to be satisfied:

- The memoized code has to be transparent to the rest of the code, that is to say that it should not cause any side-effect.
- For the same input, the memoization and original code sould produce identical output.

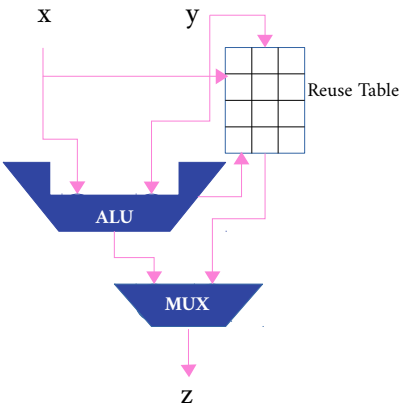


Figure 2.6 – *Illustration of the principle of memoization.*

According to these two conditions, no approximation lies in memoization. Besides, the gain brought by memoization may be annihilated by the size of the Look-Up Tables (LUTs) to store in memory to save the past execution results. To further improve the performance in terms of execution time using value locality, **fuzzy memoization** has been proposed and applied for floating-point operations by Alvarez et al. [ACV05]. Indeed, floating-point numbers offer a high dynamic range and imply the need for large reuse tables to achieve an acceptable hit rate when accessing the table. Contrary to classical memoization, when implementing fuzzy memoization, before accessing the reuse table presented in Figure 2.6, N Least Significant Bits (LSBs) of the input operands x and y are dropped. A masking operation is applied before accessing the reuse table, which implies that operands with similar Most Significant Bits (MSBs), despite being strictly different, will be affected to the same compartment of the reuse table. Fuzzy memoization is particularly well tolerated in multimedia applications, where the end-user generally tolerates errors. The number N of dropped bits can be used to trade-off the output quality and the computation time or energy consumption of the targeted application.

Mathematical approximation has been proposed for integrating complex applications composed of numerous sophisticated processing in embedded systems. In this context, mathematical functions are commonly used in embedded applications in various domains like digital communications, digital signal processing, computer vision, robotics, etc... When determining the value of an intricate function, the exact value or an approximation can be computed. The challenge is to implement these functions with enough accuracy without sacrificing the performances of the application, namely memory usage, execution time and energy consumption. Several solutions can be used to compute an approximate value of a mathematical function f over a segment I, according to a maximum error value ϵ.

Specific algorithms can be adapted to a particular function [PTVF88]. Several methods can be used to compute an approximate value of a function. Iterative methods as the shift-and-add BKM algorithm [BKM94] or the COordinate Rotation DIgital Computer (CORDIC) algorithm [MVJ+09] are generally easy to implement. For instance, the CORDIC algorithm computes approximate values of trigonometric, logarithmic or hyperbolic functions and is used in calculators. To compute the tangency of an angle θ as presented in Figure 2.7, the principle of the algorithm is to apply successive rotations to a vector $\vec{v}$ whose initial coordinates are $(1, 0)$ and final coordinates (X, Y). Indeed, to rotate a vector whose coordinates are (x_{in}, y_{in}) from an angle θ, the operation applied to compute the coordinates of the resulting vector is:

$$\begin{bmatrix} x_o \\ y_o \end{bmatrix} = \cos\theta \begin{bmatrix} 1 & -\tan\theta \\ \tan\theta & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{in} \\ y_{in} \end{bmatrix} \tag{2.5}$$

Nevertheless, to obtain an efficient implementation of the CORDIC algorithm on low cost hardware, the multiplications have to be avoided. To do so, instead of applying a single rotation of θ, several rotations of small angles θ_i are applied, such that $\theta \simeq \sum_{i=0}^{n} \theta_i$. Besides, in the efficient hardware implementation of the CORDIC algorithm, the values of $\tan\theta_i$ are taken equal to 2^{-i} to replace the multiply operations by shifts. Finally, the obtained values of x_o and y_o are equal to $\cos\theta$ and $\sin\theta$ respectively.

The accuracy of the CORDIC algorithm is strongly dependent on the number of iterations, that is to say n, used to compute the rotation. For instance, to compute the cosine and sine of angle 70°, the values obtained with 5 iterations of the CORDIC algorithm are 0.3706 and 0.9280 respectively, while with 10 iterations, the values obtained are 0.3435 and 0.9390 respectively. Nevertheless, nowadays, a growing majority of embedded pro-

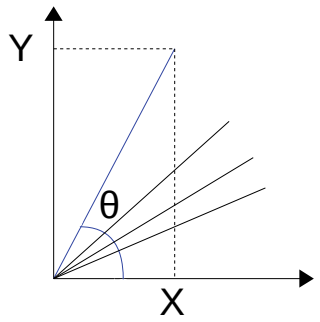


Figure 2.7 – *Illustration of the CORDIC algorithm, θ represented in blue.*

cessors possesses a hardware multiplier. Besides, the computation time of the CORDIC algorithm strongly depends on the targeted accuracy that sets the number of iterations. It is also possible to compute the approximation of a function thanks to other methods whose algorithms are described in [PTVF88].

LUT or bi-/multi-partite tables methods can be used when low-precision is required. Nevertheless, the need to store the characteristics of the approximation in tables results in some impossibilities to include these methods in embedded systems because of their memory footprint. Indeed, LUT-based method would consume the most memory space but would be the most efficient in terms of computation time. That method consists in using 0-degree polynomials to approximate the value of the function f. The interval I on which the function has to be evaluated is segmented until the error between each 0-degree polynomial and the real value of f is lower than the maximal acceptable error value ϵ on each segment. However, this type of segmentation has to be uniform: if the error criterion is not fulfilled on a single subsegment J, all the subsegments of I have to be segmented again. De Dinechin and Tisserand [DDT05] detailed improvements of the LUT method called bi-/multi-partite methods. The multi-partite method proposes to segment I to approximate f by a sequence of linear functions. The initial values of each segment as well as the values of the offsets to add to these initial values to get whichever value in a segment have to be saved in tables. The size of these tables is then reduced compared to bi-partite tables methods exploiting symmetry on each segment. That method allows quick computations and reduced tables to store but is limited to low-precision approximation. This method is efficient for hardware implementation.

Finally, polynomial approximation is a good alternative for function evaluation, especially when several elementary functions are combined. Tools like Sollya [CJL10] provide the polynomial coefficients to approximate a function f on an interval I for a predefined polynomial order. For embedded fixed-point processors, polynomial approximation can give very accurate results in a few cycles if the interval I is segmented finely enough. That is to say, a polynomial P_i approximates the function f on each segment of I. The segmentation is required so as to approximate the function f according to a maximum error of approximation $\epsilon_{\max}$. The polynomial order is then a trade-off between the approximation error and the segment size. To obtain a given maximum approximation error, the decrease in the polynomial order implies the reduction of the segment size. This increases the number of polynomials to store in memory. For a given data-path word-length, the increase in polynomial order raises the fixed-point computation errors and annihilates the benefit of lower approximation error obtained by a too high polynomial order. Thus, for fixed-point arithmetic, the polynomial order is relatively low. Consequently to obtain a low maximal approximation error, the segment size is reduced. Accordingly, non-uniform segmentation

is required to limit the number of polynomials to store in memory. Thus, the challenge is to find the accurate segmentation of the interval I.

Lee et al. have proposed different non-uniform segmentations [LCLV09] for hardware function evaluation. On each segment, the function f is approximated by the Remez algorithm. Afterwards, a simple logic circuit is used to find the segment corresponding to an input value x. LUTs are used to store the coefficients of the polynomials. For software function evaluation, Bonnot et al. [BNM16] proposed a non-uniform segmentation technique. The first step of this method consists in finding the optimal non-uniform segmentation for approximating the function f on the interval I. The non-uniform segmentation is then stored in a tree structure as presented in Figure 2.8. Each node of the tree structure represents a segment on which the accuracy contraint is not respected, while the leaves are segments where the accuracy constraint $\epsilon_{\max}$ is fullfilled. Consequently, an approximating polynomial P_i is associated to each leaf. The coefficients of the different polynomials as well as required shifts to compute the value $P_i(x) \simeq f(x)$ with input $x \in \mathcal{I}$ are stored in tables.

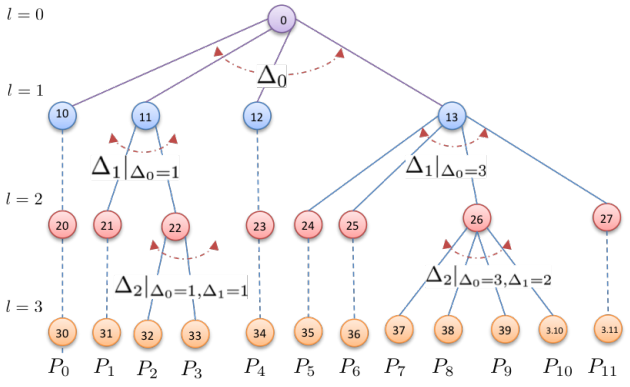


Figure 2.8 – *Non-uniform segmentation of I stored in tree structure $\mathcal{T}$.*

The challenge of the proposed method, is to efficiently access the approximating polynomial P_i depending on the input value x. To be efficient, if the input x is formatted in fixed-point with m bits to encode the integer part, the m MSB of the input x are used to know in which segment is included value x, and consequently which polynomial P_i is used to compute the approximate value of $f(x)$ as presented in Figure 2.9. In these methods, the maximum error of approximation is used as an user-defined parameter, and has an impact on the memory footprint of the system as well as computation time.

2.4 Approximate computing techniques at the hardware level

Approximations can be done on the hardware structure of a circuit: the implementation of the circuit itself or its functionning parameters can be modified. Firstly, inexact arithmetic operators are considered. They are then called approximate hardware module. The errors induced by the implementation of approximate hardware modules can then be tackled by an error correction circuit or by dedicated software, or left untackled in case of an error-resilient application. The use of an inexact operator is considered as a functional approximation: the truth table of the function is modified to tolerate errors in the name of reducing the logic complexity and the length of critical paths. The effect on the circuit is a reduction in power consumption and area. Other modifications on the hardware level are possible, and may be applied along with the modifications on arithmetic operators. These

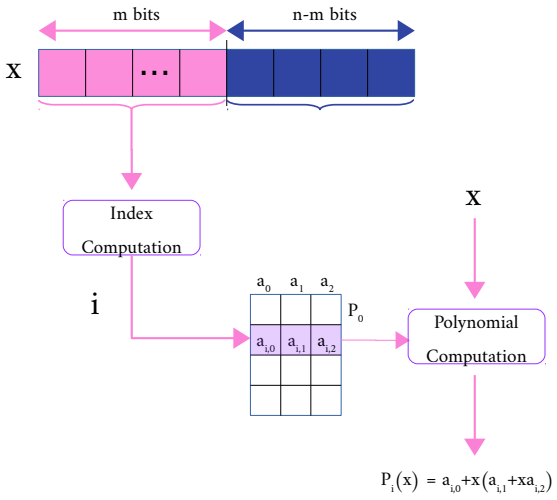


Figure 2.9 – *Evaluation of polynomial $P_i(x)$.*

modifications are done on the circuit functioning parameters and reduce the dynamic power dissipation inducing errors. Finally, unreliable modules as unreliable memory storage, have been proposed. The different AC techniques on the hardware structure are presented in Figure 2.10.

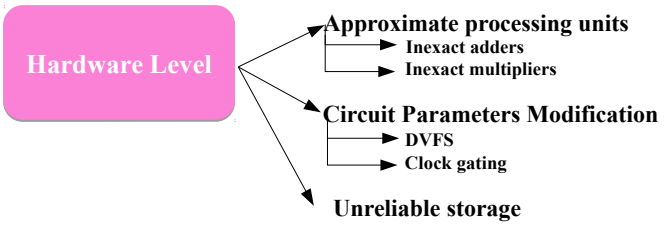


Figure 2.10 – *Approximate computing techniques: at the hardware level.*

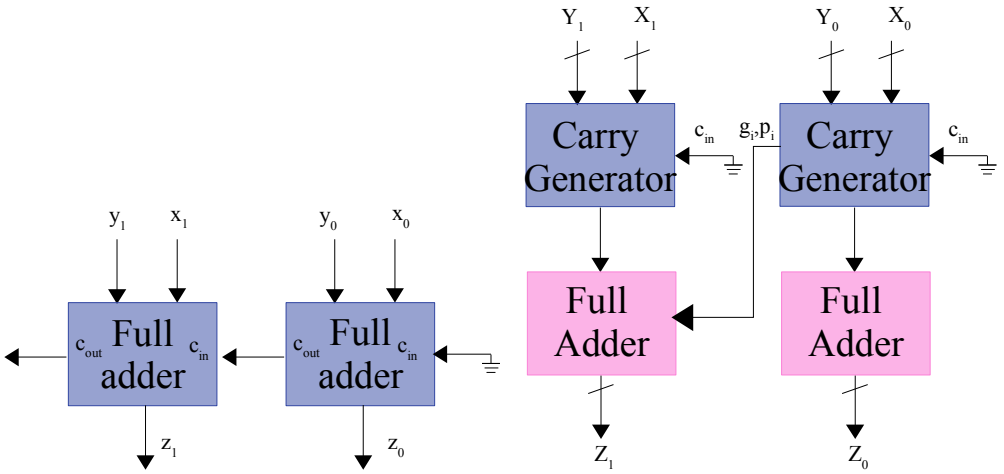
2.4.1 Approximate processing units

Inexact adders

The current arithmetic operators having reached their performance limitations, approximate hardware modules have been created. Whether it be in terms of critical path or circuit area, to get better performances, the boolean function of operators has to be slightly modified to overcome these limitations. Consequently, the output of an inexact operation is not always mathematically equal to the output of the accurate operator.

Accurate adder structure A lot of work force has been put on classical arithmetic operators as adders since they are generally the basic blocks of numerous more complex circuits. The best obtained performances with accurate adders are a logarithmic delay, for an N-bit adder, the critical path is equal to $O(log(N))$, and a linear area. These performances have been obtained with adders as the Ripple-Carry-Adder (RCA) presented in Figure 2.11(a) and the Carry-Look-Ahead Adder (CLA) presented in Figure 2.11(b).

The accurate adder RCA is composed of full adders taking as inputs 2 bits, x_i and y_i as well as the carry signal c_{in} from the previous stage of the addition. Each full adder of the RCA outputs the resulting sum bit z_i as well as the carry out signal c_{out}. The truth



(a) Ripple-Carry Adder. (b) Carry-Look-Ahead Adder.

Figure 2.11 – *Design of two accurate adders.*

table of the full adder is represented in Table 2.2. The area required to implement an N-bit RCA is equivalent to $O(N)$ with a delay equivalent to $O(N)$.

x_i	y_i	c_{in}	z_i	c_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 2.2 – *Full adder truth table.*

The accurate adder CLA is composed of full adders that compute the output sum bit z_i as well as signals p_i, propagate the carry, and g_i, generate the carry, that are sent to an independent circuit that computes the carry c_{in} for the next stage of the adder. At each stage of the addition, the signals p_i, g_i, c_i and z_i are computed as:

$$p_i = x_i \oplus y_i \tag{2.6}$$
$$g_i = x_i \wedge y_i \tag{2.7}$$
$$c_i = g_{i-1} \vee (c_{i-1} \wedge p_{i-1}) \tag{2.8}$$
$$z_i = p_i \oplus c_i \tag{2.9}$$

Using the carry look-ahead circuit, the carry generation is faster, which induces a delay reduction equivalent to $O(log(N))$. Nevertheless, the required additionnal circuit leads to an important increase in the global circuit area.

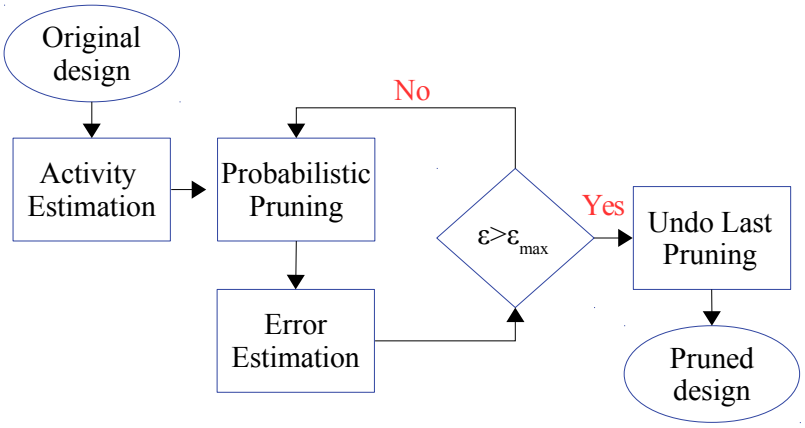


Figure 2.12 – *Optimization process for probabilistic pruning of a circuit.*

To improve the performances of adders, accurate adders can automatically be modified into inexact circuits or new designs can be proposed from scratch.

Probabilistic pruning [LEN+11] is a design technique that relies on architectural modifications of an original circuit by automatically pruning portions of the circuit to reduce its power consumption, depending on their probability of being active during the operation. To choose the pruned portion of the circuit, error metrics are considered. The circuit is modeled as a Directed Acyclic Graph (DAG) $\mathcal{C}$ with the circuit components being the nodes of the DAG and wires being the edges. To find the pruned circuit under an error constraint, an optimization problem has to be solved. The DAG of the pruned circuit $\mathcal{C}'$ is a sub-graph of $\mathcal{C}$, with similar set of inputs/outputs but a minimized number of components. The error metric considered by Lingamneni et al. [LEN+11] is the average error amplitude. The optimization process is presented in Figure 2.12.

The first step when applying probabilistic pruning to a circuit is to estimate the activity of each node in the circuit. The goal is to prune the circuit parts with highest activity. Then, before applying probabilistic pruning to the circuit, two strategies can be adopted:

- Weighted Probabilistic Pruning, where the choice of the pruned portions of the circuits is made according to a cost function assigned to each node, which is the activity of the node multiplied by its significance in terms of induced error.
- Uniform Probabilistic Pruning, where all the portions of the circuits have the same cost for pruning.

Lingamneni et al. [LEN+11] applied probabilistic pruning to several classical adders, as for instance RCA or CLA. The obtained results for uniform probabilistic pruning leads to savings in energy-delay-area product between 2 and 7.5× with error rates comprised between 10^{-6}% and 10%. When it comes to weighted probabilistic pruning, the savings are up to 5.3× for corresponding error rate of 37%.

The majority of the proposed **inexact adders** in the literature, are designs trying to reduce the length of critical carry chain path. For an exact addition of x and y encoded on N-bits, the carry is propagated from the lowest full addition on the LSBs x_0 and y_0, up to the upper full addition on the MSBs as presented in Figure 2.11. Nevertheless, the probability for the carry to be propagated up to the higher full adder is very low. Indeed, the carry signal c_i at stage i depends on the previous carry c_{i-1} only if the signal propagate p_i at stage i is true. According to Verma et al. [VBI08], the knowledge of the longest chain

of propagated signals which corresponds to the number L such that from stage i to $i+L$, the signals p_i are equal to 1, indicates that c_i is independent from c_{i-L-1}. It is then possible to build adders of size L in parallel without propagating the carry between them.

Verma et al. [VBI08] studied the longest sequence of propagate signals by studying the longest run of heads in a series of n coin tosses, and have identified the behavior of a coin to the behavior of a bit. The longest sequences of propagates according to a given probability p have been computed and are reported in Table 2.3. According to Table 2.3, the percentage of chances that in a 64-bit addition, the carry c_i will be independent from carry c_{i-18} is equal to 99.99%. The propagation of the carry signals from the LSBs up to the MSBs can be used to trade-off the delay of the addition, since the longest sequence of propagates is equivalent to $\log n$ where n is the adder input bit-width.

Bitwidth	**p = 99%**	**p = 99.99%**
64	11	17
128	12	18
256	13	20

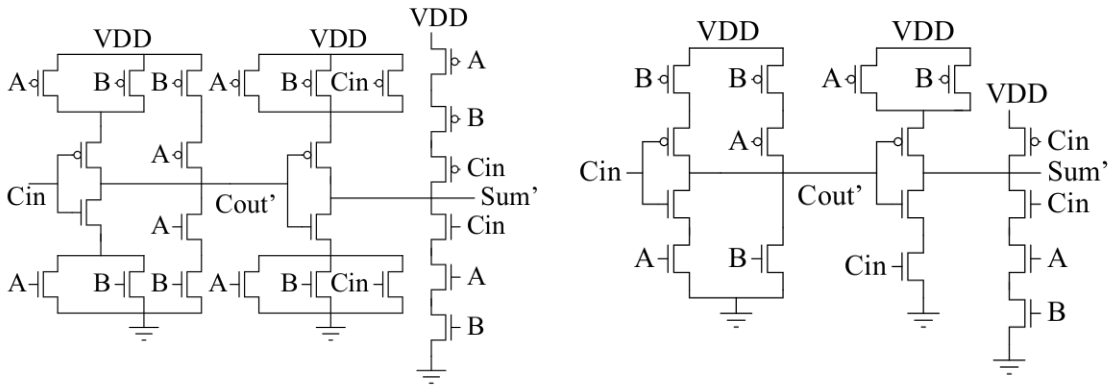
Table 2.3 – *Longest sequence of propagates for a given probability p.*

To reduce the delay of adders, four major kinds of topology of **inexact adders** have been explored in the literature: segmented adders, speculative adders, and carry cut-back adders.

Segmented adders are processing differently the MSBs, processed by accurate adders, and the LSBs processed by inexact adders. The inexact adder is segmented into an accurate and an approximate part.

Approximate Full Adders (AFAs) are for instance approximating the computations on the LSBs by implementing approximate full adders. In this case, the boolean function of the full adder is modified. Gupta et al. [GMP+11] proposed to reduce the logic complexity of an adder at the transistor level. The considered accurate full-adder is the Mirror Adder (MA) represented in Figure 2.13(a). The accurate MA is composed of symmetrical NMOS and PMOS chains which account for 24 transistors. The approximate designs built on the structure of the MA reduce the number of transistors as well as the internal node capacitance. The reduced size of the adder area induces a reduction in the switching capacitance αC_g, having an impact on the dynamic power P_{dyn} as presented in Equation 2.3. Besides, the reduction in the number of transistors is equivalent to a reduced complexity and consequently shorter critical paths. Gupta et al. proposed 3 inexact versions of the 1-bit MA, used to build larger bit-width inexact adders. The inexact adders are only used for processing the LSBs to ensure high output quality. An example of approximation on the structure of the MA is presented in Figure 2.13(b).

To obtain the inexact version of the MA, the authors have removed transistors one by one from the accurate version of the MA. The goal when designing inexact versions of this operator, was to ensure the highest possible number of accurate outputs Sum and C_{out}. In the proposed inexact structure, the number of transistors has been reduced from 24 to 16. The removed transistors correspond to the part of the circuit used to compute Sum since $Sum = \overline{C_{out}}$ in 6 cases out of 8. An error is then induced in one case on C_{out} and 3 cases on Sum, as presented in Table 2.4. The area of the accurate MA amounts to 40.66 μm^2 while the area of the simplified MA is reduced to 22.56 μm^2.



(a) Accurate mirror adder.

(b) Simplified mirror adder.

Figure 2.13 – *Structure of the mirror adder and a simplified version of the mirror adder, extracted from [GMP+11].*

A	B	C_{in}	Sum	C_{out}	Sum_{app}	$C_{out,app}$
0	0	0	0	0	1	0
0	0	1	1	0	1	0
0	1	0	1	0	0	1
0	1	1	0	1	0	1
1	0	0	1	0	1	0
1	0	1	0	1	0	1
1	1	0	0	1	0	1
1	1	1	1	1	0	1

Table 2.4 – *Mirror adder truth table: accurate and simplified version.*

Gupta et al. used the inexact version of the MA to build AFAs, by approximating only the LSBs. The IMPrecise Adder for low-power Approximate CompuTing (IMPACT) has for instance been proposed, by implementing inexact MA on the LSBs while processing the MSBs with a RCA. The number of approximated LSBs can then be used to tune the degree of approximation.

Mahdiani et al. [MAFL10] proposed another type of segmented adder: the bio-inspired inexact Lower-part-OR adder (LOA). In this case, the LSBs of the input operands are processed with an OR gate and the carry-in signals are generated with an AND gate. The MSBs are processed with an accurate adder. In the LOA, a p-bit addition is divided into two smaller additions, an m-bit accurate addition on the MSBs and a n-bit inexact addition on the LSBs such that $m + n = p$. The LOA is presented in Figure 2.14. The accurate addition on the MSBs is materialized by the Accurate adder, and outputs the sum on the MSBs $S_{p-1:n}$ as well as the carry-out signal C_{out}. The accurate adder is separated from the addition on the LSBs and the carry-in signal to provide to the accurate adder is computed using an AND gate on the input bits $n-1$, X_{n-1} and Y_{n-1}. Finally, the sum on the LSBs is processed using bitwise OR gates.

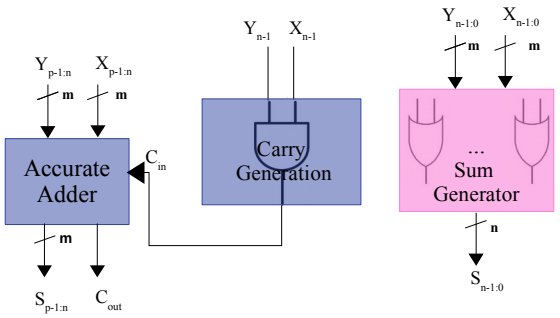


Figure 2.14 – *Structure of the Lower-part-OR adder.*

The proposed LOA frequently generates an error but the mean amplitude of the generated error stays moderate. As for the Approximate Full Adder (AFA), the frequency of errors can be adjusted with the length of inexact lower part. For instance, if the approximation is done on 2 LSBs, the frequency of errors amounts to 40% compared to an accurate adder of the same size. If the approximation is done on 8 LSBs, the frequency of errors reaches 90%. Nevertheless, the maximum error amplitude generated stays moderate since equal to $2^{n-1} - 1$ where n is the length of inexact lower part. In terms of area and delay, when implementing a LOA with an accurate adder implemented with a RCA, when the size of the inexact lower part increases by 1 bit, the number of gates decreases between 21 and 31. Besides, for a similar implementation, the delay is reduced between 0.16ns and 0.55ns.

Another type of segmented adders is composed of the different variations of Error-Tolerant Adders (ETAs). According to the Error-Tolerant Adder (ETA) proposed by Zhu et al. [ZGZ+10], the addition on the MSBs is performed as a classical addition, from the LSB of the MSBs, up to the MSB, as presented in Figure 2.15. The processing on the LSBs is different: no carry signal is generated and the processing is done from the MSB of the LSBs up to the LSB. For each stage of the addition on the LSBs, since the carry signal is eliminated, a special processing is done to minimize the induced error. The input bits at each stage of the addition are scanned from left to right. While one of the input bits x_i, y_i is different from 1, a classical addition is performed, from the left to the right. If the input bits x_i, y_i are both equal to 1, the scanning process stops and from the output bit z_i, all the output bits are set to 1.

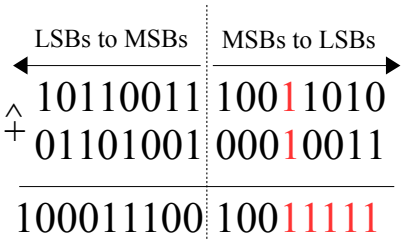


Figure 2.15 – *Structure of the ETA.*

The goal of the ETA is to reduce the delay as well as the energy consumption. Instead of waiting for the carry signals to compute the next stage of addition, the addition on the MSBs and the addition on the LSBs are performed in parallel. Nevertheless, the mechanism to approximate the carry propagation phenomenon may lead to frequent errors. To reduce the error frequency, variations of the ETA have been proposed.

Contrary to the ETA, the Error-Tolerant Adder Type II (ETAII) [ZGY09] no longer eliminates the carry propagation in the addition on the LSBs but aims at reducing the delay of propagating the carry. The ETAII combines the structure of a segmented adder since a n-bit adder is segmented in to $k < n$-bit sub-adders, with the speculative structure of the Almost Correct Adder (ACA). Indeed, Zhu et al. studied the probability that the carry signal is correct depending on the number of bits taken into account to compute the carry signal. Using only one bit to speculate the carry signal, it has 75% of chances to be correct. Adding 3 bits for the speculation increases this probability to 96%. A general formula has been derived to indicate the probability P of computing an accurate carry signal depending on the number of neighbouring bits k taken as inputs to the CLA, $P = 1 - 0.5^k$. The basic structure of the ETAII is presented in Figure 2.16. Each $\frac{n}{k}$-bit sub-adder is composed of two blocks, a carry generator and a sum generator. The adder is then regularly segmented in sub-adders. The carry is generated by CLA blocks that feed the sum generator to the next stage with the generated carry signal to reduce the induced error. In the ETAII, the longest possible carry propagation chain length is equal to $\frac{2n}{k}$. The complexity of designing the ETAII adder lies in the choice of the sub-adders length, which allows trading-off the quality for energy consumption. To evaluate the accuracy of the designed ETAII depending on the length of sub-adders, Zhu et al. have simulated bit-accurate C code reproducing the behavior of the adder for varied k, $n = 32$ and random inputs drawn in $[0; 2^{32} - 1]$. In the proposed experiments, the probability that the accuracy of the output result is higher than 95% is equal to 83.5% for $k = 2$, and reaches 98.3% for $k = 4$ and 100% if the size of the sub-adders is 16-bits. Nevertheless, the obtained error measurements have been obtained with only 10000 inputs, which is not significant on the overall input set. Besides, one of the main drawback of the ETAII is that the same number of bits are used to compute the carry signal whether it applies on the LSBs or on the MSBs. Another variation of the ETA has been proposed as the Error-Tolerant Adder Type IV (ETAIV) [ZGWY10]. The ETAIV is segmented into $\frac{N}{X}$ blocks and processes differently the LSBs and the MSBs. The carry signal on the LSBs is processed as in the ETAII, while on the MSBs two parallel blocks are used to infer the carry signal:

- A carry generator with a carry in signal $c_{in} = 0$
- A carry generator with a carry in signal $c_{in} = 1$

The obtained carry out signals are then sent to a 2-to-1 multiplexer controlled by the carry out generated from the LSBs to select the carry out signal. The induced errors in the ETAIV can then be controlled with the size of the blocks in the segmented adder. The error maps of 8-bit ETAIV are presented in Figure 2.17 for two values of X.

Zhu et al. [ZGY09] provided a comparison of different 16-bit accurate and inexact adders in terms of performance, which is presented in Table 2.5. The presented results have been obtained with an input frequency of $100MHz$ and the power is an average on 100 randomly drawn inputs. The indicated delay is the one obtained in worst-case. The most power-efficient adder is the ETA while the fastest is the accurate RCA. Nevertheless, to draw a fair comparison between the proposed adders, a fair error measurement has to be done.

Speculative adders take the risk of breaking the carry chain, as for instance the ACA and Variable Latency Speculative Adder (VLSA) presented in [VBI08]. The ACA is the most known speculative adder. Exponentially faster than traditionnal accurate adders, it is composed of an array of overlapping and translated sub-adders, so that each sum bit is constructed using exactly the same amount of preceding carry stages, except the first

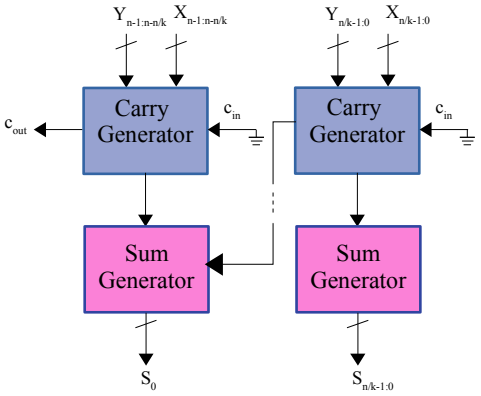
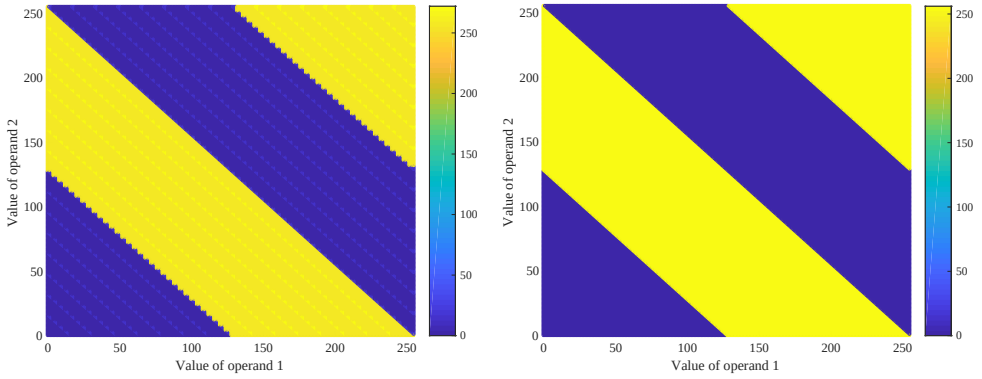


Figure 2.16 – *Structure of the ETAII.*

	Adder	Power(mW)	Delay (ns)	# of Transistors
Accurate	**RCA**	0.22	4.04	896
	CLA	0.51	2.37	2208
Approximate	**ETA**	0.13	2.29	1006
	ETAII	0.24	0.20	1372
	ETAIV	0.25	1.03	1444

Table 2.5 – *Performances of the ETA and its variations compared to accurate adders, extracted from [ZGY09].*



(a) ETAIV N=8,X=2. (b) ETAIV N=8,X=4.

Figure 2.17 – *Error maps of the ETAIV for different segmentation sizes, color scales correspond to the error.*

ones. The critical-path delay is limited and near-linear, but the circuit cost is fairly high. The area of the ACA is slightly larger than the area of a RCA. The ACA, represented in Figure 2.18, is an interesting case study due to its very low Error Rate (ER). Errors occur when carry chains are longer than the ACA sub-adder size C, which is the main ACA design parameter. The delay of the ACA is equal to the delay of the sub-adder of size C. C embodies a trade-off between speed and accuracy, as illustrated in Figure 2.19. The values of the induced errors are represented in Figure 2.19(a) for a 8-bit ACA with $C = 2$ and in Figure 2.19(b) for a 8-bit ACA with $C = 6$. The operand combinations that do not generate any error are indicated in dark blue. Thus, ACA designs have a very low

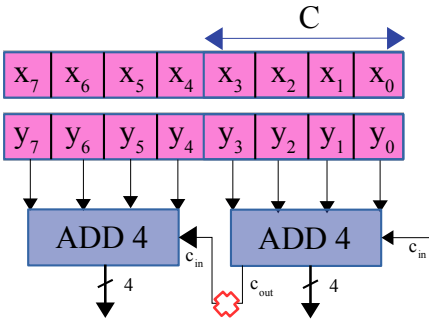
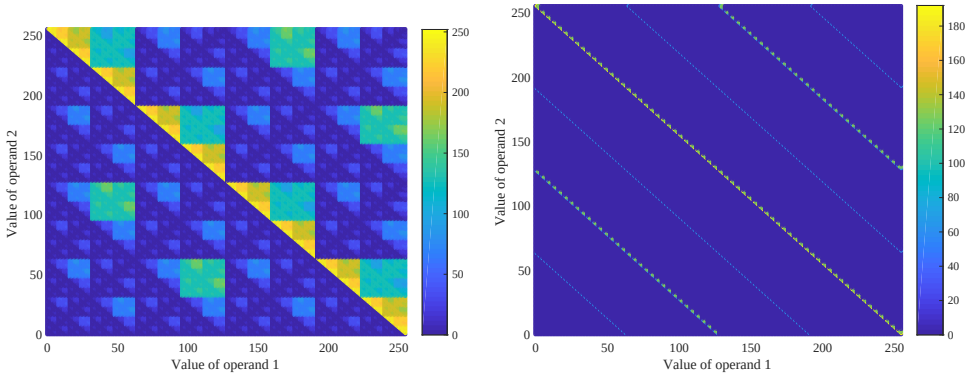


Figure 2.18 – *Structure of the ACA.*

frequency of errors, but of high arithmetic distance. For instance, for the 8-bit ACA with $C = 6$, the error rate is equal to 0.0156, the average error amplitude to 1.75 and maximum error amplitude 192. Verma et al. have proposed the VLSA based on the design of the ACA to detect when errors occur and correct them. The VLSA consists in adding an error detection circuit to the ACA. The VLSA is then composed of an ACA to which is added a signal indicating whether the result is accurate or not. Since the error rate of the ACA is relatively low, the delay of the VLSA is almost equal of the delay of the ACA. The signal to indicate whether the result is accurate or not is computed by analyzing the propagate signals. The propagate signals are computed with a AND gate on both input bits. The goal of computing this signal, is to indicate if a carry chain is longer than C. The implemented error correction circuit is composed of an $\frac{N}{C}$ CLA block, and outputs the carry signal missed because of the carry chain longer than C. Finally, when the output of the adder is not correct, an error correction circuit is used. The VLSA is then always correct but has a variable latency.



(a) ACA N=8,C=2. (b) ACA N=8,C=6.

Figure 2.19 – *Error maps of the ACA for different carry-chain lengths, color scales correspond to the error.*

The Inexact Speculative Adder (ISA) [CSE15] is an optimized version of the architecture of speculative adders. As an evolution of the ETAII, it also segments the addition into several sub-adders with carry speculated from preceding sub-blocks. The ISA features a shorter speculative overhead that improves speed and energy efficiency, and introduces a dual-direction error correction-reduction scheme that lowers the mean and the worst-case errors. The structure of the ISA is presented in Figure 2.20. The adder is segmented into

sub-adders executed in parallel, themselves composed of 3 blocks. A first block speculates the value of the carry, depicted by Carry Speculation block, and feeds a sub-adder, depicted by Sum Generator block, to generate a temporary sum value. The obtained value as well as the carry speculated are feeding a compensation block, depicted by Compensation block, to correct the induced errors. The compensation block compares the carry signal generated by the Carry Speculation block with the carry signal from the previous Sum Generator block. If those two values are inconsistent, a correction on the sum is applied. If this correction cannot be applied, the error magnitude is reduced using the preceding sum bits.

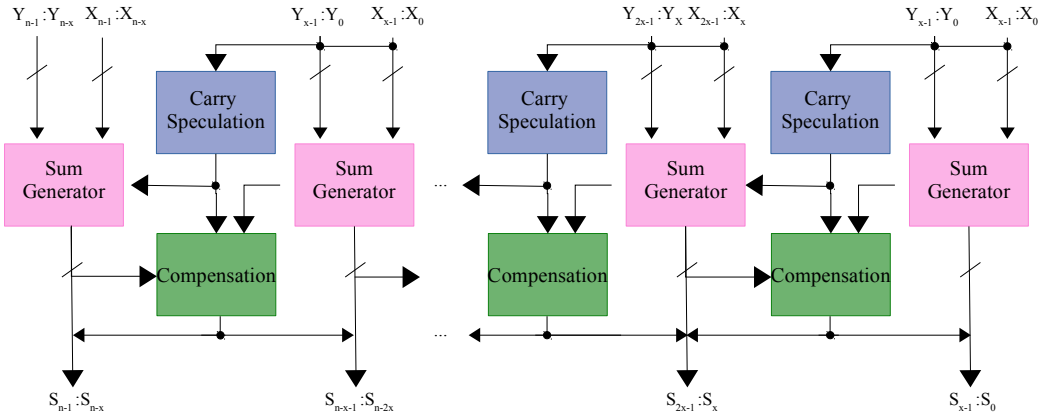


Figure 2.20 – *Structure of the ISA.*

The ISA design typically yields higher Error Rates than ACA but with lower error values, depending of the number of sub-blocks and error compensation level, which are the main ISA design parameters. Nevertheless, the multiplexers required for a good error compensation still represent a substantial area and energy overhead, particularly for low-speed implementations.

Finally, a last type of inexact adders is the Carry-Cut Back Adder (CCBA) [CSE16]. The CCBA exploits a novel idea of artificially-built false paths (i.e. paths that cannot be logically activated), co-optimizing arithmetic precision together with physical netlist delay. The CCBA is segmented into several sub-adders, and multiplexers are inserted between sub-adders to cut the carry propagation, so as to reduce the critical path delay. The carry chain is cut depending on a Propagation block analyzing whether the carry signal has to be propagated or not. Carry Speculation blocks can be used optionnally. To guarantee floating-point-like precision, high-significance carry stages are monitored to cut the carry chain at lower-significance positions. These cuts prevent the critical-path activation, thus relaxing timing constraints and enabling energy efficiency levels out of reach from conventionally designed circuits. The ER ranges similarly as for the ISA, but the error values are lower than those generated by the ACA and the ISA, depending of the number of cuts and cutting distance, which are the main CCBA design parameters.

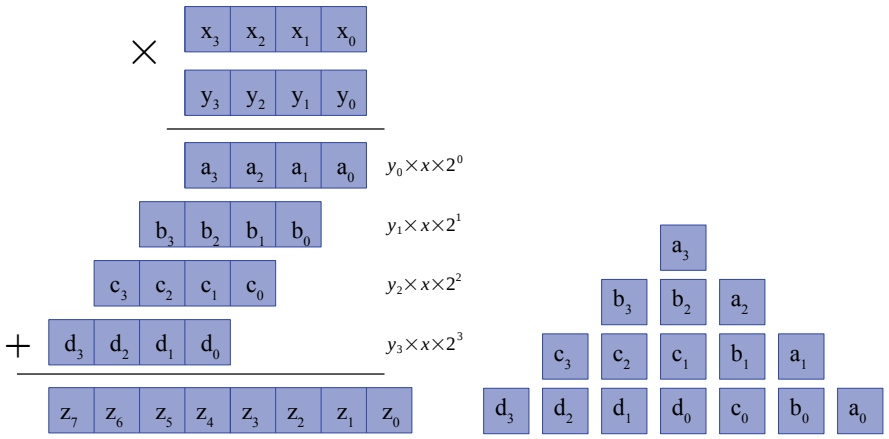
Approximate multipliers

Accurate multiplier structure An integer multiplication of two N-bit integers can be decomposed into two steps:

- Generation of the partial products

- Addition of the partial products

The output of the multiplication is on $2N$ bits. The principle of integer multiplication is depicted in Figure 2.21 with 4-bit integers $[x_3 : x_0]$ and $[y_3 : y_0]$. The generated partial products are represented with the integers $[a_3 : a_0], [b_3 : b_0], [c_3 : c_0]$ and $[d_3 : d_0]$. When computing a binary multiplication, the output of a partial product can be 0 or the multiplicand. Consequently, the most greedy part of the multiplication in terms of resources is the addition of the partial products.



(a) Generation of partial products and addition.

(b) Partial products addition: representation of the partial product tree.

Figure 2.21 – *Multiplication of two integers on* 4 *bits.*

As shown in Figure 2.21(b), the partial product addition can be represented as a triangle, called the partial product tree. To build more efficient multipliers, the reduction of the partial product tree can be targeted. To reduce the partial product tree, Full Adder (FA) and Half Adder (HA) cells are cascaded until 2 lines are left in the tree. FA cells take 3 inputs, for instance bits b_2, c_1 and d_0 in Figure 2.21(b) and compress them into 2 bits. HA cells take 2 inputs, for instance bits c_1 and d_0 and compress them into 2 bits. The strategy to reduce the partial product tree has a major impact on the complexity of the multiplier. Particularly, tree multipliers are a class of multipliers working on this reduction of the partial product tree, as the Wallace tree multiplier that aims at reducing as early as possible the partial product bits or the Dadda tree multiplier that aims at reducing as late as possible the partial product bits. Array multipliers are a sort of tree multipliers. Array multipliers reduce the partial product tree using Carry-Save Adders (CSAs) and process the final addition with RCA. They are not the fastest or smallest multipliers but have a compact hardware layout because of their regularity, which makes them interesting for Very Large Scale Integration (VLSI).

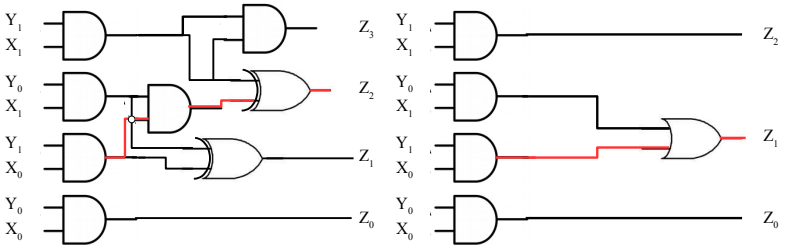
Inexact multipliers Several techniques to design approximate multipliers have been proposed, since they are main-basic blocks for Digital Signal Processing applications as Finite Impulse Response (FIR) filters.

The approximation can be done on the **generation of the partial products**. Kulkarni et al. [KGE11] used simpler structure to generate the partial products and to build a low-power 2×2 multiplier block: the Underdesigned Multiplier (UDM). The proposed basic block can be used to build larger multipliers. To build the proposed 2×2 multiplier block, the authors noticed that representing the output of a 2×2 multiplication on 3 bits

instead of 4 could lead to significant circuit simplifications. The modified Karnaugh table is represented in Table 2.6. As depicted in Table 2.6 in red, when representing the output on 3 bits, the only erroneous multiplication output is the output of 11×11, set to 111. The error rate of the proposed block is then $\frac{1}{16}$. The original accurate circuit and the approximate version are represented for 2-bit inputs in Figure 2.22. The approximation allows reducing the circuit area by 50%, and the critical path in both circuits is indicated in red. Larger multipliers can be built using the proposed 2×2 multiplier to generate the partial products. The mean relative error induced by the UDM ranges in $[1.39\%; 3.35\%]$ for 2 to 16-bit adders, while saving power between $[30\%; 50\%]$.

X \ Y	00	01	11	01
00	000	000	000	000
01	000	001	011	010
11	000	010	110	100
10	000	011	111	110

Table 2.6 – *Karnaugh table of a* 2×2 *UDM block.*



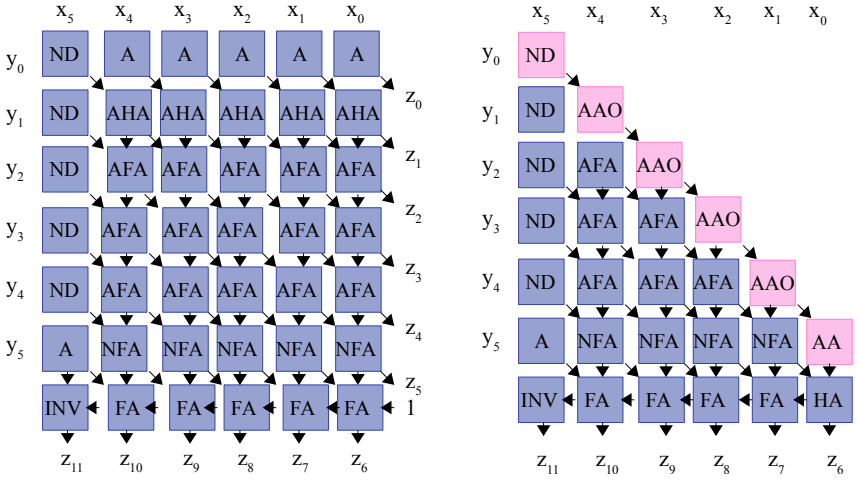
(a) Circuit of accurate 2×2 multiplier.

(b) Circuit of approximate 2×2 multiplier.

Figure 2.22 – *Circuits of accurate and approximate* 2×2 *multiply block, critical path in red.*

The approximation of multipliers can be done on the **partial product tree**. Approximate Array Multiplier (AAM) have been proposed, based on the efficient implementation of accurate array multipliers based on the Baugh-Wooley algorithm [Hwa79]. Van et al. [VWF00] proposed an approximate array multiplier, the lower error fixed-width multiplier called the AAM in the rest of the manuscript. The AAM is a fixed-width multiplier: for multiplying two inputs on $N-$bits, the output is formatted on $N-$bits. Consequently, during the multiplication, the LSBs are truncated preserving only the MSBs. The principle used in the AAM is to prune the cells used to compute the LSBs that will not be kept in the output. An error compensation mechanism is implemented to reduce the error amplitude. The structure of the AAM is represented in Figure 2.23(b) as well as the structure of the accurate array multiplier in Figure 2.23(a). As shown on the structure of the AAM, the cells to compute the partial products on the LSBs have been pruned.

Figure 2.23(a) shows the regularity of the layout of the array multiplier, which induces an efficient implementation for VLSI. To obtain a signed version of the array multiplier, the MSBs of all the partial products are inverted, except for the last partial product for which all the bits are inverted except for the MSB. In Figure 2.23(a), the different cells correspond to:



(a) Circuit of 6×6 array multiplier. (b) Circuit of 6×6 AAM.

Figure 2.23 – *Circuits of accurate and approximate* 6×6 *signed array multipliers, error compensation cells in pink. Extracted from [Bar17]*

- **ND**: NAND gate
- **A**: AND gate
- **AHA**: a half adder whose inputs are combined by an AND gate
- **AFA**: a full adder whose inputs are combined by an AND gate
- **NFA**: a full adder whose inputs are combined by a NAND gate

Classically, to implement a fixed-width array multiplier, the LSBs cells are simply truncated. The LSBs cells are the cells at the upper-right diagonal in Figure 2.23(a) which represent nearly half of the cells of the multiplier. Truncating the LSBs of the array multiplier has been proposed by Kidambi et al. [KEGA96], who demonstrated that the induced truncation bias and variance of the error were growing linearly with the size of the multiplier. Nevertheless, since this truncation induced a non-negligible error, the approximate multiplier depicted in Figure 2.23(b) has been proposed. Van et al. [VWF00] proposed to add an error compensation mechanism on the diagonal (cells depicted in pink). The correction cells are composed of a NAND gate, followed by $N-2$ AAO gates, where N is the size of the multiplier, which represent two AND gates and an OR gate and finally an AA gate composed by two successive AND gates. Consequently, the error compensation mechanism is very simple. The error map of the AAM is depicted in Figure 2.24.

A survey on approximate arithmetic operators is presented in [JHL15]. The study compares the different operators in terms of energy savings as well as error induced.

2.4.2 Circuit parameters modification

Another hardware approximation technique consists in producing a circuit which, under normal conditions, always functionnally match its design. Nevertheless, under modified conditions, the circuit gives approximate results. To do so, the fact that circuit manufacturers take margins on the indicated operating voltage/frequency of a circuit is exploited. These margins are originally added to ensure that the uncertainties induced by the variations of physical processes, temperature for instance, are negligible. However, these margins are generally greater than really needed. Lower voltage can then be applied without

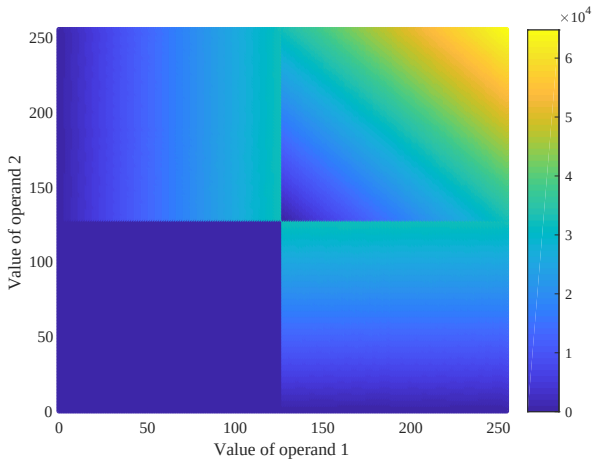


Figure 2.24 – *Error map of the 8-bit AAM, color scales correspond to the error.*

inducing any functionnal errors, if the voltage stays higher than critical voltage imposed by the critical path, or inducing errors if the applied voltage is too low to meet the critical path delay. In this case, the stability of the circuit is not guaranteed and errors can occur. Errors can be corrected using an error-correction circuit. Working at near or sub threshold achieves important energy savings. Indeed, transistors are generally considered as ideal switches [RD15]. If the operating voltage V_{dd} is higher than a threshold voltage V_T they conduct the current, and on the contrary if it is lower than V_T they are turned off. Nevertheless, in reality, the current is not completely cut off when $V_{dd} < V_T$ but it reduces exponentially. Consequently, the circuit can be operating at voltages lower than V_T, but reducing the operating voltage implies an increase in the circuit delay. V_T is called the critical supply voltage, the minimum supply voltage to produce a correct output.

The dynamic energy consumption being proportionnal to the square of the operating voltage, reducing V_{dd} by a factor K decreases the dynamic power consumption by a factor K^2.

Figure 2.25 indicates the evolution of energy consumption per operation depending on the supply voltage, while Figure 2.26 indicates the evolution of delay (in logarithmic scale) per operation depending on the supply voltage.

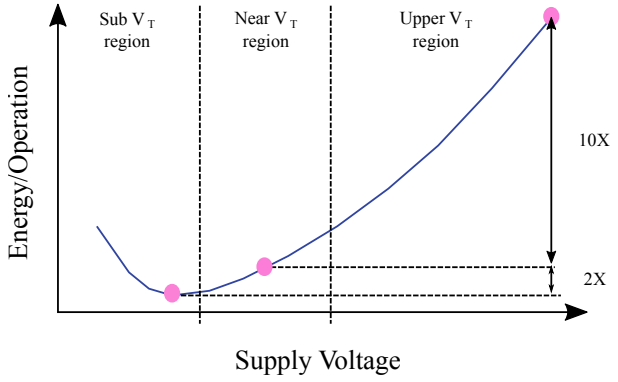


Figure 2.25 – *Impact of a decrease in voltage on the energy consumption from [Mit16].*

Nevertheless, reducing the supply voltage or increasing the clock frequency may break critical paths inducing timing errors and consequently computation errors. Reducing the supply voltage implies that the voltage is below the one at which the critical path delay is

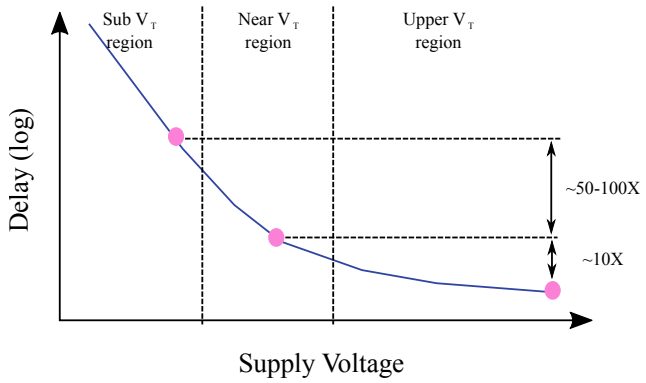


Figure 2.26 – *Impact of a decrease in voltage on the delay from [Mit16].*

equal to the sample period. The propagation time increases and may be higher than the critical path. In this case, the required results may not be available at the clock edge and the value considered for computation or result may be erroneous. Besides, circuits functionning at sub-threshold are suffering from exponential sensitivity to possible variations (for instance, temperature). The main problem of working at sub-threshold is process variability. Manufacturing processes are suffering from variation, for instance for two similar transistors, the oxide thickness or critical dimensions for instance are not exactly the same. The margin on the voltage to work at sub-threshold will not necessarily be similar for two similar circuits. To prevent from high-amplitude errors that may endanger the global circuit behavior, an error-correction circuit can be added to the approximate circuit. This error-correction circuit may induce a non-negligible overhead in terms of area, energy and delay.

For instance, Hegde et al. [HS01] proposed soft Digital Signal Processors (DSP), that is to say scaling the supply voltage of a DSP system beyond V_T. By proposing algorithmic noise-tolerance, an error-compensation mechanism to heal the system from errors induced if the critical path is triggered, Hegde et al. adress other errors sources as deep submicron noise. Deep submicron noise represents all the noise sources in deep submicron technology due to the reduced feature sizes, the reduced supply voltages, the interconnects or the density of manufactured System on Chip (SoC) [HS98]. Nevertheless, to be efficient, soft DSP depends on the scaling of the operating voltage, the error frequency as well as the overhead of error-compensation mechanisms. To create an efficient error-compensation mechanism, algorithmic noise-tolerance relies on the study of the transfer function as well as on the inputs and outputs distributions. Finally, the error-compensation mechanisms can also be erroneous. Tang et al. [TBJJ11] have proposed a new metric that allows taking into account the imperfection of the error-compensation mechanism to heal a system from deep submicron noise.

According to voltage scaling, benefits can be demonstrated with a simple 8-bit RCA. Figure 2.27 indicates the percentage of input combinations requiring a path delay of $k \times T_{FA}$ to be added, T_{FA} being the path delay of a FA. Operating at V_T, the critical path of the 8-bit RCA is equal to $8 \times T_{FA}$. However, Figure 2.27 indicates that more than 95% of the input combinations can be processed in $5 \times T_{FA}$. If the voltage is reduced such that the new FA delay is equal to $\frac{5}{8} \times T_{FA}$, the probability of error occurrence is equal to 0.05.

Soft DSP allows reducing the energy dissipation for filter-based systems within a range of 60-81% for a degradation of 0.5 dB on the output signal to noise ratio.

Ernst et al. [EDL+04] proposed Razor, a method that adapts the operating voltage depending on the induced error rate. As the method proposed by Hedge et al., Razor heals

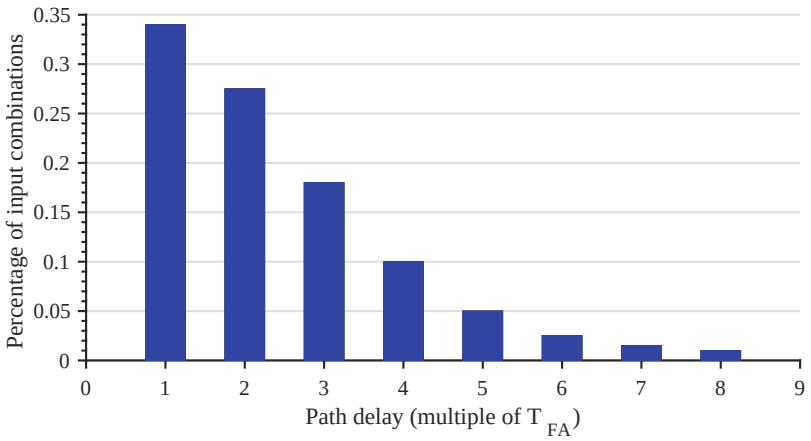


Figure 2.27 – *Proportion of input combinations (%) requiring $k \times T_{FA}$ to be added using a RCA, results from [HS01].*

the system from errors induced by voltage scaling or deep submicron noise. To detect and correct induced errors, Razor combines architectural and circuit-level techniques. Each flip-flop in the design is augmented to allow detecting if the computation could not finish on time. Using the Razor method, the ER depending on the operating voltage has been measured for a 18×18 Field-Programmable Gate Array (FPGA) multiplier block operating at 90MHz and 27°C and is depicted in Figure 2.28. This method can achieve up to 35% energy savings while inducing an error at a frequency of 1.3%. The obtained results using the Razor method clearly show the conservative margin taken by manufacturers to ensure a good behavior of their circuits, since no error occurs at the safety margin nor at zero margin.

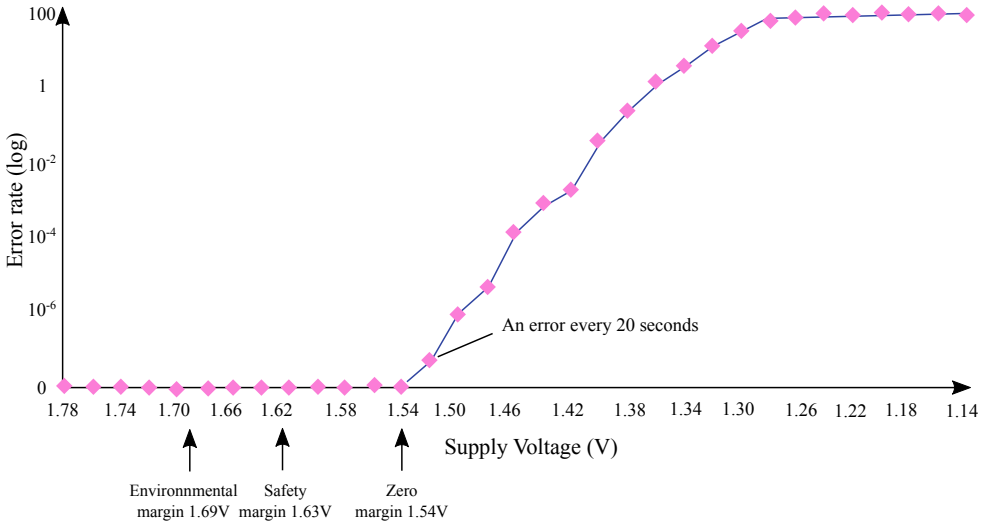


Figure 2.28 – *Error rate depending on the operating voltage for a 18 × 18 FPGA multiplier block, results from [EDL+04].*

Constantin et al. [CWK+15] proposed a predictive instruction-based dynamic clock adjustment to dynamically study the critical paths of each instruction so as to apply voltage scaling to achieve energy consumption reduction in a pipelined processor. This technique achieves up to 24% power consumption reduction. Indeed, to derive the operating frequency of a circuit, static timing analysis is done on the circuit under the worst

conditions at which the critical path is always respected. In this method, the instructions determining the critical path at instant t are the instructions in the pipeline in the current stage. Dynamic Voltage and Frequency Scaling (DVFS) benefits from the fact that the critical path is not always reached.

To conclude, voltage over-scaling has not yet been adopted by industrials because of its non-reproductible behavior from a chip to another.

2.4.3 Unreliable storage of resilient data

In 2001, the International Technology Roadmap for Semiconductors [AEJ+02] was predicting an increase in the area occupied by embedded memory in SoC from 52% to 94% in 4 years. Efficiently managing the power required by a SoC goes with efficiently managing the memory system. Besides, the SoC yield is strongly dependent on the memory yield. Nevertheless, while targeting efficiency when designing embedded memories, the number of expected faults increases. Errors in memory appear because of the shrinking size of manufactured components, particle strikes or a use at a lower voltage. To ensure a sufficient quality, repair strategies have been proposed. For instance, redundancy in the elements in the memory array has been proposed [LYHW10] to repair the faulty memory cells. However, using redundancy in the memory elements goes along with an area and cost overhead.

To avoid redundancy, Smolyakov et al. [SGGL13] proposed to manage differently faulty memory cells depending on their impact on the overall result. The sensitivity of the result to a fault on a memory cell is evaluated based on the Bit Error Rate (BER) metric. For instance, a faulty LSB will lead to a lower sensitivity to faults than a faulty MSB and data before filtering are less sensitive to faults. After having evaluated the sensitivity of memory cells, faulty memory blocks leading to a high sensitivity can be permuted with memory blocks with lower sensitivity. Finally, a forward error correction at system level is proposed to allow correcting faults while avoiding the area overhead caused when using redundant modules.

Frustaci et al. [FKB+15] proposed another type of memory for error-tolerant applications. They proposed to dynamically apply voltage scaling to Static Random Access Memory (SRAM) depending on the energy/quality trade-off required by the considered application. Indeed, voltage scaling is generally limited at the system level because of the SRAM embedded in the system. The faults due to voltage scaling on SRAM are either due to a difference between the bitcell speed and the operating frequency, or due to random process variations leading to wrong read write margins which are decreasing as the operating voltage scales down. Besides, the BER increases exponentially as the operating voltage scales down. Consequently, the MSBs have to be protected and the reduction of the operating voltage has to stop whenever the quality is degraded too much, allowing only a few LSBs to be wrong to get an acceptable quality (higher than 30 dB). This solution is not satisfying in terms of energy savings and since to get enough savings, the majority of LSBs is wrong, it is better not to transport them. As proposed by Smolyakov et al. [SGGL13], the quality can be adjusted by applying error correction to a few sensitive MSBs using for instance selective bit-level circuit techniques and allowing more aggressive voltage scaling. The proposed voltage scaling on SRAM allows saving up to 32% energy for similar quality on a 32kb SRAM testchip in 28nm.

Another method to reduce energy-consumption is to lower the Dynamic Random Access Memory (DRAM) refresh rate. DRAM is a volatile memory which implies that constant refreshing is required to keep the data stored in cells consistent. The refresh rate

is generally conservative to ensure that no data is lost even if physical parameters are changing, for instance, an increase in the temperature. However, frequently refreshing the DRAM implies an increase in the power consumption. This rate can be reduced with error-resilient data. Nevertheless, this process is highly dependent on physical parameters as the considered chip, the temperature or the management of the power between successive refreshes, leading to unpredictable behavior. To analyze the influence of these parameters, Rahmati et al. [RHHF14] proposed to create a reproducible platform allowing reproducing the results for different temperature ranges and varied power management strategies. For instance, the impact of the implemented power management strategy, namely power gating, has been measured and reproduced in Figure 2.29.

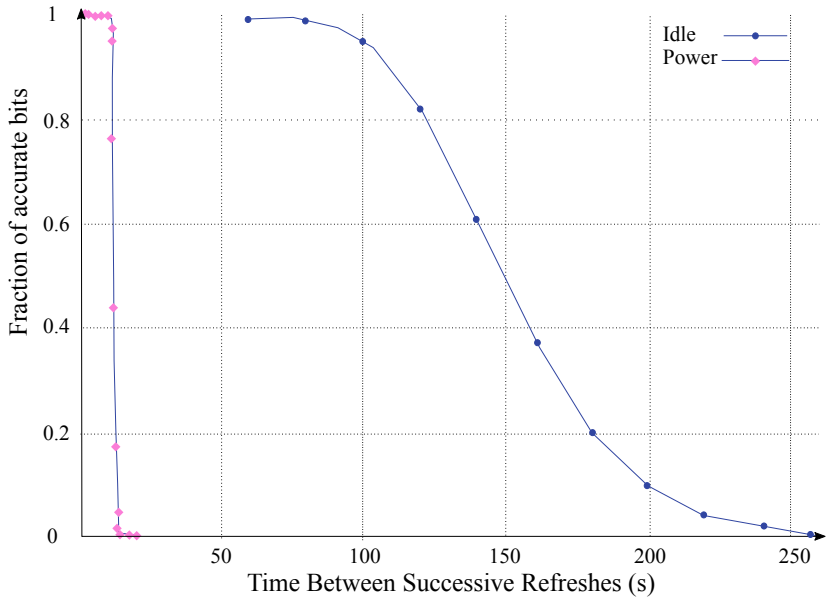


Figure 2.29 – *Impact of power management on the fraction of accurate bits when reducing the DRAM refresh rate, results from [RHHF14].*

The authors studied the fraction of correct bits depending on the refresh rate, in the case when the power is completely cut off between successive refreshes and in the other case when the power in maintained. The proposed results have been obtained at a constant temperature of 30°C, and indicate that when the power is cut off between refreshes, the accuracy drops by an order of magnitude faster than when the power is maintained. This technique to save energy must then be used carefully not to damage the quality at the output of the application. Another important factor to take into account when reducing the DRAM refresh rate is temperature. Increasing temperature affects the power leakage at the transistor level. As a consequence, temperature increases logarithmically the volatility of the data contained in the DRAM. To end with, even though reducing the DRAM refresh rate can lead to important power savings, several important factors have to be taken into account to ensure the accuracy of the stored data.

Finally, the error-resilient data can be compressed using a compression algorithm that will cause lossy information but will allow increasing the data transfer rate and decreasing the energy consumption. This technique has been proposed by Reinhardt et al. [RCHS09] for wireless sensors networks. Energy is a critical resource in wireless sensors networks because it is battery-powered. In wireless sensors networks, the data collected by sensors have to be periodically transmitted to a base station with a low latency, since data have to constantly be up-to-date. The power consumption of the communication unit in a sensor

network is generally higher than the power required for transmitting or receiving. Data compression before transmission allows reducing the transmission time or increasing the density of information transmitted for a similar cost. Nevertheless, the data compression is done with additional microcontrollers operations, thus at the detriment of energy. A trade-off between compression and power savings has to be found.

2.5 Background: conclusion and perspectives

In this chapter, a non exhaustive description of AC techniques was presented. Several techniques within the different levels to apply AC were described. When studying the different contributions in the existing literature, the striking need for error analysis techniques appeared. Indeed, when proposing a new AC technique, the performance of the technique in terms of energy consumption, area or latency are studied. However, the error induced by these techniques as well as their impact on the output application quality metric is generally done with exotic metrics and the link to the application is often non-reproducible. The study of the impact of approximations on the application quality metric is critical to use AC techniques in the industry. The main goal of this thesis is to model the errors induced by various AC techniques. A thorough understanding of the approximation techniques is consequently a pre-requisite to further work on error modeling.

In the next chapter, a description of existing error modeling techniques is presented.

CHAPTER 3

Related Works: Analysis of the Impact of Approximate Computing on the Application Quality

> "*Les erreurs sont les portes de la découverte*"
>
> James Joyce

3.1 Introduction

As presented in Chapter 2, Approximate Computing (AC) is one of the main approaches for post-Moore's Law computing. It exploits the error resilience of numerous applications in order to save energy or accelerate processing. The numerical accuracy of an application is now taken as a new tunable parameter to design more efficient systems. Nevertheless, the numerical accuracy of an application has to stay within an acceptable limit for the output of the application to be usable. For this reason, the impact of the induced errors on the application has to be studied. Before analyzing the effects of the errors induced by the chosen approximations on the application quality metric, the errors induced by the AC technique themselves have to be characterized. A thorough characterization of the approximation error allows, during the Design Space Exploration (DSE) phase, choosing the most suitable AC technique with respect to the implementation constraints and to quantify the impact of the approximation on the application quality metric. The impact of the approximation on the application quality metric is generally evaluated numerous times because the DSE requires testing many configurations. Consequently, this evaluation has to be fast so as to limit the DSE time.

AC techniques generate various error profiles. When implementing AC in an application, the objective of error analysis is to derive the impact of the induced approximations on the application quality metric.

The evaluation of the impact of the approximation on the application quality metric can be done in three steps as presented in Figure 3.1.

The errors induced by the approximations are first characterized so as to provide an AC source error model according to several error metrics for further error propagation. The error characterization is presented in Section 3.4 and can be done with two different

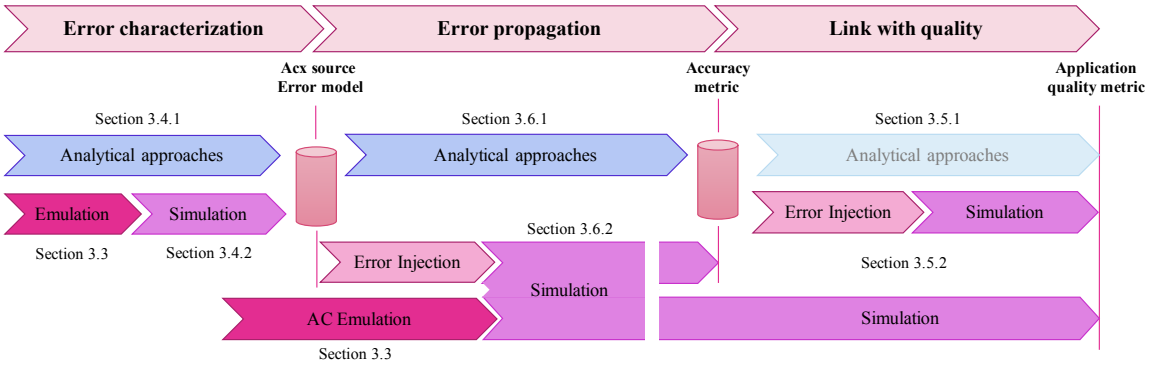


Figure 3.1 – *Different steps to analyze the impact of approximation on quality.*

techniques. Analytical techniques are presented in Section 3.4.1 provide a mathematical model of the error. Simulation-based techniques are presented in Section 3.4.2 and are more generic but require the emulation of the implemented approximation. The different methods to emulate AC techniques are presented in Section 3.3. Varied error metrics are used for error characterization. The error metrics can be the mean error amplitude μ, the standard deviation of the error σ, the probability mass function (PMF) of the error, or an interval that contains the error bounds.

Then, the errors are propagated through the application. This step described in Section 3.6 allows deriving an intermediate accuracy metric as for instance the noise power P, or the Signal-to-Quantization-Noise Ratio (SQNR). As for the error characterization step, analytical techniques presented in Section 3.6.1 can be used. Simulation-based techniques presented in Section 3.6.2 can also be used along with emulation or error injection techniques.

Finally, the approximation errors may have to be linked to the output quality as presented in Section 3.5. In this case, a few analytical approaches have been proposed and are described in Section 3.5.1. Analytical approaches are depicted as shaded in Figure 3.1 since no general analytical approach has been proposed. They have to be derived specifically for a quality metric. Simulation-based techniques are generally preferred and are described in Section 3.5.2. They can be implemented along with error injection.

3.2 Metrics to analyze the impact of AC

3.2.1 AC error and accuracy metrics

Introducing approximations in an application is equivalent to tolerating errors in the application. Emulation techniques have been proposed to reproduce the introduced errors at the approximation technique level, and are presented in Section 3.3. The introduced errors are characterized and modeled with error metrics to ease the process of linking the induced errors to the quality evaluation function of the application. The error of approximation $\widehat{e_i}$ on a given input of the computation is expressed as:

$$\widehat{e_i} = \widehat{z} - z \tag{3.1}$$

where $\widehat{z}$ and z are the erroneous and exact outputs of the computation, respectively.

Numerous error metrics have been proposed and the choice of the considered error metrics depends on the implemented AC technique as well as on the nature of the output of the application, as presented by Akturk et al. [AKK15]. To quantify the errors induced

by an approximation, the deviation between the approximate output and the accurate output has to be measured. Nevertheless, as explained by Akturk et al. [AKK15], the considered error metric has to be robust to several side-effects. For instance, when taking a relative error metric, the reference should not introduce any bias on the measured deviation. Besides, averaging the error metric over a certain number of points may hide large deviations on particular points, hence the need for a metric to measure the extreme errors. Another important point in deriving error metrics is to define how to aggregate errors.

When these restrictions have been taken into account, different metrics to compute the deviation exist: statistical, bitwise and interval-based metrics.

Statistical metrics

According to Chippa et al. [CCRR13], the induced errors by an AC technique can be characterized with statistics according to three parameters. The three parameters are all derived from the deviation expressed as the error distance e_i:

$$e_i = |\widehat{z} - z| \tag{3.2}$$

The first parameter to characterize the error is the mean error amplitude, μ_e, which corresponds to the average value of the different error distances e. The second parameter f is the Error Rate (ER), which represents the frequency of error occurrence. The third parameter is the standard deviation of the error e_{rms} which represents the dispersion around the average error value and is considered as the error predictability by Chippa et al. [CCRR13]. The computation of the three parameters is detailed in the following Equations:

$$\mu_e = \frac{1}{N} \sum_{i \in \mathcal{I}} e_i \tag{3.3}$$

$$f = \frac{1}{N} \sum_{i \in \mathcal{I}} f_{e_i}, \text{with } f_{e_i} = \begin{cases} 0 \text{ if } e_i = 0 \\ 1 \text{ else} \end{cases} \tag{3.4}$$

$$e_{\text{rms}} = \sqrt{\frac{1}{N} \sum_{i \in \mathcal{I}} {e_i}^2} \tag{3.5}$$

For a continuous statistical distribution of the error, the different parameters are represented with a distribution as in Figure 3.2. This continuous distribution is called a probability density function (PDF). The ER is represented by the area under the distribution, the mean error amplitude is the mean value of the distribution, and the error predictability is the standard deviation of the distribution.

When the distribution of the error is discrete, as for instance with inexact arithmetic operators, the statistical distribution of the error is the probability mass function (PMF). The PMF of the approximation error is the function indicating the probability that the error distance is equal to a particular value. It represents the Error Rate (ER) depending on the Error Distance (ED) of the induced errors. The PMF can be represented as a bar chart, and the higher a bar is, the more frequent the considered error occurs. PMFs can be highly asymmetric as presented in Figures 3.3(a) and 3.3(b) for the Almost Correct Adder (ACA) on 16 bits with a carry chain length of 4 and the Approximate Array Multiplier (AAM) on 16 bits, respectively. These PMFs have been obtained with 10000 uniformely drawn inputs in the input space $[0; 2^{16}]$. The more inputs are drawn, the more accurate

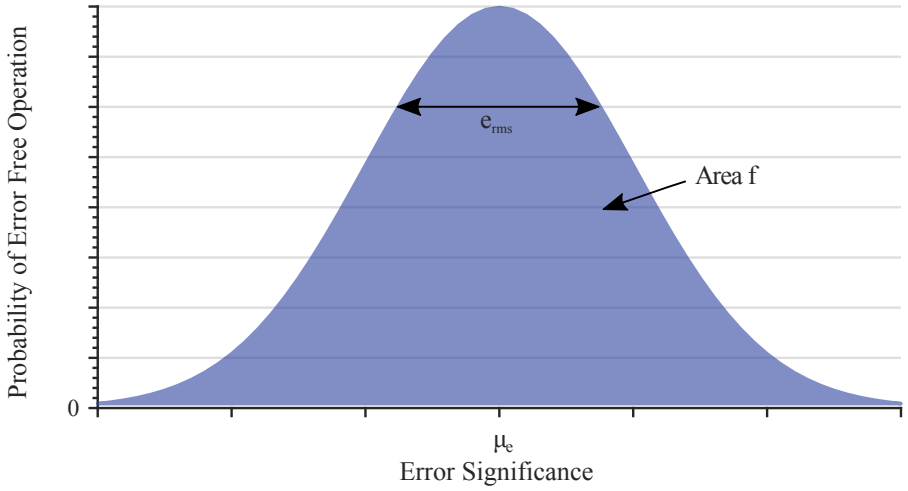


Figure 3.2 – *Continuous statistical distribution of the error: Probability Density Function.*

the PMF is. To build the PMF of an approximate operator error, or statistics on the error, Monte Carlo simulations are generally used [CSE15].

The error can also be characterized in terms of Maximum Error Distance (maximum ED) M_e defined as:

$$M_e = \max_{i \in \mathcal{I}} e_i \tag{3.6}$$

Bitwise metric

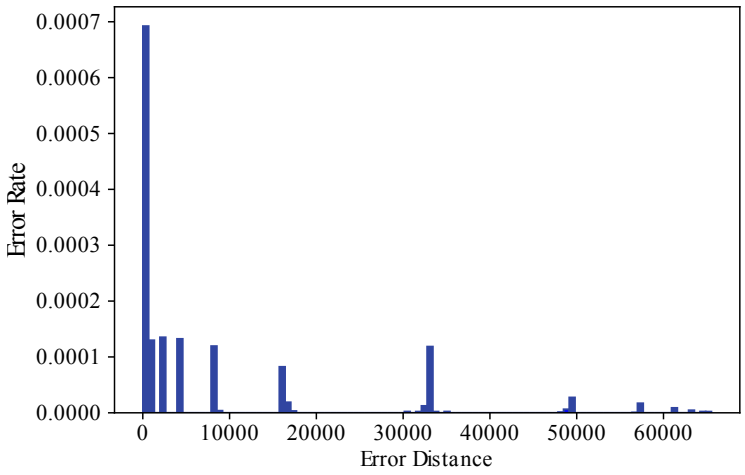
In digital communication systems, a widely used error model is the Bit Error Rate (BER). Indeed, the goal of a digital communication system is to receive the closest data to the data that was sent by the transmitter. The BER indicates the system performance and is expressed as a ratio that measures the number of corrupted bits in a given number of transmitted bits [Fre15] as in Equation 3.7.

$$BER = \frac{\text{Number of erroneous bits}}{\text{Total number of transferred bits}} \tag{3.7}$$

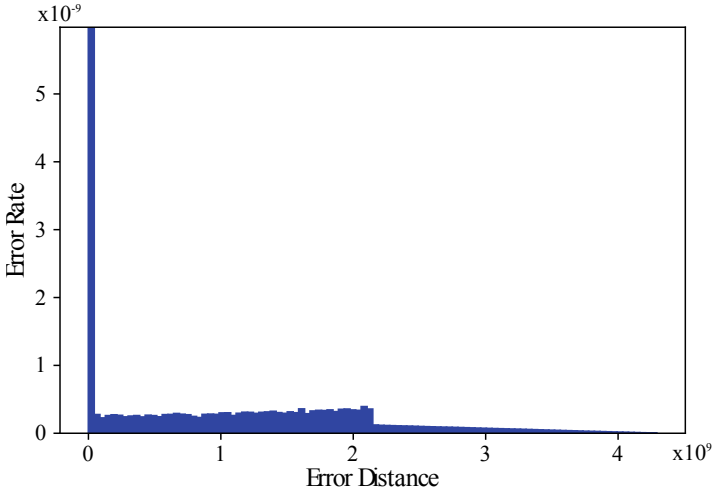
A similar metric can be used for inexact operators with the Bitwise Error Rate (BWER). The BWER represents the BER of each bit position in a binary word. For instance, if x is an n-bit binary input, the BWER is a vector $\mathbf{b}$ depending on x expressed as $\mathbf{b} = \{p_i\}$ with $i \in [\![0\ ;\ n-1]\!]$, where p_i is the probability of bit i in x to be erroneous. Numerous methods have been proposed to propagate the BWER error metric through an inexact operator, and are presented in Section 3.6.1.

Interval-based metric

Error metrics can be represented by intervals and propagated by Interval Arithmetic (IA). IA has first been proposed by Ramon Moore [Moo62] and consists in propagating intervals instead of real numbers in the application. For instance, if variable x lies in $[\underline{x};\overline{x}]$, the interval will be propagated through the different computations of the application. IA is for instance used to bound the effects of round off errors in computations and allows guaranteeing the output accuracy. For instance, intervals are used to produce conservative



(a) Almost Correct Adder, $N = 16, C = 4$.



(b) Approximate Array Multiplier, $N = 16$.

Figure 3.3 – *Probability Mass Function of two inexact operators.*

error bounds for the computations in a digital computing machine where the computations are a succession of rounded arithmetic operations whether it be in floating-point or in fixed-point arithmetic. In this case, the produced interval is guaranteed to contain the accurate output of the computation and the radius of the interval is the error bound.

Nevertheless, in the case of errors induced by inexact operators, the error may be unsmooth and the resulting PMF highly asymmetrical. To better render the error induced, Huang et al. [HLR11] proposed an adaptation of IA to inexact circuits called Modified Interval Arithmetic (MIA). In the proposed method, each bar of the PMF of an operator is modeled by an interval, as:

$$\text{MIA}(x) = P(a \leq x \leq b), \text{ if } a \leq x \leq b \tag{3.8}$$

3.2.2 Application quality metrics

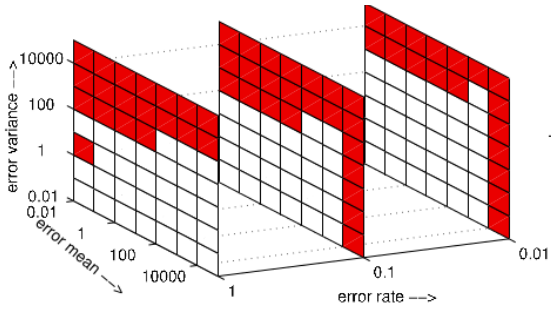
The characterization of the error induced by AC allows knowing the impact of the approximation on the application output quality. This impact can be measured with the application quality of service (QoS) or with an intermediate quality metric. This step corresponds to Blocks 2 and 3 in Figure 3.1. The application quality metric, whose nature and measurement depends on the application, quantifies the output quality of the application. For instance, for a signal processing application, the application quality metric can be the signal-to-noise ratio (SNR), whereas for an image processing application, the application quality metric can be the Structural Similarity Index Measure (SSIM). The application quality metric is used to compare the output generated by the approximate version of the algorithm with the output generated by the reference version of the algorithm. Nevertheless, in some cases, an intermediate metric can be easier to compute. An intermediate metric is a generic error metric independent from the QoS metric, and that may be linked to the QoS of the application.

Application quality metrics are generally user-defined quality evaluation functions and have to be provided along with the error tolerance of the application. For an application, several quality evaluation functions can be used and the impact of the approximation on the different functions may strongly vary. Different quality evaluation functions have been reported in Table 3.1 for various error-tolerant applications.

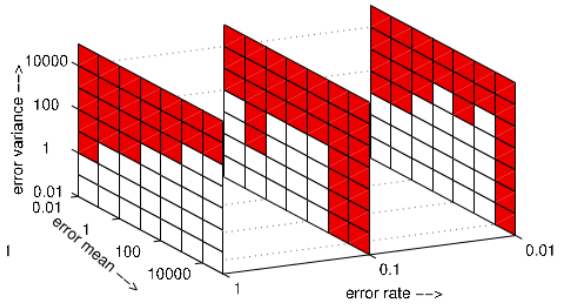
Data Processing Domain	**Quality Evaluation Function**
Digital Signal Processing	Peak Signal to Noise Ratio Mean Squared Error Relative difference
Image Processing	Middlebury metric (stereovision) SSIM Mean Squared Error Pixel Difference
Image Segmentation	Percentage of Misclustered Points Mean Centroid Distance
Video Processing	Peak Signal to Noise Ratio SSIM
Communications Systems	Bit Error Rate
Web Search	Number of Correct Results in Top 25 Results
Classification	Classification Accuracy: Percentage of Correctly Classified Points

Table 3.1 – *Various application quality metrics depending on the nature of the application.*

The impact of the choice of the quality function evaluation has been studied by Chippa et al. [CCRR13] for image segmentation with k-means clustering. The two considered metrics are the percentage of miclustered points and the mean centroid distance. The obtained error characteristics depending on both quality evaluation functions are presented in Figure 3.4 extracted from [CCRR13]. The white parts on Figure 3.4 indicate an acceptable error, in this case the degradation in the output quality is lower than 1%. Figure 3.4(a) uses as quality evaluation function the mean centroid distance while Figure 3.4(b) the percentage of mis-clustered points. It can be seen that the application appears as being more resilient in the case of mean centroid distance since the white parts are larger.



(a) Metric: mean centroid distance.



(b) Metric: percentage of mis-clustered points.

Figure 3.4 – *Impact of quality evaluation function on error-resilience (from [CCRR13]).*

Gillani et al. [GK17] studied the impact of the quality function evaluation on the error resiliency of a particular iterative workload: the StEFCal [SW14], a radio astronomy calibration algorithm. Gillani et al. noticed that when using the quality function classically used for the evaluation of the accurate algorithm, namely, the convergence criterion, that computes the relative Euclidean distance between the vectors of current and previous solution, the insights on the application quality when implementing AC were not useful. When evaluating the quality of the approximate version of the algorithm using this quality function, it happened that the solution had converged in the wrong plane. To prevent from this phenomenon, they proposed another metric that takes into account the convergence as well as the distance to the accurate solution.

In our contributions, we have studied the dependence of error-resiliency to the quality evaluation function. The considered application is a stereovision algorithm that produces a depth map, and the implemented AC technique is an algorithm-level approximation detailed in Chapter 4. Two quality evaluation functions have been tested. The Middlebury metric is particular to the evaluation of the depth maps quality and gives the percentage of good pixels according to a ground truth. The Middlebury metric takes into account an error threshold ϵ. If the depth of a given pixel is different from the depth of the same pixel in the ground truth map from less than ϵ disparity levels, the pixel is counted as correct. The Middlebury metric does not evaluate the smoothness of the disparity map and does not render the human visual perception of quality. The other considered metric is the SSIM which better renders the human visual perception of quality. The proposed approximation technique was rejected according to the Middlebury metric while accepted when the quality function evaluation was the SSIM.

3.2.3 Conclusion

Numerous error metrics have been proposed to quantify the errors induced by AC techniques. In this thesis, a focus has been made on the statistical error metrics since they possess mathematic properties to estimate them as well as confidence information on the obtained estimations. According to quality metrics, in Chapter 4, it is demonstrated that the choice of the quality metric has a strong impact on the tolerance of the application to imprecisions. In the proposed contributions, an intermediate metric has been used to get away from the dependance to the quality metric.

3.3 Approximations emulation techniques

Numerous techniques are using simulations to evaluate the impact of AC on the application quality metric or on an intermediate accuracy metric. In this case, the approximation mechanism may have to be emulated so as to reproduce its internal behavior. Emulation techniques have mainly been proposed for inexact arithmetic operators as well as for finite precision arithmetic. In both cases, emulation can be done with functional simulation techniques used to reproduce the behavior of the approximation instead of simulating the approximation behavior on the hardware. Functional simulation techniques for inexact operators and finite precision aim at reproducing the behavior of the approximation at the logic level and at the bit level, respectively.

3.3.1 Inexact arithmetic operators

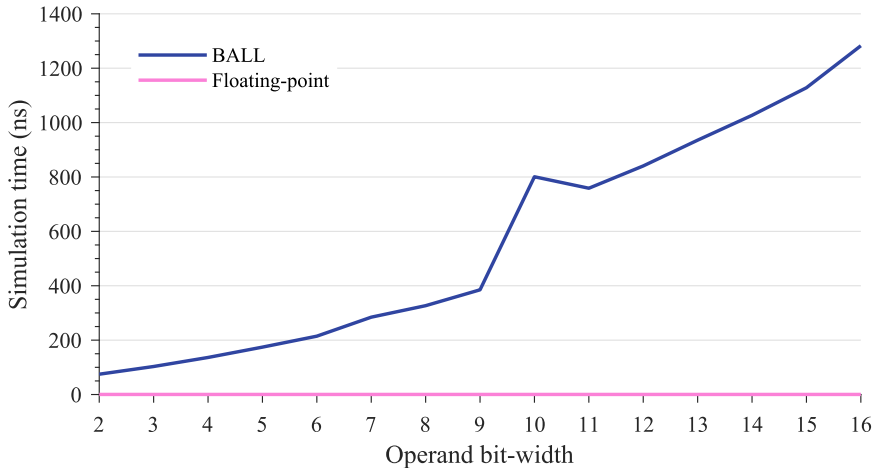
The functional simulation of the behavior of inexact arithmetic operators is used in [DVM12, JLL+17, CCSE18] to characterize the induced errors or to analyze the QoS at the output of an application implementing inexact arithmetic operators. In this case, emulation by functional simulation allows studying the behavior of the inexact operator before the hardware implementation. Nevertheless, the emulation of inexact arithmetic operators is complex. To mimic the behavior of inexact arithmetic operators, emulation is done with bit-accurate simulations at the logic level (Bit-Accurate Logic-Level (BALL) simulations) that are required to reproduce the internal structure modifications of the operator at the logic-level. The complexity of reproducing the internal behavior of the operator leads to long simulation times. For instance, as presented in Figure 3.5, the BALL simulation time of a 16-bit inexact adder, in this case, the ACA, is around 300 times longer than the one of a native accurate processor instruction, represented in Figure 3.5(b) as floating-point simulation. In the case of an inexact multiplier, in this case the AAM represented in Figure 3.5(a), it is 4000 times longer.

The ratio r between the BALL simulation time and the simulation time for the corresponding accurate floating-point operation of several 32-bit inexact operators are indicated in Table 3.2. The ratio r is evolving in $[31942; 73864819]$.

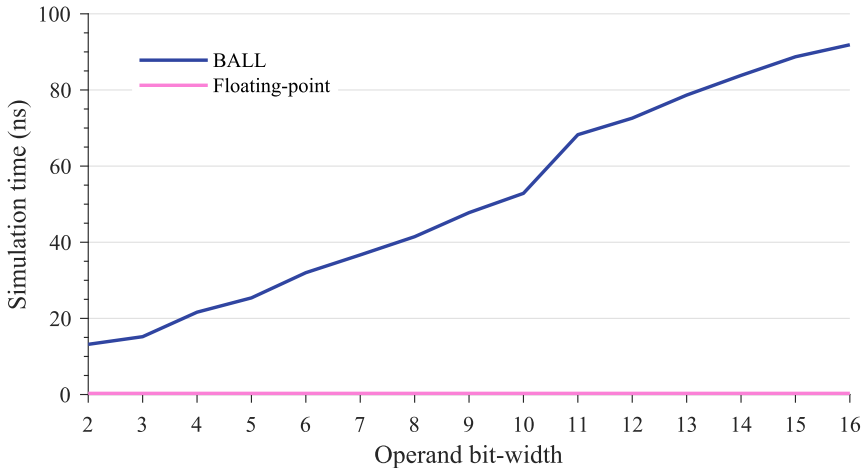
For the emulation of 32-bit inexact arithmetic operators, the required time is very long. Consequently, the simulation of a whole application so as to analyze the impact of 32-bit inexact operators on the QoS at the output of the application becomes prohibitive, if not impossible.

3.3.2 Fixed-point arithmetic

To mimic the finite precision effects, fixed-point arithmetic can be emulated. Several commercial high-level tools to design digital signal processing applications can be used to



(a) AAM.



(b) ACA.

Figure 3.5 – *Comparison of the simulation time for the BALL and floating-point simulation of two inexact arithmetic operators.*

	Op. name	r
ADD	ETAIV	31942
	ACA	859406
	ISA	1799519
MPY	AAM	13375154
	ABM	73864819

Table 3.2 – *Ratio r between BALL simulation and simulation of accurate floating-point operation times for 32-bit operators.*

emulate fixed-point arithmetic, as Signal Processing Worksystem (Cadence), DSP Station (Frontier Design), CoCentric (Synopsis) [BN04], or C++ classes that have been proposed in SystemC [GLMS10, MRR07, AT03]. More recently, Matlab/Simulink have proposed a fixed-point designer toolbox [BBW14] to emulate the behavior of an application in finite

precision. Given the target architecture, the fixed-point simulation of the application can be bit-accurate.

C++ based fixed-point data-types are particularly slow to simulate since they can be two to three orders of magnitude slower than the execution of floating-point data-types. The emulation of fixed-point arithmetic is done on floating-point architectures. The integer word-length, the total word-length as well as the quantization and overflow modes can be specified. The quantization mode specifies how to manage a value whose accuracy is greater than the one of the fixed-point variable embedding it, while the overflow mode specifies how to manage a value whose amplitude is larger than the largest that can be encoded on the fixed-point variable. Two different types of fixed-point simulation have been proposed: 1) Constrained data types also called static fixed-point data-type `sc_fixed` with data-type arguments known at compile time. 2) Unconstrained data types also called dynamic fixed-point data-type `sc_fix` with data-type arguments that can be variables and then modified. The static fixed-point data-type simulation is faster than the dynamic one, but the application has to be re-compiled each time the data word-lengths are modified. To improve the simulation speed of SystemC fixed-point data-types, a type `_fast` has been proposed for both constrained and unconstrained SystemC data-types, limiting the precision to 53 bits.

3.3.3 Operator overloading and approximate data types for simulation

For an efficient simulation of an application implementing AC techniques, Sampson et al. proposed EnerJ [SDF+11], a Java extension with type qualifiers to indicate which data are approximated and which data have to be accurate. The programmer annotates the code implementing its application and indicates the approximable parts, and error-sensitive parts. Approximate storage, for instance unreliable memory modules as unreliable registers, data caches, or main memory, and computation are emulated to allow quality analysis at the output of the simulation. The inexact arithmetic operators are implemented by overloading the existing accurate operators. Several AC techniques may be emulated through EnerJ, as Dynamic Voltage and Frequency Scaling (DVFS), reduced width in floating-point operations or reduction of the Dynamic Random Access Memory (DRAM) refresh rate.

3.3.4 Conclusion

Emulation is an important part of error modeling for AC since it allows avoiding testing the implemented technique within the real system. However, the proposed methods to emulate the impact of inexact operators or finite precision arithmetic for instance lead to long simulation times which impedes the use of exhaustive simulations for characterizing the approximation error and limits the possibilities for the design space exploration. The different abstraction levels for emulation and their associated times are represented in Figure 3.6. For AC techniques at the computation level, the whole application has to be simulated for emulation with native data-types which leads to low emulation time. Finite precision arithmetic has to be emulated at architecture level and inexact circuits at circuit levels, which leads to moderate emulation time. The longest to emulate is voltage over-scaling which has to be emulated at technological level, that is to say at the transistors level.

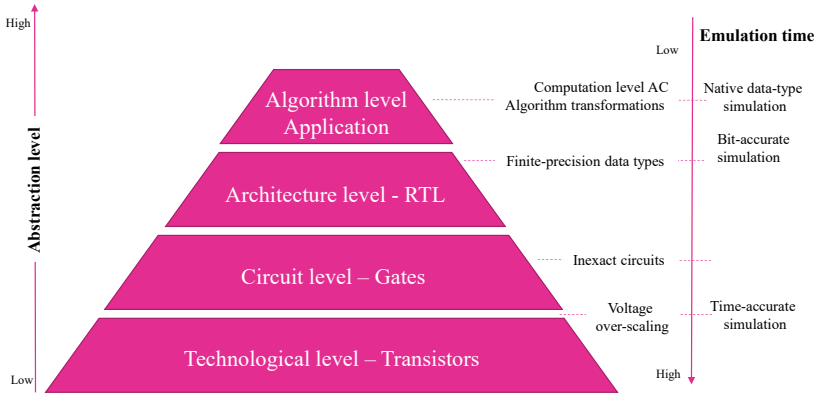


Figure 3.6 – *The different abstraction levels and times for emulation.*

3.4 AC error characterization

From the different emulation techniques presented in Section 3.3, the induced errors by the AC technique are reproduced and can be characterized with varied techniques. Two types of state-of-the-art techniques exist to characterize the error metrics intrinsic to the implemented approximations: analytical and simulation-based techniques.

3.4.1 Analytical techniques

Analytical techniques have been proposed to provide a mathematical model to evaluate the considered error metric.

Widrow model for fixed-point arithmetic

Using a statistical representation of the induced error, an error model can be created for various AC techniques. For instance, for modeling the error induced by fixed-point arithmetic, the Widrow model [WKL96] can be used. The quantization error by truncation is characterized with statistical parameters as the mean error amplitude, μ_e and the standard deviation e_{rms} expressed as:

$$\mu_e = \frac{q}{2}(1 - 2^{-k}) \tag{3.9}$$

$${e_{\text{rms}}}^2 = \frac{q^2}{12}(1 - 2^{-2k}) \tag{3.10}$$

where k represents the number of eliminated bits and $q = 2^{-n}$ where n is the number of bits to encode the fractional part after quantization. Using the statistical error model, the quantization process in fixed-point arithmetic can be replaced by an additive white noise, as presented in Figure 3.7 with the following properties:

- Stationary and ergodic random variable.
- Non-correlated with the input signal x.
- Independent from the other noise sources.
- Uniform distribution.

In Figure 3.7, $x(n)$ represents the input signal at time n, $x_Q(n)$ the input signal after conversion in fixed-point (Q for quantization) and $e_x(n)$ the statistical error model characterized by its first and second order moments expressed in Equation 3.9.

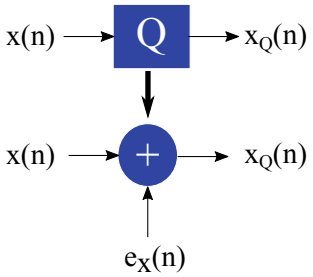


Figure 3.7 – *Widrow model for fixed-point quantization noise.*

Probabilistic analysis - inexact operators

To characterize the error metrics of inexact operators, several analytical techniques have been proposed. Liu et al. [LHL15] analytically derive estimated values for the ER and the Mean Error Distance (mean ED) of several block-based inexact adders, namely the ACA, the Error-Tolerant Adder Type II (ETAII), the Equally Segmented Adder (ESA) and the Speculative Carry Selection Adder (SCSA). After having established how to compute the signal propagate p_i, which indicates whether a carry signal is propagated to the i^{th} sum bit, they handle the derivation of error metrics for the different adders separately. The assumption that inputs are uniformely distributed is taken.

To derive the error metrics of the ACA, the authors form the universal error set composed by all the possible error patterns in the inexact operator. With a n-bit ACA, it is possible to derive n disjoint subsets whose union form the universal error set. Each subset, denoted Π_i, is composed of the error patterns in which the i^{th} bit is erroneous, the upper bits are accurate, and the lower bits are either accurate or not. The total mean ED of the operator is then defined by the sum of the mean ED in each subset. The mean ED in each subset Π_i is approximately equal to $2^i \cdot q_i$ where q_i is the probability to be in the considered subset Π_i. Indeed, the error induced by the i^{th} bit is dominant while the possible errors on the Least Significant Bits (LSBs) may cancelled each other.

The ER can be derived as $\sum_i q_i$. Through the probabilistic analysis of the inexact operator, the values of q_i are analytically derived.

When it comes to an n-bit ESA divided into $r = \lceil \frac{n}{k} \rceil - 1$ sub-adders, since all the sub-adders have an equal size except the first sub-adder which is exact, the ER is approximately equal to $1-(\frac{1}{2})^r$. An approximation is then used to compute the mean ED, since the errors in the lower sub-adders can be neglected compared to the one of the higher sub-adders. A similar method is applied for the ETAII giving approximate values of the ER and mean ED. The proposed method strongly relies on a probabilistic analysis of the structures of each considered inexact adder, hence the impossibility to generalize this method to other structure of inexact operators or other AC techniques.

As an improvement of the method proposed in [LHL15],Wu et al. [WLGQ17] derived a method to compute the exact error profile of block-based inexact adders. Another improvement brought by Wu et al. is to provide a generic method to compute the error statistics of block-based adders. Making the assumption that the inputs are uniformely distributed in $[0; 2^N - 1]$ where N is the size of the adder, the authors compute the

probabilities of the signals propagate, generate and kill the carry, p, g and k respectively. The signals p, g and k are defined hereafter, if A, B are the input bits:

$$p = A \oplus B \tag{3.11}$$
$$g = A \cdot B \tag{3.12}$$
$$k = \overline{A} \cdot \overline{B} \tag{3.13}$$

Given these probabilites, the computation of the ER is possible. Finally, a result of the inexact aritmetic adder is correct if and only if all the speculated input of carries are correct. The authors compute in a recursive way the probability of this event. To derive the error distribution, the binary representation of the ED, named the "error pattern" is analyzed. All the possible error patterns are enumerated and their probability of occurrence is computed, giving the exact PMF of the error induced by the inexact adder.

Mazahir et al. [MHH+17, MHHS17] proposed a complete study on a probabilistic evaluation of the exact PMF of inexact adders and inexact recursive multipliers. The targeted class of inexact adders is adders implementing carry chain truncation and carry prediction between successive accurate sub-adders. An error occurs in these adders when the number of bits to predict the carry is not sufficient to predict the accurate carry signal. In this case, an error in a sub-adder can propagate to the upper sub-adders, and leads to an output of the adder lower than the accurate adder output. In the end, the method analyzes the probabilities that an error occur in each sub-adder to derive the accurate PMF. The method trades complexity off for genericity, not only targetting block-based adders. Nevertheless, this method is particularly long to analyze large bit-width adders. Again, the conditions on the inputs that led to an error are identified and treated as independent events using probabilities.

Matrix-based determination of the mean ED

Roy and Dhar [RD18] extend the method proposed in [MHH+17], deriving the accurate value of the mean ED of inexact Least Significant Bits (LSB) adders. This method is based on the structure of these n-bit adders decomposed into an $n-m$-bit accurate adder on the Most Significant Bits (MSBs) and several inexact sub-adders on the LSBs. The analysis of the mean ED is done by building a 2-D memory database of size $(M, 2)$, where $M = 2^{m+1}$ if m LSBs are approximated. To build this database, 4 matrices are used, which consider 4 different carry-out conditions on the m^{th} bit:

- MAT_{00}: the accurate and approximate carry-out signals at the m^{th} bit are 0.
- MAT_{01}: the accurate carry-out signal at the m^{th} bit is 0, and the approximate carry-out signal at the m^{th} bit 1.
- MAT_{10}: the accurate carry-out signal at the m^{th} bit is 1, and the approximate carry-out signal at the m^{th} bit 0.
- MAT_{11}: the accurate and approximate carry-out signals at the m^{th} bit are 1.

Given the truth tables of accurate and inexact adders, the 4 matrices storing the different error amplitudes for each sub-adder are built to finally compute the mean ED. The asymptotic runtime of the proposed matrix-based method for mean ED computation is linear with the number of approximated LSBs, $\mathcal{O}(m)$, if the size of the inexact sub-adders is negligible compared to m. Nevertheless, the proposed method targets only the estimation of the mean ED which is not enough to characterize the error generated by an inexact operator, and is particular to a class of operator.

Empirical model and gate-level error characterization

Sengupta et al. [SSHS17] proposed a gate-level error characterization method to determine the error variance e_{rms} of an adder whose approximation relies on the LSBs. The error variance is characterized as a function of the number of approximated LSBs y. The induced error e can be modeled as a random variable x that lies in $[-(2^y-1);(2^y-1)]$ since y LSBs are approximated, and whose probability to be equal to e is p_e. The error variance is then computed as:

$$e^2_{rms}(y) = \sum_{x=-(2^y-1)}^{2^y-1} x^2 p_x \tag{3.14}$$

Assuming that x is uniformely distributed in $[-(2^y-1);(2^y-1)]$, the error variance can be written as:

$$e^2_{rms}(y) = \frac{(2^{y+1}-1)^2}{12} \tag{3.15}$$

Finally, since the error variance when no LSBs are approximated is 0, an empirical formulation of the error variance is derived as $e^2_{rms}(y) = a \cdot (2^{by}-1)$. The values (a,b) are constants derived from fitting experimentally obtained error variance values with Monte-Carlo simulations to the empirical model.

Hierarchical analysis

Sengupta et al [SSHS18] proposed to derive the error PMF at the output of inexact adders or multipliers by first focusing on the characterization of smaller units, for instance Full Adder (FA).

For a signed approximate FA, the error can affects the output of the sum ΔS, as well as the sign bit Δs. For instance, for the inexact FA structure presented in [GMRR13], the truth table of the FA with inputs a, b, carry-in signal c_i, carry-out signal c_o and sum S is detailed in Table 3.3. The accurate values of S and s are indicated as S_{acc} and s_{acc} allowing computing ΔS and Δs.

a	b	c_i	c_o	s	s_{acc}	Δs	S	S_{acc}	ΔS
0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	1	0	1	1	0
0	1	0	1	0	1	-1	2	1	1
0	1	1	1	0	0	0	2	2	0
1	0	0	0	0	1	-1	0	1	-1
1	0	1	1	0	0	0	2	2	0
1	1	0	1	0	0	0	2	2	0
1	1	1	1	1	1	0	3	3	0

Table 3.3 – *Truth table of FA from [GMRR13].*

From the truth table of the FA, and for a known input distribution, in this example, the inputs are independent and uniformely distributed random variables. Consequently, the derivation of the PMF is immediate. The PMF of three signals are presented in Figure 3.8. Indeed, to derive the PMF for uniformely distributed inputs, the occurrence of the different signal values have to be counted. For instance, to derive $f_{\Delta S(n)}$, we can

observe in the truth table for signal ΔS, 6 occurrences out of 8 of value 0, and 1 occurrence out of 8 of values 1 and -1.

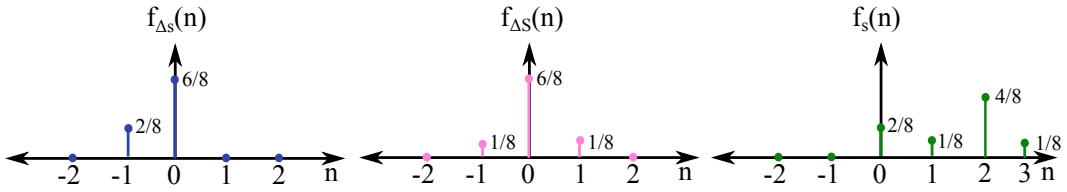


Figure 3.8 – *PMF of the signals Δs, ΔS and S for the approximate FA from [GMRR13].*

The derivation of the error PMF is done in a general case where the input distribution is not uniform. Nevertheless, the obtained expressions for the PMF depends on the probabilities of the input signals to be 1, which are known only when the input distribution is known. This leads to a highly restrictive condition to apply the proposed method to derive the error PMF of an inexact adder.

Error analysis in voltage-overscaled circuits

Liu et al. [LZP10] proposed an analytical analysis of the impact of supply voltage overscaling on arithmetic units, as adders or multipliers. The proposed method estimates the mean ED at the output of an exact operator subject to voltage overscaling. The authors' objective was to generalize a method to derive the mean ED under voltage overscaling since the behavior of three different 16-bit accurate adders, the Ripple-Carry-Adder (RCA), Carry-Look-Ahead Adder (CLA) and Carry-Select Adder with similar critical supply voltage, was very different under similar voltage overscaling operation as presented in Figure 3.9. The represented power consumption is normalized with respect to the power consumption at which the adders are accurate.

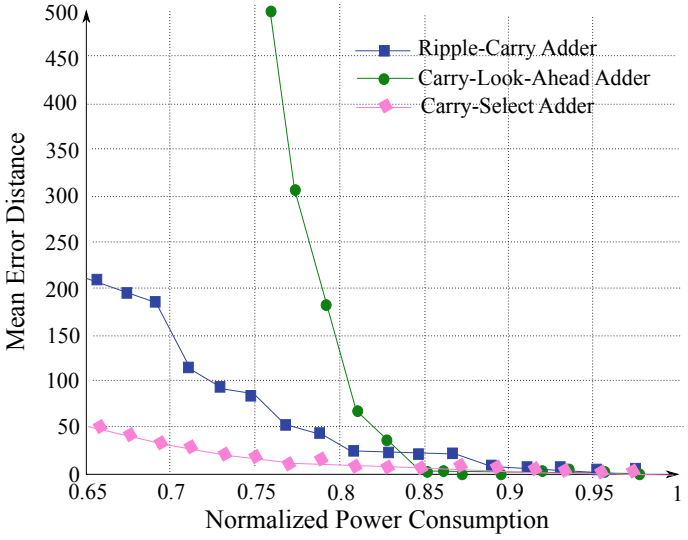


Figure 3.9 – *Impact of voltage overscaling on the mean ED at the output of accurate adders from [LZP10].*

To compute the mean ED at the output of an arithmetic operator subject to voltage overscaling, for each internal signal of the application, the error significance W_k^e, which depends on the maximum error amplitude as well as on the switching activity, is determined for each internal signal k. If the considered internal signal k impacts the computation of a single output bit, W_k^e is equal to the weight of the output bit (if the output bit is the i^{th} bit, the weight is equal to 2^i). If the considered signal k impacts several output bits, then W_k^e is the minimum of the weights of the output bits. Then, the switching activity is estimated in each node of the circuit and at all discrete time points. The switching activity corresponds to the probability of a transition from a bit a to b, with $(a, b) \in \{0; 1\}$. For a transition $0 \rightarrow 1$ of signal k at time $t = T_{CLK}$, the switching activity is denoted P_k^{01}, and for a transition $1 \rightarrow 0$, P_k^{10}. Given these information, the hypothesis that the different signals independently and additively contribute to the computation of the mean ED gives the following Equation for the mean ED computation:

$$\mu_e = \sum_{k \in \mathcal{S}} (W_k^e \times (P_k^{01} + P_k^{10})) \tag{3.16}$$

Indeed, with overscaled supply voltage, the critical path delay may be larger than the clock period T_{CLK}. Consequently, at time $t = T_{CLK}$, switching may induce errors. The proposed method has been demonstrated on the error analysis of circuits implementing digital signal processing applications subject to supply voltage overscaling. Giving an accurate analysis of the mean ED at the output of the circuit, it reduces the characterization time by several orders of magnitude compared to classical Monte-Carlo simulation techniques. The complexity of the proposed characterization is in $[\mathcal{O}(N); \mathcal{O}(N^m)]$ for the characterization of $N-$bit operations with m inputs.

Conclusion

Numerous methods have been proposed to characterize the errors induced by inexact operators as in [LHL15, MHH+17, WLGQ17, RD18] or to quantify the error induced by finite precision. A few work has been done to analytically evaluate the error induced by DVFS on an arithmetic circuit.

Focusing on the numerous methods proposed for inexact operators, their major drawback is that they all are dedicated to specific structures of inexact operators. If the application designer is willing to test inexact operators belonging to different types, the analytical method to compute the error statistics requires a new mathematical derivation. Besides, to compute the desired metrics, the amount of computations to do becomes really high with the bit-width and an important memory storage is required. To end with, no estimation has been proposed on the maximum ED, which is a critical parameter when implementing an approximation in an application.

3.4.2 Simulation-based techniques

To characterize the errors induced by AC techniques, simulation-based techniques are massively used. Simulation-based techniques are more and more employed due to their ease of use. Functional simulation techniques run the approximate system on a representative input data set and compute the required statistics for computing the error metrics. To mimic the behavior of the approximation, emulation techniques can be used.

The principle of functional simulation-based techniques for error metric characterization is presented in Figure 3.10. Functional simulation techniques can be used to link the approximation with an error metric. In this case, the approximate application and its

accurate version are run on $N_{Samples}$ points extracted from real input data. The accurate and approximate output values, z and $\widehat{z}$ respectively, are used to measure the obtained error according to a chosen metric, for instance the mean ED μ, the standard deviation of the error σ or the error PMF.

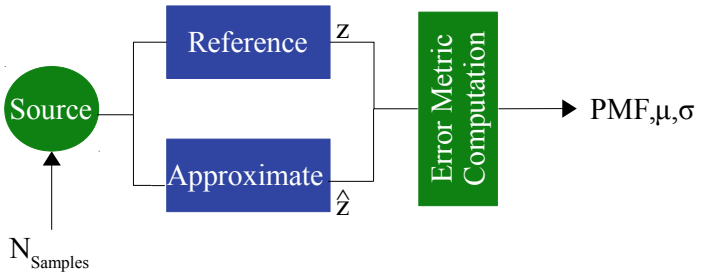


Figure 3.10 – *Illustration of the simulation-based determination of error metric.*

Exhaustive simulations

Exhaustive functional simulations can be used to compute exact statistics of the error induced by an AC technique. In this case, the AC technique is simulated for its whole input data set, and statistics on the induced error are computed.

For instance, an inexact operator can be simulated exhaustively which means simulated for all possible inputs. If the considered inexact operator has two unsigned inputs x and y coded on N_x bits and N_y bits respectively, the exhaustive input set $\mathcal{I} = \mathcal{I}_x \times \mathcal{I}_y$ is composed of 2^{Nx+Ny} values. Consequently, exhaustive functional simulations for high bit-widths inexact operators, and more generally if the input design space of an AC technique is large, are not feasible because of the required simulation time. Besides, as presented in Section 3.3.1, the emulation of the approximation mechanisms at the hardware level is complex, and long to simulate. To mimic the inexact operator behavior, bit-accurate simulations at the logic-level are required to model the internal structure modifications of the operator. Nevertheless, BALL simulations are two to three orders of magnitude more complex than classical simulations with native data types for 16-bit operators. This simulation time overhead can reach 7 orders of magnitude for complex 32-bit inexact operators. Thus, exhaustively testing the operator for all the input value combinations is not feasible for high bit-widths because of the required simulation time.

Mazahir et al. [MHH+17] have exhaustively simulated the inexact adder proposed in [SAHH15] on an Intel Core i7 processor working at 2.4 GHz for various input operands word-length. The evolution of the simulation time depending on the input operands word-length is presented in Figure 3.11.

As presented in Figure 3.11, the simulation time for inexact operators becomes prohibitive with the input operands word-length, since having an exponential evolution with the input operands word-length. Given this important simulation time overhead, exhaustive simulation, which means simulating the application implementing inexact operators for all its possible inputs, is impossible in most cases.

Monte-Carlo simulations

Functional simulation is commonly applied on a given number of random inputs, as presented for inexact operators in [DVM12, JLL+17, CCSE18] or for DVFS in [LZP10].

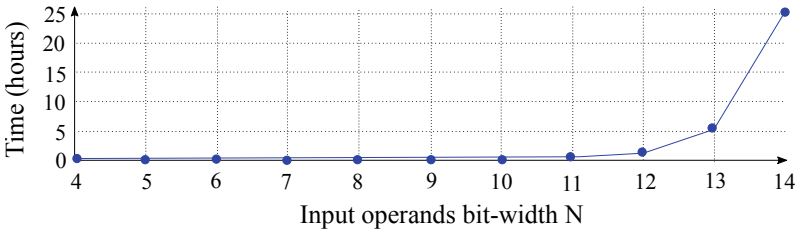


Figure 3.11 – *Simulation time of an inexact adder depending on input operands bit-width from [MHH+17].*

Inexact operators are generally simulated with 5 million random inputs as proposed in [CCSE18], which is the typical inexact circuit characterization method. The quality of the statistical characterization obtained from a random sampling is highly dependent on the number of samples taken and on the chosen input distribution. Besides, classical simulation-based analysis do not provide any confidence information on the obtained statistical estimation. Using a great number of samples can be ineffective in terms of simulation time.

Another drawback of Monte-Carlo simulations compared to analytical techniques is that they do not give any information on the causes of the induced errors.

Pre-characterization for analytical techniques

Noteworthy, a large part of the analytical techniques to characterize error metrics presented in Section 3.4.1 relies on a pre-characterization phase. The pre-characterization phase generally relies on simulations to get error information required for the analytical derivation of the error. For instance, before using analytical techniques as the MIA or Modified Affine Arithmetic (MAA) to propagate the errors through an application, Huang et al. [HLR11, HLR12] launch a characterization phase based on simulations. This characterization phase is required to derive the PMF of the error-free input data, the PMF of the error generated by the inexact operator, and the PMF of the error on the input data if the input is noisy. Once the different PMFs have been derived, they are stored in Look-Up Tables (LUTs).

Similarly, Chan et al. [CKK+13] need to characterize the behavior of different error metrics as the ER, mean ED or maximum ED depending on various input distributions and various hardware configurations. They first simulate the considered inexact operator for various hardware configurations, for instance different carry-chain length for the ACA. They then record the evolution of different error metrics depending on the standard deviation of the input distribution. The obtained results are saved in LUTs and further used for error composition.

Sengupta et al [SSHS17] used a pre-characterization phase with gate level characterization of inexact adders depending on the number of approximated LSBs. As presented in Section 3.4.1, the error pre-characterization step for inexact adders consists in fitting the parameters (a, b) in Equation $e^2_{rms}(y) = a \cdot (2^{by} - 1)$ giving the standard deviation of the error depending on the number of approximated LSBs y. To do so, the standard deviation of the error is computed for different values of y with exhaustive simulations, and (a, b) are derived by regression.

Conclusion

To analyze the errors induced by AC techniques, simulation-based techniques are hardly scalable for large applications, large input data sets or numerous and different perturbation types. Since exhaustive simulations are not scalable, Monte-Carlo simulations are generally used. Nevertheless, a study on the number of samples to simulate and on the quality of the obtained statistics has, to the best of our knowledge, never been proposed. Work on simulation-based techniques was to be done to develop generic and more robust error analysis methods.

3.5 Quality metric determination

3.5.1 Analytical techniques

Analytical techniques derive a mathematical expression of the application quality metric. For instance, Liu et al. [LHL15] proposed to analyze the impact of inexact adders on an image processing application, and particularly to analytically derive the Peak Signal to Noise Ration (PSNR) metric at the output of an image processing application implementing specific types of inexact arithmetic adders. When implementing this analytical technique, the error metric required from Block 1 in Figure 3.1 is the mean ED.

The PSNR is a commonly used quality metric or intermediate accuracy metric for image processing applications. The computation of the PSNR is done as:

$$\text{PSNR} = 20 \log \frac{d}{\sqrt{\text{MSE}(I,K)}} \tag{3.17}$$

where d represents the maximum possible value for a pixel in the reference image I of size (m,n), K represents the image of size (m,n) obtained from approximate application implementation, and MSE is the mean squared error function defined as:

$$\text{MSE}(I,K) = \frac{1}{m \cdot n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \tag{3.18}$$

which can be rewritten as:

$$\text{MSE}(I,K) = E[M_E \circ M_E] \tag{3.19}$$

where M_E is the error matrix defined as $M_E = K - I$ and E represents the expected value of the Hadamard product of M_E with itself. Since the implemented inexact circuit is composed by a inexact adders, Equation 3.19 can also be written as:

$$\text{MSE}(I,K) = E[(\sum_{i=1}^{a} M_{E,i}) \circ (\sum_{i=1}^{a} M_{E,i})] \tag{3.20}$$

Two strong hypotheses are made: the inputs are uniformely distributed, the mean ED μ_e and Mean Squared Error Distance (MSED) e_{rms} of each inexact adder i are similar to the mean ED and MSED of the error matrix $M_{E,i}$, which leads to the following approximation to compute the MSE(I,K):

$$\text{MSE}(I,K) \approx a \cdot e_{rms} + a \cdot (a-1) \cdot \mu_e^2 \tag{3.21}$$

For particular inexact adders, the authors derived the values of e_{rms} depending on μ_e, leading to an expression of the PSNR solely depending on the mean ED of the inexact

adders. Nevertheless this analytical framework is specific to particular inexact adder types and specific to the considered application quality metric. To end with, the input distribution is generally not known which leads to question the hypotheses taken for this analytical development.

The drawback of analytical techniques is their specificity to the considered application as well as to the application quality metric, which makes them not automatic. Being more generic and easier to automate, simulation-based techniques are often preferred to analytical techniques to evaluate the quality metric.

3.5.2 Simulation-based techniques

Direct quality metric determination

Application quality metric can be directly evaluated at the output of the application through simulations. In this case, the application has to be simulated on a representative input data set and the quality metric is measured at the output. Nevertheless, some quality function evaluation are very difficult to evaluate directly. For instance the BER in digital communication systems, is long to directly evaluate because of the required evaluation time. When a BER at receiver output of 10^{-k} is targeted, 10^{2+k} simulations are required for a good quality evaluation of the application. This metric is really long to evaluate when introducing approximation in an application, which leads to the need to separate the quality metric determination process in two steps. First, the impact of the approximation on a intermediate quality metric is evaluated and then the link between the intermediate accuracy metric and the application quality metric is established.

Error injection

To analyze the impact of an AC technique on the quality metric, the errors have to be emulated within the application. Error emulation can be done at the bit level for finite precision, or at the logic level for inexact operators. When error emulation is not directly possible, perturbation-based methods can be implemented. In this case, errors are directly injected in the application, as for instance proposed in the framework REACT [Wys] with dynamic error injection that extends the ACCEPT framework [SBR+15]. To use the REACT framework for error injection, the code has to be annotated with the approximable parts and the injected errors are extracted from a pre-built library of several AC techniques, provided by the user. Errors can be injected at two granularity levels. At the fine grain level, errors are injected in the instructions, while at the coarse grain level, errors are injected at the output of functions.

The error injection technique is widely used to derive the relationship between the intermediate accuracy metric and the application quality metric. Chippa et al. [CCRR13] propose the ARC framework to analyze the sensitivity of the different parts of an application in order to identify the error-resilient parts. In this case, the intermediate accuracy metrics are the error amplitude and the error rate. In the context of fixed-point refinement, Parashar et al. [PRMS10] proposed to deal with the finite precision conversion of a multi-kernels system by modeling the behavior of each kernel converted in fixed-point by a single noise source located at the output of the kernel. In this case, the intermediate accuracy metric is the noise power. This technique based on error-injection allows finding the different noise power values at the output of each kernel subject to a quality constraint at the output of the application. An optimization problem is formulated to budget the

noise power on each kernel such that the quality constraint at the output of the application is satisfied. A steepest descent greedy algorithm is used to solve this problem.

3.5.3 Conclusion

Analytical techniques have been proposed to determine the quality metric at the output of an application but are often giving approximate values of the quality metric at the output of an application. Simulation-based techniques can lead to very accurate estimation of the quality metric at the output of an application but the accuracy depends on the number of input samples simulated for the quality metric measurement. Besides, simulation-based techniques are more generic for the determination of the quality metric without relying on the type of implemented approximation. In the contributions of this thesis, simulation-based techniques have been used. However, contrary to the previously proposed approaches, the impact of the number of input samples on the accuracy of estimation has been studied and measured to avoid simulating large data sets. Moreover, simulation-based techniques are long to evaluate the QoS at the output of an application. In this thesis as well as in state-of-the-art techniques, the evaluation of intermediate accuracy metrics can be preferred to the evaluation of the QoS.

3.6 Accuracy metric determination

As presented for the characterization of error metrics, two types of state-of-the-art approaches can be used to evaluate the accuracy metric: analytical and simulation-based approaches. These techniques allow propagating errors within an application.

3.6.1 Analytical techniques

Analytical methods mathematically express the error characteristics at the output of the application.

Interval/Affine arithmetic

In IA, an interval is assigned to each internal variable of the application. The interval is then propagated within the different computations according to arithmetic rules. Let's define $[\underline{x};\overline{x}] = \{x|\underline{x} \leq x \leq \overline{x}\}\}$, with $\underline{x}$ and $\overline{x}$ the minimum and maximum values of a variable x respectively. IA is particularly suited to simple operations, and the intervals can be propagated from inputs to outputs through basic arithmetic operations as additions, substractions, multiplications and divisions represented by the $\diamond$ operator in Equation 3.22, and if the system is non-recursive. In this case, the non-recursivity means that a variable in the system does not depend on its previous values as it is the case in Infinite Impulse Response (IIR) filters. The general propagation rule is:

$$[\underline{x};\overline{x}] \diamond [\underline{y};\overline{y}] = [\min(\underline{x} \diamond \underline{y}, \underline{x} \diamond \overline{y}, \overline{x} \diamond \underline{y}, \overline{x} \diamond \overline{y}); \max(\underline{x} \diamond \underline{y}, \underline{x} \diamond \overline{y}, \overline{x} \diamond \underline{y}, \overline{x} \diamond \overline{y})] \tag{3.22}$$

IA allows keeping track of measurement errors, errors caused by the inputs, and errors caused by inexact computations. The asset of using IA is its ease to compute but the produced error bounds are not tight and generally conservative and pessimistic. Indeed, IA does not take into account any correlation between the variables to be composed and is particularly pessimistic when variables are correlated. Numerous libraries have been proposed to directly compute with intervals, as the C++ library Boost [BMP06].

To improve the estimation of error bounds, Affine Arithmetic (AA) has been proposed by Stolfi et al. [DFS04] in the 1990's. The variables are no longer modeled with intervals but with affine forms as presented in Equation 3.23. AA improves the tightness of the error bounds by taking into account the first-order correlations between the variables to be composed. In Equation 3.23, x_0 is the central value of variable x, x_i the partial deviations and e_i the error terms in $[-1;1]$. Rules have been proposed to compose different affine forms together and are presented in Equations 3.24-3.27, where $\hat{x}, \hat{y}$ are affine forms and c is a constant. As shown for the composition of two affine forms by a multiplication, which is not an affine operation, a residual error symbol is produced as $rad(\hat{x})rad(\hat{y})$, where rad corresponds to the radius of the affine form. As proposed for IA, numerous libraries have been proposed to compute with affine forms, as the C++ library LibAffa [GCH06].

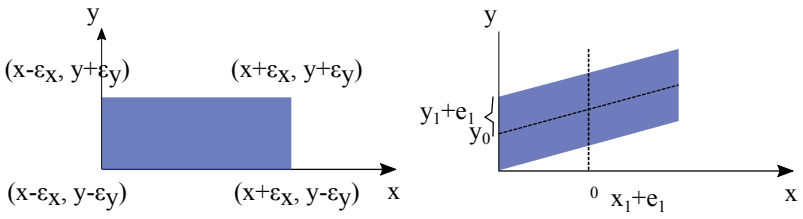
$$\hat{x} = x_0 + \sum_{i=1}^{n} x_i \times e_i \tag{3.23}$$

$$c \times \hat{x} = c \times x_0 + c\sum_{i=1}^{n} x_i \times e_i \tag{3.24}$$

$$c \pm \hat{x} = (c \pm x_0) + \sum_{i=1}^{n} x_i \times e_i \tag{3.25}$$

$$\hat{x} \pm \hat{y} = (x_0 \pm y_0) \pm \sum_{i=1}^{n} (x_i \pm y_i) \times e_i \tag{3.26}$$

$$\hat{x} \times \hat{y} = x_0 y_0 + \sum_{i=1}^{n} (x_i y_0 + y_i x_0) \times e_i + rad(\hat{x})rad(\hat{y}) \tag{3.27}$$



(a) Representation of IA. (b) Representation of AA.

Figure 3.12 – *Comparison of IA and AA in terms of tightness of the bounds.*

As presented in Figure 3.12 with the area covered by both IA and AA, AA improves the tightness of the error bounds by considering the first-order correlations of error signals through the sharing of the error terms e_i, but to the detriment of more complex computations. It should be noted though that both types of arithmetic ensure guaranteed error bounds. To demonstrate the pessimistic error bounds obtained with IA compared to AA, the evaluation of $\hat{z} = \hat{x} + \hat{y}$ with both types of arithmetic and given several information is done. A relationship between $\hat{x}$ and $\hat{y}$ is known since $\hat{y} = -\hat{x}$ as well as their belonging intervals since $\hat{x} \in [-1;1]$ and $\hat{y} \in [-1;1]$.

Using IA, the interval of $\hat{z}$ is computed as:

$$\hat{z} = [-1;1] + [-1;1] = \boxed{[-2;2]} \tag{3.28}$$

While with AA the obtained output for $\hat{z}$ takes into account the correlation between both input variables and is more realistic and less pessimistic, as presented in Equation 3.29.

$$\hat{z} = 0 + 1 \cdot \epsilon + 0 - 1 \cdot \epsilon = \boxed{0} \tag{3.29}$$

To sum up, a comparison of both types of arithmetic is proposed in Table 3.4.

	Assets	Drawbacks
IA	Numerically stable	Unefficient for interval products Linear convergence
AA	More accurate for interval products Keep track of correlation	Less stable Higher computational cost

Table 3.4 – *Comparison between IA and AA.*

Both techniques are well adapted to represent symmetric distributions, for instance the errors induced by fixed-point arithmetic [FRC03]. Caffarena et al. [CLFHC10] proposed a method based on AA to estimate the Signal-to-Quantization-Noise Ratio (SQNR) at the output of a digital signal processing application converted in fixed-point and to target non-linear algorithms.

Highly asymmetric distributions such as the errors produced by inexact arithmetic operators are not well represented by either IA or AA. To better render their asymmetric error distributions, modifed interval and affine arithmetics have been proposed.

Modified interval/affine arithmetic

Based on the error propagation method proposed with IA or AA, Huang et al. [HLR11, HLR12] proposed an adaptation of these methods to inexact circuits, more adapted to the highly asymmetric PMFs representing the errors induced by inexact operators as presented in Figure 3.3. Indeed, IA and AA need to be centered and consequently fail to represent errors of inexact operators. When implementing MIA or MAA, the error metric required from Block 1 in Figure 3.1 is the PMF of the inexact operator.

In [HLR11], MIA and MAA are proposed to represent asymmetric distributions. The proposed method allows statically estimating the impact of errors induced by inexact operators on the application quality metric. In MIA or MAA, the whole distribution is decomposed into multiple intervals/affine forms. In the case of MIA, each bar of the PMF of an inexact operator is modeled by an interval as in Equations 3.30 and 3.31. In Equation 3.31, the variable n represents the error magnitude.

$$\text{MIA}(x) = P(a \leq x \leq b), \text{ if } a \leq x \leq b \tag{3.30}$$

$$f_X(n) = \begin{cases} P(2^{\epsilon+n-1} \leq X \leq 2^{\epsilon+n}) & \text{if } n > 0 \\ P(-2^{\epsilon+n+3} \leq X \leq -2^{\epsilon+n+2}) & \text{if } n < -1 \\ P(0 \leq X \leq 2^{\epsilon}) & \text{if } n = 0 \\ P(-2^{\epsilon} \leq X \leq 0) & \text{if } n = -1 \end{cases} \tag{3.31}$$

To compute the total error probability the function MIA(x) has to be integrated. The rules to compose different intervals are similar to IA but in MIA, each bar of the first PMF has to be composed with each bar of the second. MIA can be used to propagate the error induced by inexact circuits through simple blocks. To do so, rules are proposed by Huang et al. [HLR12]. Nevertheless, MIA is still very pessimistic on the error bounds and suffers from range explosion.

In the case of MAA, each bar of the PMF of an inexact operator is modeled by an affine form as in Equation 3.32.

$$f_X = \begin{cases} P_1 : x_{1,0} + x_{1,1}\alpha_0 + x_{1,2}\beta_0 + ... \\ P_2 : x_{2,0} + x_{2,1}\alpha_1 + x_{2,2}\beta_1 + ... \\ P_3 : x_{3,0} + x_{3,1}\alpha_2 + x_{3,2}\beta_2 + ... \end{cases} \tag{3.32}$$

The problem of range explosion is tackled since MAA takes into account the first order correlation between variables. However, when dealing with operations such as multiplications or divisions, the output form is approximated to an affine form. Consequently, the output of multiplications or divisions is not guaranteed to be more optimistic than the result obtained with MIA. In addition to this problem of range explosion, MAA can suffer from storage explosion due to the operation of several affine forms which can result in the storage of numerous additional terms. The solution to this problem is to group some terms of the PMF but then reducing the output accuracy. All these constraints pushed us to develop another way to simulate quickly and without need for additional storage, large bit-width inexact operators.

To propagate the error PMF through the inexact circuit, the PMF of the errors induced by inexact operators have to be characterized. To do so, Huang et al. proposed exhaustive simulations to derive the PMF and needed to store them for further computations. The inputs of a circuit are represented by random variables since they can be affected by external noise, and are consequently characterized by PMFs. The PMF of the error-free input has to be propagated through the whole system then, the table storing the PMFs information has to be accessed to know the error PMF of every approximate operator and is finally propagated through the whole system. Nevertheless, if the range of input signals increases, the runtime to execute the proposed method becomes prohibitive. One of the main drawbacks of this method is that recursive systems as an IIR filter cannot be directly handled by these techniques, since the recursion must be unrolled. Besides, the proposed technique can be used to precisely estimate the error rate in the system but the estimation of the maximum error amplitude is overly pessimistic and may lead to a wrong rejection of the proposed approximation technique. The application of such techniques in complex applications is then questionned.

Accuracy metric determination at the output of a Directed Acyclic Graph (DAG)

Sengupta et al. [SSHS17] proposed an analytical framework to characterize the variance of the error at the output of a DAG composed of inexact adders. Again, the proposed method for application quality metric determination is particular to the implementation of inexact adders. The circuit is represented as a DAG given as a dataflow graph. The nodes of the DAG are the potentially approximated arithmetic units and the edges are the connections between the different nodes of the circuit. The proposed framework, named SABER, has been proposed to explore the quality versus power savings trade-off to know how many bits should be approximated in each arithmetic operation.

In the proposed method, the approximation is supposed to be applied only on the LSBs. For each node of the DAG composed of a potentially inexact adder, the variance of the error is computed depending on the number y of LSBs approximated and on the error distribution. For instance, if the error is uniformly distributed, the variance is expressed as:

$$\sigma_e^2(y) = \frac{(2^{y+1} - 1)^2}{12} \tag{3.33}$$

The variance is more generally expressed as:

$$\sigma_e^2(y) = a \cdot (2^{b \cdot y} - 1) \tag{3.34}$$

To determine the coefficients (a, b) for each considered inexact adder, exhaustive simulations are used. This is the precharacterization phase, allowing characterizing the error variance at gate level of multi-bit adders, depending on y, the number of approximated LSBs. This phase allows building a library for several inexact adders.

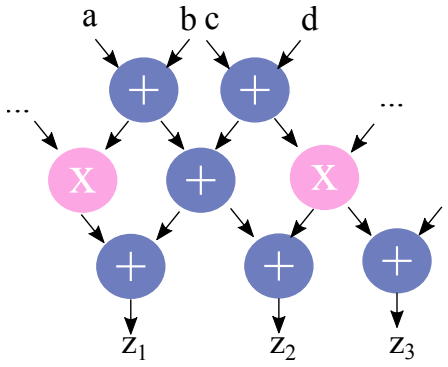


Figure 3.13 – *Example of a DAG composed of adders and multipliers.*

Once the variance at the output of each node of the DAG has been characterized, the error at each node of the DAG are considered as independent random variables. An example of a DAG is presented in Figure 3.13, where (a, b, c, d) are primary inputs of the DAG, and (z_1, z_2, z_3) primary outputs of the DAG. The presented DAG is composed of adders and multipliers, the multiply operation can indeed be decomposed as adders and shift operations. The variance at the output of each node in pink and blue is considered as known through the precharacterization phase.

The structural correlations between the different nodes of the DAG have still to be taken into account to compute the overall standard deviation of the error at the output of the DAG. Indeed, the error induced by approximation on a node impacts the nodes connected to it. The error variance at the output of the DAG is then expressed as a function of the number of approximated LSBs in each node of the DAG, and is also dependent on β_i, the error sensitivity of a node to a primary output. The obtained expression is a non linear expression detailed in Equation 3.35 for a DAG with T nodes. Variables y_i and β_i are the values y and β at node i. The values β_i are computed with a depth-first search of the DAG.

$$\sigma = \sum_{i=1}^{T} a \cdot (2^{b \cdot y_i} - 1) \cdot \beta_i \tag{3.35}$$

The major drawbacks of the proposed method are its restrictive assumption that the approximation relies only on the LSBs, which restricts the type of considered inexact adder, as well as the need to determine the relationship between the variance and the number of approximated LSBs that has to be done with time-consuming simulations.

Sengupta et al. [SSHS18] then proposed a more generic analytical approach based on the Fourier and Mellin transforms to evaluate the PMF of the error at the output of a circuit implementing inexact operators. Again, the considered inexact circuit is represented as a DAG.

The PMF of individual inexact operators as adders or multipliers is first derived, and if these operators are placed in a DAG, topological traversal is used to propagate the errors induced by the operators through the application. The proposed method derives an analytical expression of the PMF of the errors induced by inexact operators, from a Boolean function representing the approximation at the logical level.

As proposed by Huang et al. [HLR11], the proposed method is based on the PMF of the inputs of the circuit, in case they are tainted with errors, on the PMF of the output with no error and on the PMF of the output error. The inputs and outputs of the DAG are considered as random variables. Once the PMF of the individual nodes, adders, substractors or multipliers, are obtained, the error PMF has to be propagated through the DAG up to the output of the DAG. To compute the error PMF at the output of a node, the error generated at the node has to be composed with the error PMF induced by the input operands of the node. The different PMFs are modeled as sums of Dirac functions.

The propagation of the error PMF through an inexact adder is computed as the convolution of the PMF of the errors on the inputs and of the error induced by the inexact adder. The propagation through an inexact multiplier is more complex, since it is computed as the sum of the product of random variables. The product of the PMF of the random variables is computed with the Mellin transform [Spr79] and their sum if computed as for the propagation through the inexact adder, with a convolution of the obtained PMFs. The convolution operation is done with a Fourier transform.

To be able to use the Fourier and Mellin transforms, assumptions are required on the independent nature of the inputs treated as random variables, as well as on the independence between several full adders representing the global inexact operator. The asset of this method is its applicability to various inexact operators. However, the proposed method based on the Fourier and Mellin transforms has an exponential theoretical complexity. The gain in terms of time are not sufficient to answer the growing space to explore when implementing inexact operators in a complex application. This method has first been derived for inexact multipliers [SS15] where the multiplication process is decomposed into the generation of the partial product to which are assigned PMFs which are convoluted to give the PMF at the output of the whole multiplication. The proposed method offers a good quality on the computation of the error PMF as well as an improvement on the characterization speed of one order of magnitude compared to Monte-Carlo simulations. The major drawback is its exponential theoretical complexity.

Regression-based technique for output quality metric determination

Chan et al. [CKK+13] proposed a regression-based technique to compute the quality metric at the output of a circuit composed with inexact arithmetic operators. After having obtained the error metric characterization stored in a library for each arithmetic unit, the goal of the proposed method is to obtain a function f giving the error metric EM_{out} depending on the error metrics EM_i obtained at each arithmetic unit, such that:

$$\mathrm{EM}_{out} = f(\mathrm{EM}_1, \mathrm{EM}_2, ..., \mathrm{EM}_n) \tag{3.36}$$

with n the number of arithmetic units in the considered circuit. The proposed method does not assume that the inputs of the application are uniformely distributed. As presented in Figure 3.14, the required information to compute an error metric at the output of an arithmetic unit, are the standard deviation $e_{\mathrm{rms,a}}$ and $e_{\mathrm{rms,b}}$, generated by the previous unit that feed the considered unit, the error metric generated by the previous unit EM_a and EM_b, and the intrinsic error metric EM_i induced by the considered arithmetic unit.

The standard deviation of the different units are obtained with topological traversal and are stored in a Look-Up Table (LUT). A precharacterization allows getting the values EM_i for different standard deviation values.

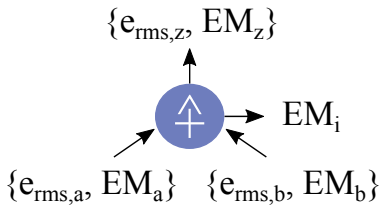


Figure 3.14 – *Example of the estimation of the error metric of an arithmetic unit.*

Finally, general composition rules have been proposed to propagate the error through a circuit composed of inexact adders. To adjust the composition rules depending on the considered structure, regression is used. This technique has been compared to MIA and for a similar runtime is 3.75× more accurate. For the error composition part, the proposed method achieves a runtime reduction of 8.4×. Nevertheless, this technique is only applicable to adder-based circuits.

Boolean analysis

Venkatesan et al. [VARR11] proposed a general methodology to analyze the quality at the output of a circuit subject to timing-induced approximation as DVFS or the implementation of inexact arithmetic operators. The first step of the proposed methodology is to generate an untimed version of the considered circuit, when subject to DVFS, that represents its behavior under DVFS. This step corresponds to the conversion of the timing-induced approximation into the functional domain. The indicated behavior of the circuit is particular to a given clock period and voltage. The critical paths of the circuit with delay larger than the clock period are analyzed to generate the untimed version of the circuit. Subject to timing-induced approximations, the output of a combinatorial circuit are dependent on the inputs at time t, as well as the previous at time $t-k, k \in \mathbb{N}$. The untimed version of the circuit functionnally exposes these dependencies. Then, the comparison between the accurate circuit implementation and its inexact version is done by generating a virtual error circuit. Finally, conventional verification tools as Boolean satisfiability solvers or binary decision diagrams are applied on the obtained virtual error circuit to generate the desired error metrics.

Fixed-point systems

Numerous analytical techniques have been proposed to propagate the different noise sources in a finite precision system and evaluate the noise power at the output of an application. Analytical methods mathematically express the noise power at the output of the application due to finite precision. The study of the impact of finite precision on the accuracy metric has firstly been developped for classical signal processing kernels. Liu et al. [Liu71] developped an analytical model for digital filters. The different noise sources in the filter are identified, as the quantization on the input signal, the quantization of the coefficients and finally the accumulation of rounding errors in the arithmetical operations. Finally, the analytical computation of the mean squared error at the output of the filter is analytically derived. Hilaire et al. [HMS08] have proposed an analytical framework to derive the round-

off noise at the output of filters and compare the noise values for different realizations of the filter.

The analytical study of fixed-point errors has then been separated for smooth and unsmooth applications (applications with non-linear operators). For smooth applications, the used method depends on whether the application is Linear-Time Invariant (LTI) or not. Menard and Sentieys [MS02] proposed an automatic analytical evaluation technique of the SQNR at the output of an application implementing non-recursive and linear recursive structures. Authors represent the application as Signal Flow Graph (SFG), listing all the information of fixed-point format. The SFG allows obtaining the different transfer functions whose frequency responses are then linked to the SQNR at the output of the application. The proposed method takes into account the different quantization modes, for instance rounding or truncation. This method is extended to non-LTI systems by Rocher et al. [RMSS07] using possibly time-varying transfer functions or impulse responses to model the application. The transfer function or impulse response of the application is linked to the different noise sources, given their mean and variance to give the output noise power.

Different approaches based on perturbation theory have been proposed to determine the analytical expression of the noise power at the system output. Each quantization leads to a noise source. The challenge is to determine the gain between each noise source and the output. A linear model is used for the noise propagation through the different operations thanks to a first order Taylor expansion of each operation. The gains can be computed from time-varying impulse responses [RMSS07], affine arithmetic analysis [LCCNT08] or simulation [LGC+06].

Nevertheless, analytical techniques are complex, hard to automate and their applicability is limited to systems having only smooth operations, that is to say differentiable operations.

3.6.2 Simulation-based techniques

When using simulation-based techniques, an intermediate accuracy metric is generally evaluated. Indeed, the direct evaluation of the quality metric at the output of an application may require numerous samples to be simulated. For instance, Huang et al. [HLR12] illustrated the example of functional simulation technique on a length-10 dot product with data formatted on 32-bit. To simulate the approximate application covering the whole input space, $32^{20} \leq 1.3 \times 10^{30}$ different input vectors would be simulated. In digital communication systems, to evaluate the BER metric, if it reaches 10^{-k}, 10^{2+k} samples have to be simulated.

For the evaluation of an intermediate accuracy metric as the noise power, the metric is generally computed by simulating an arbitrary, and large, number of random inputs N_S [KHWC98]. For fixed-point arithmetic, in the literature the number of samples ranges from 10^5 [KHWC98] to 10^{12} [KWCM98]. For determining the noise power P induced by finite precision, two different versions of the application are simulated as presented in Figure 3.10. The distance between the output of the application with infinite precision and the output of the application with finite precision is measured and squared for each simulated sample. The expected value of these distances is then computed to obtain the value of the noise power P at the output of the application. The slow software simulation of fixed-point data-types as well as the high number of samples to simulate makes generally fixed-point conversion a long and tedious task. Sedano et al. [SLC12] proposed to improve the speed of fixed-point conversion, to use inferential statistics to infer the number of

inputs to simulate. They derived that for a noise constraint of 10^{-k}, 10^{k+1} input points were required for a fair evaluation of the noise power.

The number of significant bits for finite precision arithmetic can be considered as an intermediate accuracy metric. In this context, discrete stochastic arithmetic has been proposed by Jezequel and Chesneaux [JC08] for floating-point and fixed-point arithmetic. In this method, for each data, the last significant bit is randomly perturbed and the application is run N times using the same input data for each run. The average over these N runs is used as an estimate of the exact value. A t-distribution is used to provide a confidence interval for the estimated value. The number of significant bits is deduced from the number of common bits between the obtained values and the estimation. The main advantage of this approach is the low number of runs N required to obtain a good estimation. In practice, $N = 3$ is used.

Finally, simulation-based technique are widely employed since they are not limited by the applicability of analytical techniques. However, due to the long emulation time of the approximations techniques, functional simulation techniques do not scale well with large applications.

3.6.3 Conclusion

Analytical techniques have majorly been proposed to characterize errors induced by approximations. These techniques are particular to the considered error metric and the considered approximation technique. Simulation-based techniques have the asset of being generic but do not scale with large systems. In this thesis, simulation-based techniques have been used to characterize errors induced by approximations using statistical models. Statistical models allow studying the number of simulations to lead so as to get an estimation of the error properties according to user-defined confidence requirements.

3.7 Conclusion

The error analysis is an important step of implementing AC, as well as analyzing the errors tolerated by the application. The errors due to approximations can be classified into three separate classes as presented in Figure 3.15.

1. The **fail small** category of errors corresponds to errors with a high probability of occurence but a small amplitude. Errors induced by finite precision and especially fixed-point arithmetic fall in this category.

2. The **fail moderate** category of errors is a hybrid category corresponding to errors with a moderate amplitude and probability of occurence.

3. The **fail rare** category of errors corresponds to errors with a low probability of occurence but a high amplitude. Errors induced by inexact operators fall in this category.

In this thesis, methods and tools have been proposed to quickly evaluate the effects of various approximation techniques on an application quality metric. The proposed contributions have tackled both the modelization of the error induced by approximations, and the analysis of the impact of errors on the application quality metric as presented in Figure 3.16. This thesis focuses on two different approximation techniques, namely the implementation of inexact operators, a hardware-level AC technique, and the conversion

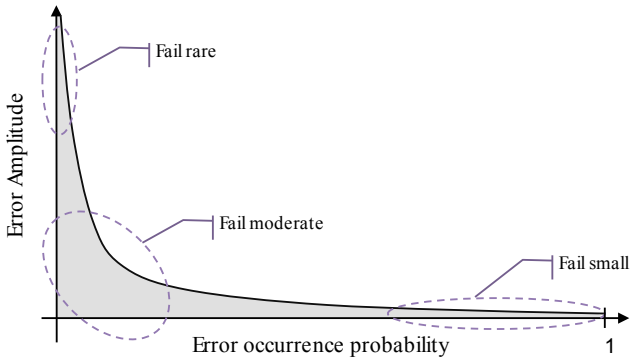


Figure 3.15 – *Different error categories in approximate computing: evolution of the error frequency depending on the error amplitude.*

to finite precision, which is a data-level AC technique. The first main contribution of this thesis has been to develop a methodology to quickly characterize the category of errors induced by an approximation technique. The statistical model of the approximation error has then be used to simulate and analyze the resilience of a given application to an approximation technique.

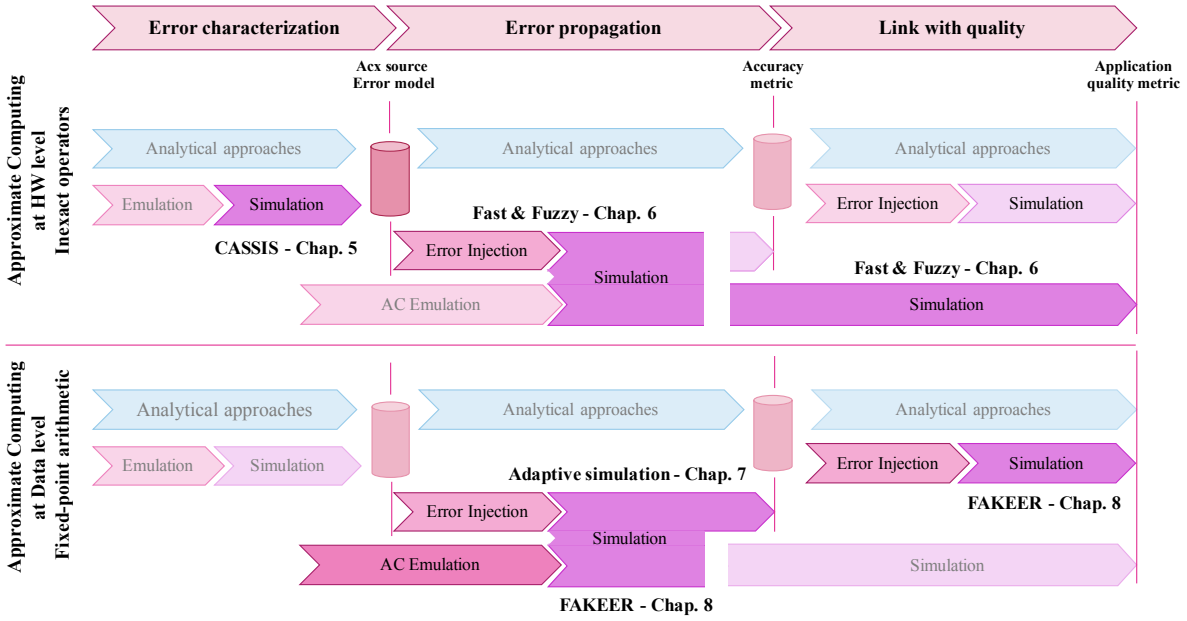


Figure 3.16 – *Contributions of the thesis.*

As presented in Figure 3.16, the error profile has been characterized for two different levels of approximation: at hardware level with inexact operators and at data level with finite precision arithmetic. The characterization of the errors induced by inexact operators has been proposed with simulations in Chapter 5. The characterization of the error profile is done in terms of error amplitude and error occurence. For fixed-point arithmetic, a model for a commonly used intermediate accuracy metric is proposed in Chapter 7. For both inexact operators and fixed-point arithmetic, the proposed characterization of the error profile is done using adaptive simulations depending on confidence requirements on the characterization.

Then, for the error propagation step, a stochastic modelization of the errors induced by inexact operators has been proposed, and can also be applied to fixed-point arithmetic. The proposed stochastic modelization allows building a fast functional simulator, the Fast and Fuzzy (FnF) simulator, mainly useful for inexact operators and presented in Chapter 6. The simulator has been built to take data as inputs, and output data so as to be able to measure the impact of the approximation on the application quality metric. The characterization method for fixed-point arithmetic has been used to build a framework to process the Approximation Design Space Exploration (ADSE) when converting an algorithm to fixed-point arithmetic described in Chapter 7. To end with, FAKEER, an estimation method has been proposed to avoid simulation for the characterization of the error metric for finite precision arithmetic. This contribution is detailed in Chapter 8.

Part II

Contributions

CHAPTER 4

Motivations for Approximation Error Modeling
Algorithm-level Approximation for Embedded Stereovision Algorithm

"Il y a de l'or dans l'erreur"

Matthieu Chédid

The first contribution of this thesis, after a non-exhaustive presentation of the existing Approximate Computing (AC) techniques in Chapter 2 as well as the existing methods to quantify and analyze the impact of approximations on the application quality of service (QoS) in Chapter 3, is an AC technique at the computation level applied to a stereovision algorithm. The majority of the contributions of this thesis are focused on the characterization of errors induced by AC techniques at the data or at the hardware level. This work allows studying an AC technique at the computation level and has led to an interesting comparison of the results obtained with two different QoS metrics, which motivated the need for generic methods to evaluate the errors induced by AC. This contribution has been presented at the International Conference on Embedded Computer Systems Architecture MOdeling and Simulation SAMOS 2018 [BDM18b]. The contributions brought by the proposed work are as follows:

- Presentation of the algorithm-level approximation technique
- Comparison of the theoretical results on the complexity reduction with the obtained experimental results according to two different QoS metrics

The rest of this chapter is organized as follows: Section 4.1 introduces the stereo matching algorithm and the related works. Section 4.2 presents the proposed approximation method for computing the depth map, which is the output of the considered stereovision algorithm. Then, the results are exposed in Section 4.3 for both metrics, followed by a discussion on the need for efficient and generic quality metrics before implementing an AC technique in an application.

4.1 Introduction and Related Works

4.1.1 Introduction

Most embedded applications incorporate data-oriented processing with sophisticated mathematical computations. The complexity of embedded applications is increasing even though the energy constraints they must fulfill are more and more drastic. The design challenge is to provide a real-time implementation of these applications without sacrificing energy consumption. Currently, this challenge is answered with dedicated hardware presenting two major drawbacks: they are costly to produce and can hardly be reprogrammable. In this chapter, the considered application to embed is a computer vision application, the stereo matching algorithm that can be used for Advanced Driver-Assistance Systems (ADAS), for 3D reconstruction, or for replacing the Kinect active device [KE12] that uses an infra-red grid emitted on the observed scene and is consequently limited to an indoor use within a 5 meters range. The choice of this application has been done since the targeted stereo matching algorithm is built with basic blocks massively used in computer vision algorithms as in [ABBB13].

The stereo matching algorithm is used to extract a 3D information from two 2D images taken by two rectified cameras spaced by a small distance. This 3D information is represented in a depth map, also called disparity map. The disparity map, whose size is identical to the size of the input images, represents the horizontal distance between the position of a pixel in the first image, and its position in the second image. The stereo matching algorithm has been mostly designed and studied on Desktop Graphics Processing Units (GPUs) which consume several hundred watts. Consequently, work is required to optimize it for embedded systems. The high complexity of state-of-the-art stereo matching algorithms limits their embeddability. Indeed, if no special care is taken during the prototyping phase of an algorithm, the high precision of the produced results is often obtained at the expense of high latency, memory storage or energy consumption. However, embedded systems are interesting targets for computer vision applications. Instead of using dedicated hardware to embed the stereo matching algorithm, AC techniques can be implemented to reduce the energy consumption of this application. AC is interesting for the stereo matching algorithm since the end-user of the algorithm can be a human or a neural network. These end-users are both error-resilient since the perception of a human user limitates the required accuracy, and a neural network learns from the approximate output and compensates the induced errors.

The challenge, when introducing AC in the stereo matching algorithm, is to reduce the computational load of the algorithm to be able to embed it in widespread platforms, as for instance Digital Signal Processorss (DSPs). Several techniques have been proposed to reduce the complexity of the stereo matching algorithm.

For instance, Menant et al. [MPMN14] proposed two different AC techniques to tradeoff the accuracy of the algorithm for its energy so as to embed the stereo matching algorithm on a C66x DSP. The first approximation proposed is at the data-level. The data format is modified using fixed-point format. Indeed, formatting data in fixed-point allows benefitting from Single Instruction Multiple Data (SIMD) instructions up to 8 ways. Besides, reducing the operand bit-width allows saving energy as presented in Chapter 2. The second approximation proposed is at the computation level and is mainly driven by the fixed-point conversion. The sophisticated mathematical functions as exponential or square root functions have been replaced by piecewise linear approximations. To evaluate the quality degradation induced by these modifications, the Middlebury metric, which is presented in Section 4.3.1 and which compares the obtained depth map with a ground

truth, is used. The induced quality degradation leads to a difference between the reference implementation and the approximate one of 0.5% of bad pixels obtained in the output depth map compared to the ground truth. The negligible quality degradation leads to a significant speed-up in terms of computation time, which amounts to 60× using SIMD instructions.

Another approximation technique that can be implemented is the downsampling of the input images as presented in Chapter 2. Downsampling the input images reduces the volume of computations but also drastically reduces the output quality. When implementing an AC technique, quality deteriorations may appear. In order to check that the application QoS is still met despite the induced approximations, several metrics can be used, and may impact the obtained measurement of the QoS. This chapter describes a novel approximation technique at the computation level which allows reducing the computational workload. The quality of the proposed approximation technique has been evaluated with two different quality metrics, the reference Middlebury metric and the Structural Similarity Index Measure (SSIM) metric. Despite an interesting quality/complexity trade-off, the obtained results on the quality/computation time trade-off are strongly dependent on the metric used.

4.1.2 Reference algorithm

Our contribution focuses on a binocular stereo matching algorithm. The considered algorithm aims at reconstructing the depth information in an observed scene from a pair of images. For each pixel in the observed scene, the algorithm outputs a disparity level, which corresponds to the projection of the pixel present in the pair of images on a 3D scene. The images have to be taken from distinct and rectified viewpoints. The stereo matching algorithm mimics the human visual system. It outputs a depth map with the disparity levels of each pixel, from the two rectified input images which correspond to the images seen from the left and right eye respectively, in the human visual system, as illustrated in Figure 4.1 extracted from [Dum15].

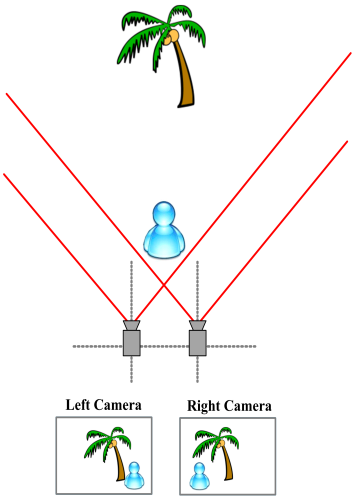


Figure 4.1 – *Principle of the stereo matching algorithm.*

In the stereo matching algorithm, a pixel in the first image is selected, and its matching pixel in the second image is searched. The searched pixel corresponds to the same physical point in the captured scene as the pixel in the first image. Then, the horizontal distance

between those two pixels is computed: this is the disparity of this pixel. The disparity of a pixel corresponds to its depth in the input images, and is found minimizing a cost function. The cost function to minimize computes the cost of matching a pixel in the first image to a pixel in the second image.

Two categories of algorithms have been proposed in the literature, depending on the technique used to minimize the cost function. The minimization technique can be global or local. Local methods optimize the computed cost of matching a pixel in the first image with a pixel in the second image using only the neighboring pixels of the pixel under consideration. Global methods optimize the computed cost over the whole image. Intuitively, the matching cost will be lower for pixels with similar colors and layout of neighboring pixels (similar gradient, colors, edges). Local methods provide a lower quality compared to global methods but are more efficient in terms of computation time. The chosen reference stereo matching algorithm uses a local minimization method to better suit the constraints on embedded platforms.

The main steps of the stereo matching algorithm are detailed below and represented in Figure 4.2:

- **Cost Construction**: computation of the cost of matching a pixel in the left image with a pixel in the right image for a given disparity which represents the distance between both pixels. This step is computed for each pixel and for each disparity level.
- **Cost Aggregation**: refinement of the cost maps obtained from the cost computation step. This step is computed for each pixel and each disparity level.
- **Cost Minimization**: selection of the disparity leading to the minimum cost. This step computes the *argmin* of the cost maps obtained in the cost aggregation step, over the different possible disparity levels. This step is computed for each pixel.

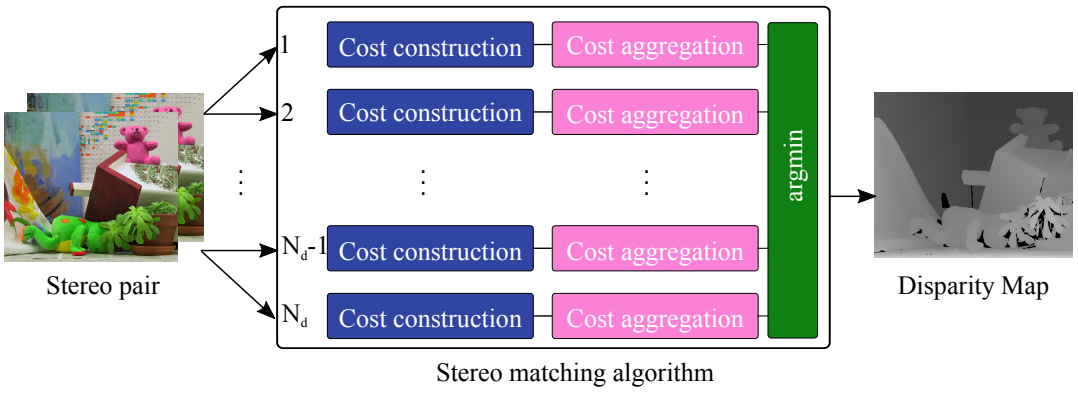


Figure 4.2 – *Illustration of the reference stereo matching algorithm: exhaustive test of all the disparity levels.*

Let N_p and N_d be the total number of pixels in the image and the number of tested disparity levels, respectively. This computationally intensive algorithm studies, for a pair of pixels, all the disparity possibilities in order to create the disparity map. In other words, the cost construction, aggregation and minimization functions are applied $N_p * N_d$ times. For instance, if the image is 375 pixels long, 450 pixels wide and if 60 disparity levels are tested, the three functions are applied $375 \times 450 \times 60$ times, that is to say 10125000

times. An exhaustive search of the minimum of the cost function is processed. This costly exhaustive search appears to be a challenge when it comes to embedding the stereo matching algorithm.

4.2 Proposed algorithm-level approximation

4.2.1 The approximate stereo matching algorithm

The reference stereo matching algorithm requires the computation of the cost function on each disparity level, to then select the disparity level leading to minimum cost function. The proposed approximation method aims at skipping computations of the cost function. Instead of computing the cost function on each disparity levels, a few disparity levels will be selected to reduce the computational load. The proposed method is inspired by the work of Lou et al. [LNLB16]. Lou et al. proposed an adaptation of loop perforation to image processing, then called image perforation. Image perforation is the application of loop perforation to image pipelines. An image pipeline is a succession of processing stages where an intermediate image is produced in each stage, and is fed to the next stage. The authors proposed to skip the computations of costly intermediate images to the benefit of a speed-up in the algorithm computation time. Reconstruction is then applied to reconstruct the missing intermediate images. A similar approach has been implemented by Mercat et al. [MBP+17] for developing a low energy High Efficiency Video Coding (HEVC) encoder where an optimization process is launched and the search space is reduced by applying AC at the computation level. This process in HEVC is called the Rough Mode Decision.

In the case of the stereo matching algorithm, to avoid the exhaustive search of the disparity level minimizing the cost function, an empirical study of the cost function is proposed. The cost function computes the cost for each disparity level d, of matching a pixel of coordinates (x, y) in the first image to a pixel of coordinates $(x + d, y)$ in the second image. The cost function computes the similarity between those two pixels and their neighbourhood, in terms of intensity and local texture. In terms of complexity, the computation of the matching cost for a given pixel and a given disparity requires $2 + 28 * N$ multiplications, where N is a parameter of the algorithm representing the number of iterations to refine the cost computation. N allows adjusting the accuracy of the obtained matching and also controls the size of the considered neighbourhood.

The cost function is finally minimized over all the disparity levels. In the original stereo-matching algorithm, the minimization is done by computing the cost function on each disparity level and selecting the disparity level leading to minimum cost. Nevertheless, analyzing the general shape of the cost functions, a property can be observed. A few examples of the obtained cost functions depending on the disparity level d, $\mathcal{C} = f(d)$, are presented in Figure 4.3.

As it can be observed in Figure 4.3, the studied cost functions are quasi-unimodal. The unimodality property is defined in the following Proposition:

Proposition 4.2.1. *A function f, defined on the interval $I = [a; b]$, is unimodal on I if it has a unique minimum x_0 on I, it is strictly decreasing on $[a; x_0]$ and strictly increasing on $[x_0; b]$.*

The unimodality property is interesting to observe on a function since it implies that it has a single peak. The extremum search can then be eased using methods as the Golden Ratio method or Fibonnacci method for instance as presented by Malenge [Mal77]. These methods are searching for the extremum in interval $I = [a; b]$ by studying the sign of the

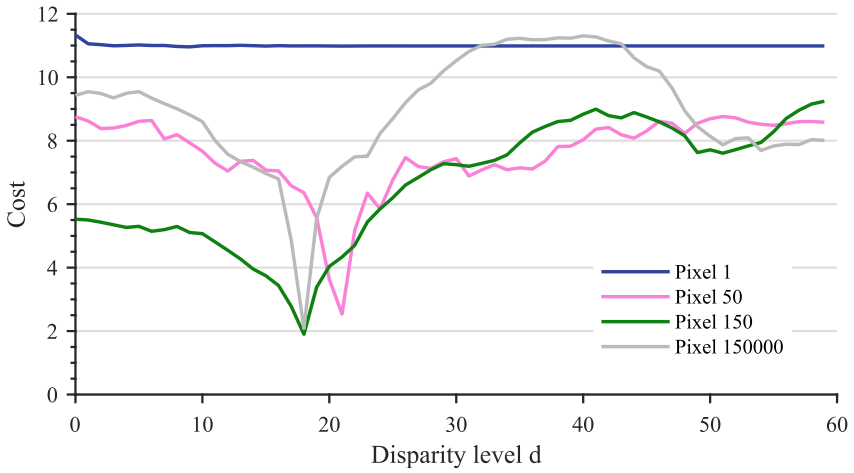


Figure 4.3 – *Cost against the disparity level for different pixels in image Teddy.*

difference of the function taken in two points of I. In the case of the stereo matching algorithm, the obtained cost functions are not strictly unimodals. Indeed, they possess a global minimum but also local minima, as it can be seen in Figure 4.3. The strict monotonicity is not guaranteed and methods to find the extremum of unimodal functions have to be adapted. In our case, for each pixel, the cost function is quasi strictly monotonic on each side of its minimum (pixel 50 and 150 in Figure 4.3). It can be noted that for the first pixel in image Teddy, the cost function does not vary a lot, though a minimum can still be observed by zooming at disparity level $d = 3$. This is due to the fact that this pixel is located in the border of the image. Consequently, as proposed with the Golden Ratio or Fibonnacci methods, a search space decimation technique can be implemented to find the minimum of the function without testing all the disparity levels. A subset of the disparity levels is tested. Iteratively refining the search space around local minima converges towards the global minimum in case of a unimodal function, if the number of iterations is big enough. If the search space is well derived, the output quality is acceptable.

The proposed method is composed of several iterations. The challenge of the proposed search space decimation technique is to encompass the global minimum in the decimated space. As an answer to this challenge, the proposed technique aims at exploring the more disparity levels in the first iteration, so as to ensure the convergence towards the global minimum and not a local one. In the next iterations, the global minimum is searched around the minimum obtained in the first iteration.

The decimated search space is modeled by a tree structure $\mathcal{T}$. At each level of the tree $\mathcal{T}$, the number of tested disparity levels is represented by the number of children $\mathcal{T}[i]$. The depth of the tree N_l represents the number of iterations to converge towards the solution which is the searched global minimum. In Figure 4.3, the exhaustive search of the disparity leading to minimum cost for the considered pixel requires testing 60 disparity levels. The disparity level leading to minimum cost is $d = 33$. With the proposed approximate algorithm, an example of the tree structure used on this cost function is represented in Figure 4.4. This example has been taken from the stereo matching algorithm applied on the Teddy image [SS03].

In the proposed example, the original search space $\mathcal{S}$, in which the disparity level leading to minimum cost is searched, is composed of 60 disparity levels. The proposed approximation method is applied on each pixel of the rectified image. The depth N_l of the

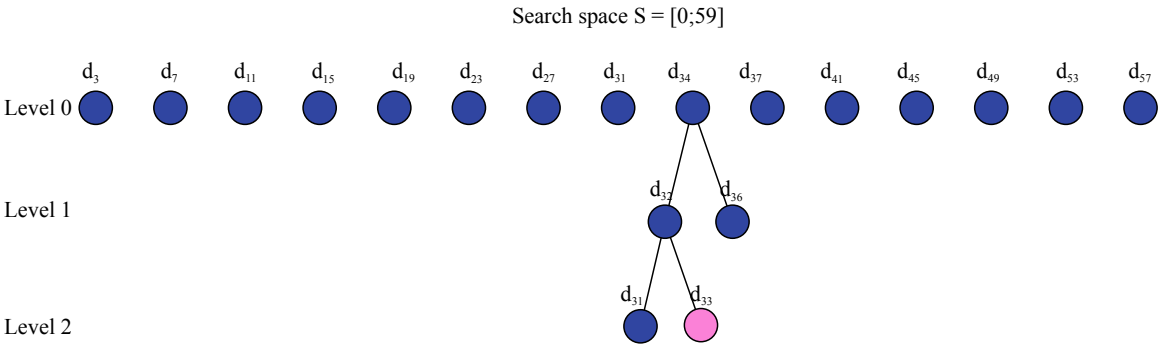


Figure 4.4 – *Tree Structure $\mathcal{T}$ obtained for a given pixel in image Teddy ($N_l = 3$).*

proposed tree structure is 3, which means that the algorithm is composed of 3 successive iterations to refine the minimum found in each level of the tree. To ensure that the tested tree structure encompasses the disparity level leading to the minimum cost, the higher number of disparity levels are tested in the first level. The cost computation and cost minimization steps are applied on a subset of the original search space $\mathcal{S}$. The obtained cost values on the tested disparity levels are compared and the disparity level leading to minimum cost is extracted. Once the disparity leading to the minimum cost has been found for each pixel, the neighboring disparity levels are tested in the following levels of the tree. The proposed method is illustrated on the tree presented in Figure 4.4.

In **Level** 0 which corresponds to the first iteration of the algorithm, the number of tested disparity levels has a strong impact on the obtained quality complexity trade-off. The goal is to test a sufficient number of disparity levels to be close enough to the global minimum of the cost function. In the proposed example, $\mathcal{T}[0] = 15$, thus the cost is computed on a subset of $\mathcal{S}$ composed of 15 disparity levels, for the considered pixel. The disparity leading to the minimum cost is equal to 34. The minimum will be searched in the following levels next to the disparity level 34.

In **Level** 1, the obtained result is refined. The disparity levels surrounding the disparity level selected in Level 0 are studied. In the proposed example, $\mathcal{T}[1] = 2$, thus, the cost is computed on the 2 disparity levels surrounding the disparity level selected in the previous level (disparity levels 32 and 36) and the disparity level leading to the minimum cost is selected, $d = 32$. The minimum will be searched in the following levels next to $d = 32$.

In **Level** 2, $\mathcal{T}[2] = 2$. The cost is computed on the 2 disparity levels surrounding the disparity level selected in the previous level (disparity levels 31 and 33) and the disparity level leading to the minimum cost is selected as being the disparity of the considered pixel, $d = 33$. If the search space has been decimated encompassing the minimum of the cost function, the disparity obtained at the end of the approximate algorithm is the disparity level minimizing the cost function.

The tree $\mathcal{T}$ storing the decimated search space models a trade-off between the complexity of the approximate stereo matching algorithm and the quality of the output depth map. Indeed, the complexity of the algorithm, which impacts the computation time, is reduced with the total number of tested disparity levels, since the cost computation and cost minimization functions composing the stereo matching algorithm are called a smaller number of times. On the exhaustive search, the three functions are called $N_d \times N_p$ times. With the proposed method, the number of tested disparity levels for each pixel is equal to the sum of numbers of disparity levels tested at each level of the tree $\mathcal{T}$. With the tree structure proposed in Figure 4.4, the number of tested disparity levels for each pixel is re-

duced to $\sum_{i=1}^{3} \mathcal{T}[i] = 19$ instead of 60 levels for exhaustive search. The global complexity of the algorithm is reduced along with the output quality.

4.2.2 Selection of the tree structure

The proposed method is based on the selection of the tree structure $\mathcal{T}$ modeling the decimation of the search space. To select $\mathcal{T}$, a minimization problem under constraints is solved. The tree of depth 1 has the highest computational complexity since each disparity level is tested. The tree of depth 1 gives the reference quality, outputting a depth map equivalent to the depth map given by the reference algorithm. The tree of maximum depth, which is the number of factors in the prime factorization of the number of disparity levels to test, tests the least disparity levels. If ϵ is the maximum acceptable degradation on the output quality, compared with the reference output depth map, the problem to solve can be described as follows.

Let $\mathcal{T}$ be the tradeoff tree, $\mathcal{T}[i]$ be the number of disparity levels tested at level i, N_l be the depth of $\mathcal{T}$, N_d be the maximum number of disparity levels and $\mathcal{Q}$ the degradation on the output quality compared to the reference depth map:

$$\begin{aligned} \underset{i}{\text{minimize}} \quad & \sum_{i} \mathcal{T}[i] \\ \text{subject to} \quad & \mathcal{Q} \leq \epsilon \\ & \prod_{i=1}^{N_l} \mathcal{T}[i] = N_d \end{aligned}$$

The first constraint expresses the acceptable quality degradation. The second constraint indicates that no overlapping is tolerated when testing the disparity levels. This means that a disparity level is only tested once. The deeper the tree, the lower the output quality. Indeed, if the global minimum of the cost function has to be reached in a fixed number of iterations, N_l, the more disparity levels are tested in the first iteration, the more chances to reach the global minimum there are. The tree that tests the most disparity levels in the first iteration is the least deep. On the proposed example, the maximum depth of the tree $\mathcal{T}$ is 4.

The second constraint, $\prod_{i=1}^{N_l} \mathcal{T}[i] = N_d$, can be relaxed so as to increase the output quality while testing a lower number of disparity levels. Two other parameters can be adjusted to derive the subset of different disparity levels to test: the spacing between each tested disparity level at each level of the tree, represented in column $\mathcal{S} = \{s_1, ..., s_n\}$ of Table 4.1, where s_i corresponds to the spacing considered at level i. The spacing is the horizontal distance between two tested consecutive disparity levels. The second parameter that can be adjusted is the set of the values used to compute the first disparity level at each level of the tree, from the first disparity of the previous level of the tree, represented in column $\mathcal{F} = \{f_1, ..., f_n\}$ of Table 4.1. For instance, the tree represented in Figure 4.4 corresponds to the tree in Table 4.1 $\mathcal{T} = \{15, 2, 2\}$, with a spacing between consecutive disparity levels of 4 in Level 0 of $\mathcal{T}$, 2 in Level 1 and 1 in Level 2 and $\mathcal{F} = \{3, -2, -1\}$. At Level 0, the first tested disparity level is equal to f_1. The computation of the i^{th} tested disparity level d_{ij} at level j of the tree is indicated in Equation 4.1 depending on Tables $\mathcal{S}$ and $\mathcal{F}$.

	$\mathcal{T}$	$\mathcal{S}$	$\mathcal{F}$	Complexity
Depth 1	$\{60\}$	$\{1\}$	$\{0\}$	60
Depth 2	$\{30, 2\}$	$\{2, 1\}$	$\{1, -1\}$	32
	$\{20, 3\}$	$\{3, 1\}$	$\{1, -1\}$	23
Depth 3	$\{15, 2, 2\}$	$\{4, 2, 1\}$	$\{3, -2, -1\}$	19
	$\{16, 2, 2\}$	$\{4, 2, 1\}$	$\{3, -2, -1\}$	20
	$\{17, 2, 2\}$	$\{4, 2, 1\}$	$\{3, -2, -1\}$	21
	$\{18, 2, 2\}$	$\{4, 2, 1\}$	$\{3, -2, -1\}$	22
Depth 4	$\{5, 3, 2, 2\}$	$\{12, 4, 2, 1\}$	$\{9, -6, -2, -1\}$	12

Table 4.1 – *Different tested tree structures for an original search space $\mathcal{S}$ composed of* 60 *disparity levels.*

$$d_{00} = \mathcal{F}(0) \tag{4.1}$$
$$d_{0j} = d_{0j-1} + \mathcal{F}(j) \tag{4.2}$$
$$d_{ij} = d_{0j} + i \times \mathcal{S}(j) \tag{4.3}$$

Then, to compute the first disparity level d_{01} at Level 1 depending on the first disparity level d_{00} of Level 0, f_2 is used as $d_{01} = d_{00} + f_2$.

The different tree structures tested in the experimental results are presented in Table 4.1. In Table 4.1, the complexity represents the number of tested disparity levels on each pixel. The reference stereo matching algorithm computes the cost construction, aggregation and minimization on $N_d = 60$ disparity levels. Reducing the number of computations of these three functions should reduce the computation time of the stereo matching algorithm along with the output disparity map quality.

4.3 Analysis of the results and conclusion

The proposed algorithmic-level approximation provides a trade-off between the quality of the output depth map and the complexity of the stereo matching algorithm, which should have an impact on the computation time of the algorithm.

To analyze the obtained results, the proposed approximation is tested on the 2003 dataset of the Middlebury database [SS03] composed of two images in full resolution, Teddy and Cones. The obtained results are compared in terms of quality to the Ground Truth given by the Middlebury database and obtained thanks to laser measurements of the depth of the pixels. The reference stereo matching algorithm, with no approximation, and a downsampling method are tested and the quality of the three algorithms is evaluated with two quality metrics. The first quality metric is the reference metric on the stereo matching algorithm, the Middlebury metric, and the second one is more representative of the subjective quality, the structural similarity metric. The obtained results are compared to another type of approximation at the data-level, a downsampling technique. The downsampling technique consists of applying the reference stereo matching algorithm to a pair of images of resolution divided by 4.

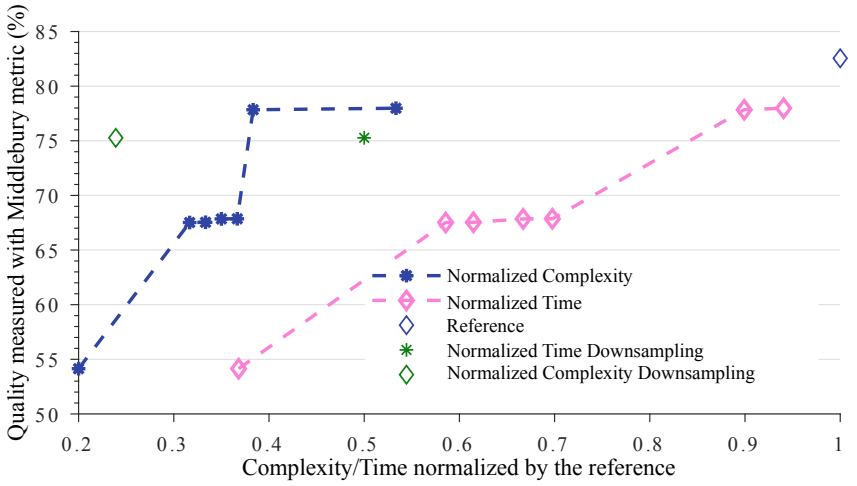
The results have been obtained on an Odroid XU3 board[1] possessing a Cortex A15 processor working at 1.8 GHz. The Odroid XU3 board is heavily used in the embedded

[1] http://www.hardkernel.com

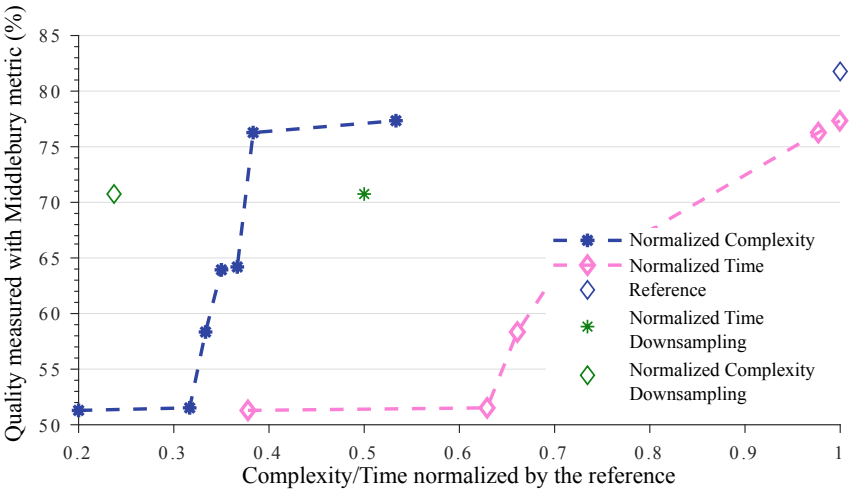
systems domain, since it is highly energy-efficient while offering an important computational capacity. This target allows being in the real conditions of embedding the stereo matching algorithm.

4.3.1 Experimental Study with Middlebury Metric

The first considered quality metric is proposed along with the Middlebury database. The quality measurements done with the Middlebury metric compares the disparity levels obtained in the Ground Truth depth image to the disparity levels obtained at the output of the stereo matching algorithm, for each pixel of the depth map. This metric outputs a percentage of good pixels, corresponding to the percentage of pixels sharing the same disparity level in both images. The complexity for each tested algorithm is computed by computing the number of times $N_p \times N_d$ that the cost construction, aggregation and minimization functions are applied.



(a) Image Teddy.



(b) Image Cones.

Figure 4.5 – *Quality/complexity, computation time trade-off for two images with the Middlebury metric, percentage of good pixels.*

Figures 4.5(a) and 4.5(b) show the trade-off between the quality of the output depth map measured with the Middlebury metric, and the complexity/computation time of the algorithm, for the images Teddy and Cones respectively. The theoretical results on the complexity reduction are compared with the obtained results on the computation time. Both curves are following the same trend, but the normalized computation time is shifted to the right compared to the normalized complexity. This shift means that the observed execution time remains much higher than expected when studying the algorithm complexity. Indeed, the reduction of the number of disparity levels to test requires the computation of the cost functions on a non-regular pattern, which generates an overhead in terms of computation time. This implementation overhead is not taken into account with the representation of the complexity. Besides, computing the cost function on the whole disparity map allows mutualizing the computations, thus saving computation time.

The reference algorithm, with no approximation is represented by the rightmost point ($N_l = 1$). The three functions, cost construction, aggregation and minimization, are evaluated on 60 disparity levels for each pixel. Then, the points are evolving with the different tree configurations presented in Table 4.1, showing an interesting group of points with a large complexity reduction while losing 15% of good pixels for Teddy, and 17% for Cones.

The downsampling technique is represented. The input images have been downsampled by a factor 4, hence the division of the theoretical complexity by 4. This method seems to give better results than the proposed approximate algorithm. For both images, to get a comparable quality of the output depth map, the downsampling technique takes half the time of the proposed approximation technique. According to the Middlebury metric, the proposed approximation technique possesses a non-negligible overhead compared to a more simple technique as downsampling the input images.

4.3.2 Experimental Study with SSIM Metric

The degradation generated by the downsampling technique was evaluated as negligible using the Middlebury metric. When subjectively comparing the output depth maps obtained with the downsampling technique, the quality seems to be strongly degraded though. Similarly, massively used image quality metrics as the Mean-Squared Error (MSE) or the Peak Signal to Noise Ration (PSNR) do not render the subjective quality perceived by the human visual system. AC techniques have been proposed to benefit from the imperfection of the end-sensors of some algorithms, and new error metrics have also been developed to better take into account the subjectively perceived quality. The SSIM is a metric defined in [WBSS04] notably to better take into account the human perception when evaluating the degradation of an image signal. This metric is based on the assessment that the human eye mostly detects the image structural modifications. It has been proposed to evaluate the visibility of induced errors compared to a reference, using properties of the human visual system. The other method to evaluate the visual perceived quality is through subjective tests, which are time-consuming and expensive to lead. The SSIM aims at objectively evaluating the image quality. Indeed, as presented in Figure 4.6 extracted from [WBSS04], two images can have a similar value of MSE while rendering a completely different visual quality. The image in Figure 4.6(a) has a SSIM of 0.9168 compared to the reference image because it is contrast-stretched. The structural information of the image still are preserved, hence the high value of SSIM. The closer to 1 the SSIM is, the higher the quality is. On the contrary, the image in Figure 4.6(b), the image is blurred compared to the reference image, losing important structural information, hence the lower SSIM.

(a) MSE = 210 and SSIM = 0.9168 (b) MSE = 210 and SSIM = 0.7052

Figure 4.6 – *Image quality perceived by the human visual system versus image quality metrics.*

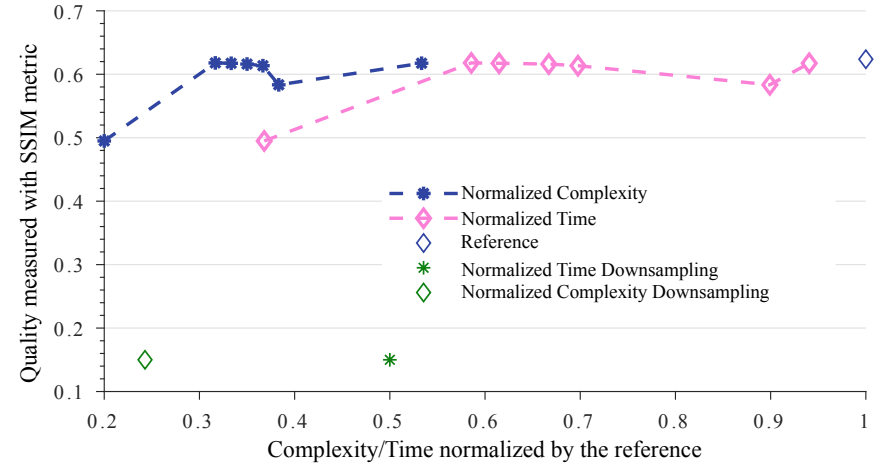
The computation of the SSIM Index is applied when the reference image is known. This image is then considered as being of perfect quality. Then, the SSIM Index computation takes into account different parameters to compare them on the two images. The structures of both images are compared since natural images are highly structured, their pixels being highly spatially correlated. The structures of both images are compared ignoring the classical luminance or contrast information, but using local luminance and constrast information. Three information are compared to measure the similarity of both images: luminance, contrast and structure. To compare the reference image and the approximated one, the SSIM metric segments the images into blocks and compares statistical information between the corresponding blocks in both images. If the input image signals are $\mathbf{x}$ and $\mathbf{y}$, the SSIM measurement is computed as:

$$\mathrm{SSIM}(\mathbf{x},\mathbf{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{4.4}$$

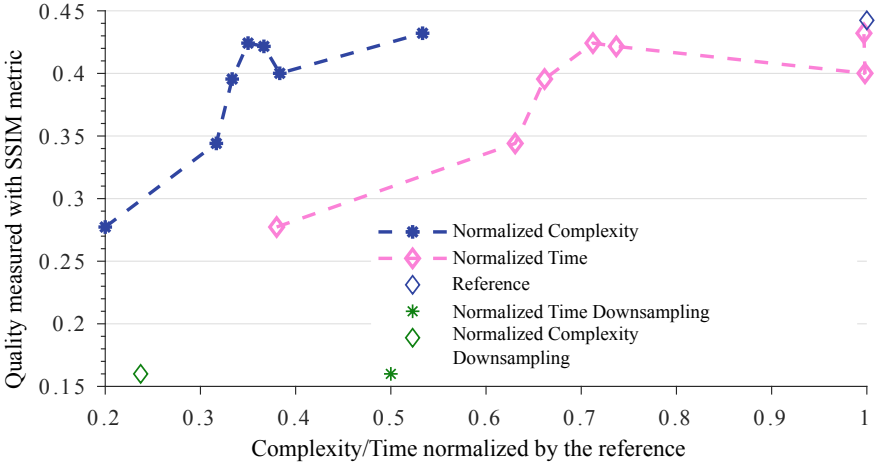
where μ_x and μ_y are the mean intensity of x and y respectively, σ_x and σ_y are estimates of the contrast of x and y respectively, and σ_{xy} measures the correlation between x and y. Finally, C_1 is a constant to compensate for $\mu_x^2 + \mu_y^2$ when they are close to 0, similarly for C_2 and $\sigma_x^2 + \sigma_y^2$.

The results obtained for the proposed algorithm-level approximation technique with the SSIM metric are presented in Figures 4.7(a) and 4.7(b) for the Teddy and Cones images respectively.

For the proposed method, the Ground Truth given by the Middlebury database is taken as being the reference image. The closer to 1 the SSIM Index is, the better the result is. Consequently, compared to a downsampling technique, which offers a quicker result but a strong quality degradation, SSIM Index equal to 0.15 for Teddy and 0.16 for Cones, the proposed method keeps the SSIM Index close to the reference quality and is, in the case of the tree of depth 4, faster than the downsampling technique. The results of the proposed approximation form a plateau that gives the possibility to reduce the computation time by 50% while keeping almost the same output quality. With the SSIM quality metric, the



(a) Image Teddy.



(b) Image Cones.

Figure 4.7 – *Quality/complexity, computation time trade-off for two images with the SSIM metric.*

quality loss on the output image is limited for a relatively important computation time saving.

4.3.3 Conclusion

To conclude, when implementing the proposed algorithm-level approximation technique, it has been noted that the obtained results compared to a more basic downsampling method are strongly dependent on the chosen quality metric. Indeed, the downsampling technique does not preserve the contours and blurs the output disparity map. The reason why the Middlebury quality metric is more in favor of the downsampling technique is that if the selected disparity level in the compared disparity maps differs from less than 3 levels for a given pixel, this pixel does not count as a wrong pixel. One may question the validity of such a metric that does not render the perceived quality, contrary to the SSIM that compares structural differences in both images and better render the quality subjectively perceived by the human visual system.

The proposed algorithm-level AC technique applied to the stereo matching application exploits the error-resilience property of this algorithm. This property allows testing fewer disparity levels per pixel using the quasi-unimodality property of the computed cost function to lighten the volume of computations. This cost is computed to select the disparity level leading to the minimum cost for each pixel. By avoiding exhaustively testing all the disparity levels for each pixel, the approximate algorithm has a reduced complexity and the computation time is reduced compared to the reference stereo matching algorithm. Nevertheless, compared to a basic approximation technique as downsampling the input images, the proposed contribution does not seem to be interesting using the reference metric. A metric more representative of the subjectively perceived quality can be considered. The SSIM has been chosen. The SSIM has been used to measure the approximation impact on the output quality and the results show that the computation time of the stereo matching algorithm can be halved using its approximated version while having a reduced impact on the quality.

Along with new approximation techniques, robust error metrics have to be proposed to quantify the loss of quality induced by the introduced approximations. Generic error metrics are required to quantify the errors induced by the approximation, and methods to link these errors to the quality of service of the application have to be proposed. This challenge is important to allow using approximations in industrial or safety-critical applications.

CHAPTER 5

Characterization of the Error Profile: Application to Inexact Operators

"Sauf erreur, je ne me trompe jamais"

Alexandre Vialatte

The first Approximate Computing (AC) technique under consideration in this thesis has been inexact arithmetic operators. Numerous designs have been proposed to reduce energy consumption, critical path or area but few methods are available to characterize these designs in terms of induced errors. Currently, analytical techniques are not generic. Simulation-based techniques are not possible for large bit-widths operators if done exhaustively and Monte-Carlo simulations are not giving any information on the confidence on the obtained statistics. In this chapter, a novel characterization method for inexact operators according to three different metrics, the Mean Error Distance (mean ED), the Error Rate (ER) and the Maximum Error Distance (maximum ED), is proposed. This characterization is done according to user-defined confidence parameters. The proposed method has first been published in the international conference European Signal Processing Conference EUSIPCO 2018 [BCDM18] for the estimation of the mean ED and the ER, and extended in the journal Microelectronics Reliability [BCDM19]. The remainder of this chapter is organized as follows: Section 5.1 introduces the problematic of error characterization for inexact operators, Section 5.2 presents the proposed methodology for inexact operator characterization. The computation of error metrics used for the characterization of inexact operators is recalled for clarity and the proposed framework combining the estimation of the mean ED, ER and maximum ED is presented. Finally, Section 5.3 presents the experimental setup and the obtained results in terms of number of simulated samples and quality of the obtained estimation.

5.1 Introduction

Whether the approximation technique used in an application is based on the data, on the computations or on the hardware structure of the system, a precise characterization of the errors induced by the approximation is required. Indeed, to be implemented in commercial or real-life applications, the quality of service (QoS) of the application implementing AC has to be ensured. When it comes to inexact circuits, the emulation is

very long as presented in Chapter 3, since done at the logic level. The determination of the QoS at the output of an application based on the emulation of inexact circuits with Bit-Accurate Logic-Level (BALL) simulations is consequently not possible. An efficient method to determine the QoS at the output of an application implementing inexact circuits is to simulate the accurate version of the application and inject errors that model the approximations. A thorough characterization of the errors induced by inexact circuits is consequently required. For instance, the "Fast and Fuzzy" simulator of inexact circuits proposed in [BDM18c] injects errors according to a model based on error characteristics as the mean ED, or the ER, which are statistical error characteristics, and the maximum ED, which is an extreme error characteristic.

Inexact operators generate errors with various amplitude and ER. The generated error amplitude depends on the location of the erroneous bits of the operator output. The variety of error profiles generated by inexact operators proposed in the literature allows targetting a particular inexact operator that may induce errors only on rare combinations of inputs, for example. Before analyzing the effects of the errors induced by the chosen approximations on the application quality metric, the errors induced by the inexact operator have to be modeled, to avoid exhaustive simulation. A thorough characterization of the approximation error allows choosing the most suitable operator with respect to the implementation constraints and to quantify the impact of the approximation on the application quality metric.

Currently, as presented in Chapter 3, the errors induced by inexact operators can be evaluated with two types of approaches: 1) Analytical methods [LHL15, WLGQ17, MHH+17] mathematically express error statistics as the mean ED or the ER, but are dedicated to specific logic structures and can become really complex to implement in terms of computation time and memory for large bit-widths operators. 2) Functional simulation techniques [DVM12, JLL+17, CCSE18] simulate the inexact operator on a representative set of data and computes statistics on the approximation error. To mimic the inexact operator behavior, bit-accurate simulations at the logic-level are required to reproduce the internal structure modifications of the operator. Nevertheless, BALL simulations are two to three orders of magnitude more complex than classical simulations with native data types, as presented in Chapter 3. Thus, exhaustively testing the operator for all the input value combinations is not feasible for large bit-widths because of the required simulation time.

Commonly, the error statistics are computed by simulating a given number of random inputs [DVM12, JLL+17, CCSE18]. The quality of the statistical characterization obtained from a random sampling is highly dependent on the number of samples taken and on the chosen input distribution. Besides, classical simulation-based analysis do not provide any confidence information on the obtained statistical estimation. Using a great number of samples can be ineffective in terms of simulation time. Similarly, using a too small number of samples can give distorted information on the obtained statistics.

When implementing inexact operators in an application, the objective is to derive the impact of the induced approximations on the application quality metric. The evaluation of the impact of the inexact operator on the application quality metric is done in two steps as presented in Figure 5.1. The errors induced by the inexact operator have first to be modeled (Block 1) for different error metrics. Then, these error metrics are used to evaluate the application output QoS (Block 2) despite the induced approximations. In this Chapter, we will focus on estimating the circuit error metrics of inexact operators (Block 1).

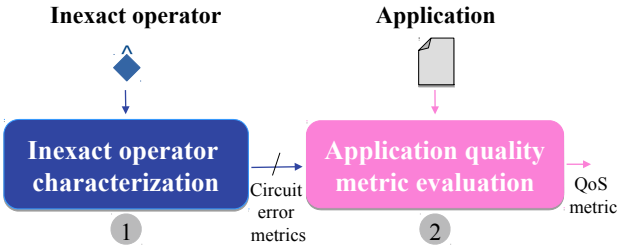


Figure 5.1 – *Proposed framework for evaluating the impact of inexact operators on an application.*

5.2 Inexact Operators Characterization Method

5.2.1 Considered Circuit Error Metrics

Numerous error metrics for inexact arithmetic operators have been proposed (Block 1 in Figure 5.1) and are detailed in Chapter 3. The considered circuit error metrics in this chapter are based on the absolute Error Distance (ED) of the calculation output for an input sample, expressed as:

$$e_i = |\widehat{z} - z| \tag{5.1}$$

where $\widehat{z}$ and z are the erroneous and exact outputs of the computation, respectively. From the absolute ED, statistical error characteristics are extracted, the mean ED μ_{e}, the Standard Deviation $\mathrm{e}_{\mathrm{rms}}$ and the ER f defined on the whole sample set $\mathcal{I}$ as:

$$\mu_e = \frac{1}{N} \sum_{i \in \mathcal{I}} e_i \tag{5.2}$$

$$f = \tfrac{1}{N} \sum_{i \in \mathcal{I}} f_{e_i}, \text{with } f_{e_i} = \begin{cases} 1 \text{ if } e_i \neq 0 \\ 0 \text{ else} \end{cases} \tag{5.3}$$

$$e_{\mathrm{rms}} = \sqrt{\frac{1}{N} \sum_{i \in \mathcal{I}} {e_i}^2} \tag{5.4}$$

where e_i is the Error Distance of the i^{th} stimuli on a sample set $\mathcal{I}$ of size N.

We have also characterized the error induced in terms of maximum ED M_{e} defined as:

$$M_{\mathrm{e}} = \max_{i \in \mathcal{I}}(e_i) \tag{5.5}$$

5.2.2 Error Characterization Framework

The proposed error characterization framework is intended to be used as Block 1 in Figure 5.1 to estimate the following metrics for inexact operators: the mean ED μ_{e}, the Error Rate f and the maximum ED M_{e}. The proposed method shows that the statistical study of the approximation error can lead to a significant reduction in the size of the sample set to simulate in order to characterize an inexact operator given user-confidence information. The proposed framework is not specific to a class of inexact operators and can be applied to adders, multipliers or even more sophisticated operators, which has not been proposed yet. First, statistical parameters as the mean ED and ER are estimated with inferential statistics. Then, the extremum bounds on the approximation error are derived with the Extreme Value Theory (EVT).

Statistical Estimation of the mean ED, the standard deviation and the Error Rate

Inferential statistics, presented by Lowry [Low14], aim at reproducing the behavior of a large population using a subset of this population. This statistical analysis is particularly interesting in the case of high bit-width inexact arithmetic operators, where the exhaustive characterization is not feasible. Using inferential statistics, the input operands set is sampled to give an estimation with an accuracy h and a probability p that the estimated value is contained within the estimated confidence interval, instead of simulating exhaustively all the possible input operands combinations in $\mathcal{I}$. This method is used to compute confidence intervals on the mean ED μ_e and the Error Rate f. Since the probabilistic laws used to estimate those parameters are centered, the obtained confidence intervals also are. In this case, the accuracy h on the estimation of the confidence interval $I = [a, b]$ is expressed as $h = \frac{b-a}{2}$. The objectives of the proposed method are:

1. To estimate the error characteristics more efficiently, using a reduced but sufficient number of samples.
2. To provide the estimated error characteristics according to a given confidence information.

The proposed method computes the minimal number of samples to simulate, to estimate the error characteristics μ_e and the ER according to (h, p). N_{μ_e} and N_f represent the minimal number of samples to estimate μ_e and the ER, respectively.

Computation of the minimal number of samples N_{μ_e} to estimate μ_e

The empirical mean $\overline{\mu_e}$, a punctual estimator of μ_e, is used to estimate the real value of the mean error distance, μ_e. That is to say, $\overline{\mu_e}$ is an estimation of μ_e computed over a given number of samples. $\overline{\mu_e}$ is used to compute the theoretical number of samples N_{μ_e} to simulate to get an estimation according to the confidence parameters (h, p). To estimate N_{μ_e}, the standard deviation of the simulated samples is needed. The empirical mean $\overline{\mu_e}$ and the empirical standard deviation $\tilde{S}^2$, a biased estimator of the standard deviation σ_e, are computed over T samples as:

$$\overline{\mu_e} = \frac{1}{T}\sum_{i=1}^{T} e_i \tag{5.6}$$

$$\tilde{S}^2 = \frac{1}{T}\sum_{i=1}^{T}(e_i - \overline{\mu_e})^2 \tag{5.7}$$

The estimators $\overline{\mu_e}$ and $\tilde{S}^2$ are associated to confidence intervals IC_{μ_e} and IC_{σ_e} respectively, defined such that they include μ_e and σ_e with a probability p. Then, according to the Central Limit Theorem, since (e_1, e_2, ..., e_T) are belonging to the same probability set, are independent and identically distributed, the property in Equation 5.8 is verified if the number of samples N_{μ_e} is higher than 30 [Low14]. Consequently, no assumption has to be made on the distribution of the population, which is generally an underlying assumption of the different analytical methods proposed in the literature. In Equation 5.8, $\mathcal{N}(0, \sigma)$ represents a gaussian distribution whose mean is 0 and standard deviation is σ.

$$\sqrt{N_{\mu_e}}(\overline{\mu_e} - \mu_e) \xrightarrow{\text{law}} \mathcal{N}(0, \sigma) \tag{5.8}$$

The confidence interval $\mathrm{IC}^p_{\mu_e}$ is developed in Equation 5.9 and contains μ_e with a probability p. The term $a^\alpha_{\mu_e}$ embodies the accuracy on the estimation and is computed as in Equation 5.10.

$$\mathrm{IC}^p_{\mu_e} = [\overline{\mu_e} - a^\alpha_{\mu_e}; \overline{\mu_e} + a^\alpha_{\mu_e}] \tag{5.9}$$

In Equation 5.10, z_α is given by the table of the standard normal distribution given p and can be computed as in Equation 5.11, where erf represents the inverse error function [Str68]. N_{μ_e} is the minimal number of samples to simulate to get an estimation respecting the user-defined parameters (h, p).

$$a^\alpha_{\mu_e} = z_\alpha \cdot \frac{\tilde{S}}{\sqrt{N_{\mu_e}-1}} \tag{5.10}$$

$$z_\alpha = \sqrt{2} \cdot \mathrm{erf}^{-1}(p) \tag{5.11}$$

The desired accuracy h on the estimation of the mean ED impacts the number of samples to simulate as expressed in Equation 5.12. To get a desired accuracy of h, $a^\alpha_{\mu_e}$ must be lower or equal to h.

$$N_{\mu_e} > \frac{z^2_\alpha \cdot \tilde{S}^2}{h^2} \tag{5.12}$$

According to Equation 5.12, if the standard deviation of the error generated by the inexact adder is very large, N_{μ_e} can be very high. Inexact operators with a large standard deviation renders circuits with poor interest. In the proposed method, a maximal number of simulated points $N_{\max}$ has been set. If the required number of points is higher than $N_{\max}$, the estimated mean ED and Error Rate ER are given according to p but with a precision h depending on $N_{\max}$. In this case, the obtained accuracy h can be computed depending on $N_{\max}$ as:

$$h = \frac{z_\alpha \cdot \tilde{S}}{\sqrt{N_{\max} - 1}} \tag{5.13}$$

Estimation of the confidence interval on the standard deviation

To estimate a confidence interval on the standard deviation σ_e, the empirical standard deviation $\tilde{S}^2$, a biased estimator of σ_e, is used. The computation of $\tilde{S}^2$ over T samples is presented in Equation 5.7.

Then, if the data $(e_1, e_2, ..., e_N)$ are independent and identically distributed as $\mathcal{N}(\mu, \sigma^2_e)$ [Low14], the sampling distribution associated with the sample variance $\tilde{S}^2$ is a chi-square distribution and Equation 5.14 is verified.

$$\frac{\tilde{S}^2(N-1)}{\sigma^2_e} \xrightarrow{\text{law}} \chi^2_{N-1} \tag{5.14}$$

where χ^2_{N-1} is a chi-square distribution with $N-1$ degrees of freedom. Indeed, the sum of random variables distributed as $\mathcal{N}(0,1)$ is distributed as a chi-square distribution.

The confidence interval $\mathrm{IC}^p_{\sigma^2_e}$ is developed in Equation 5.15 and contains σ^2_e with a probability $p = 1 - \alpha$.

$$\mathrm{IC}^p_{\sigma^2_e} = [\frac{\tilde{S}^2(N-1)}{\chi^2_{1-\alpha/2}}; \frac{\tilde{S}^2(N-1)}{\chi^2_{\alpha/2}}] \tag{5.15}$$

The number of samples N_{σ_e} is searched to find a confidence interval that contains σ^2_e with a probability p such that the width of the interval is equal to $h \cdot \tilde{S}^2$. The obtained confidence interval is asymmetric. The number N_{σ_e} is searched as:

$$\frac{(N-1)\tilde{S}^2}{\chi_{\alpha/2}} - \frac{(N-1)\tilde{S}^2}{\chi_{1-\alpha/2}} < h\tilde{S}^2 \tag{5.16}$$

$$(N-1)\cdot\left(\frac{1}{\chi_{\alpha/2}} - \frac{1}{\chi_{1-\alpha/2}}\right) < h \tag{5.17}$$

Nevertheless, the number of samples to simulate to get a required accuracy of estimation on the standard deviation σ_e is overshooting the obtained values for N_{μ_e} and N_f. Table 5.1 indicates for a few obtained values of $\max(N_{\mu_e}, N_f)$, the obtained width of the confidence interval on the standard deviation if estimated with a similar number of samples, for $p = 95\%$ and $h = 0.05$.

$\mathbf{N}_{\max}$	**h**
578	0.232
5041	0.078
11,765	0.051
35,873	0.029
178,930	0.013
3,130,201	0.003
25M	0.001

Table 5.1 – *Obtained width of the confidence interval on σ_e^2 depending on N_{max}*

The estimation of a confidence interval on the standard deviation meets the required accuracy on the width of estimated confidence interval from 35, 873 simulated samples. The estimation of this parameter has consequently not been proposed, since no guarantee can be given to the user on the accuracy of estimation. In addition, the standard deviation is not a required parameter by the simulator of inexact operators "Fast and Fuzzy" [BDM18c].

Computation of the minimal number of samples N_f to estimate the ER

The proportion of input operands in $\mathcal{I}$ that generate an error is embodied by the Error Rate ER. ER follows a hypergeometric law [Low14]. The estimator used for the error rate is f_e, the proportion of samples generating an error in the random sampling. The estimator is computed as in Equation 3.4, applied on the sampled set. Such an estimator can be associated to a confidence interval IC_f^p that is defined such that the real ER f of the population $\mathcal{E}$ is contained in this confidence interval with a probability p. The confidence interval IC_f^p is defined in Equation 5.18.

$$\mathrm{IC}_f^p = [f_e - a_f^\alpha; f_e + a_f^\alpha] \tag{5.18}$$

In Equation 5.18, a_f^α represents the accuracy on the estimation of ER, z_α is given by the table of the standard normal distribution [Low14] and in Equation 5.20 and N_f represents the minimal number of samples to simulate, to get an estimation with the user-defined parameters (h, p).

$$a_f^\alpha = z_\alpha \cdot \sqrt{\frac{f_e(1-f_e)}{N_f}} \tag{5.19}$$

$$z_\alpha = \sqrt{2}\cdot \mathrm{erf}^{-1}(p) \tag{5.20}$$

To get a desired accuracy of h, a_f^α must be lower or equal to h, which impacts N_f as in Equation 5.21.

$$N_f > \frac{z_\alpha^2 \cdot f_e(1-f_e)}{h^2} \tag{5.21}$$

Estimation of the maximum ED with Extreme Value Theory

The proposed method aims at estimating the maximum ED according to an in-range probability p. An interesting AC technique rarely generates the maximum ED which can consequently be considered as a rare event. Currently, simulation-based techniques are used to estimate the maximum ED but no guarantee is obtained that the real maximum value is not higher than the observed maximum value. The problematic of estimating maximum error values has been massively studied for dynamic range evaluation in the context of fixed-point arithmetic. In this case, if a value transiting through the algorithm is greater than the theoretically derived maximum value, an overflow occurs and may strongly damage the system. For instance, during the flight 501 of Ariane on the 4th of June 1996, the rocket has exploded 36.7 seconds after the take-off because of an overflow in the application system. The past industrial disasters lead to a growing concern towards the study of extremal cases, particularly when tolerating approximations in systems.

The user-defined confidence parameter p allows being more or less conservative on the estimation depending on the critical nature of the application. Our approach exploits the statistical properties of the maximum approximation error, using the EVT. The probability p corresponds to the probability that the real value of the maximum ED M_e is lower or equal to the estimated value of the maximum ED $\tilde{M}$. The higher p, the more conservative the estimation. The studied population for estimating the maximum ED is the set (e_1, e_2, ..., e_T) of error distance values, that are independent and identically distributed events.

EVT [Kin85], [RTR07] aims at describing the stochastic behavior of minima or maxima, and is particularly useful in domains such as finance or insurance. EVT aims at predicting the occurrence or amplitude of rare events even though no observation is available.

The Cumulative Distribution Function (CDF) of the set of error distance values is called G and its associated survivor function is $\overline{G} = 1 - G$. The ordered statistics on a sample of size T can be defined as $e_{1,T} \leq e_{2,T} \leq ... \leq e_{T,T} = M_T$. The proposed method aims at estimating the value $\tilde{M}$ such that:

$$\tilde{M} = \overline{G}^{-1}(\alpha_T)$$

where $\alpha_T = 1 - p < \frac{1}{T}$ when $\lim_{T\to\infty} \alpha_T = 0$. This corresponds to the estimation of the extreme quantile value for α_T. Nevertheless, the CDF and its survivor function are unknown. To estimate $\tilde{M}$, the following property from [Kin85] and [RTR07] is used:

Proposition 5.2.1. *The distributions of extremum values, maxima or minima, converge towards an extreme value distribution.*

Three types of extreme value distributions exist, the Gumbel, Weibull and Fréchet distributions. To identify the type of extreme value distribution followed by the obtained experiments, several information on the shape of the obtained distribution or on conditions on the maximum value can be used. For instance if the maximum is upper-bounded, the followed distribution is Weibull. In the case of the distribution followed by the maximum ED of inexact operators, the shape of the experimentally obtained distribution indicates that the followed distribution is a Gumbel distribution. Nevertheless, the Gumbel distribution is not always followed by extreme values as demonstrated by Chapoutot

et al. [CDV12]. In this case, the assumption is not true in systems as Infinite Impulse Response filters because of the dependency between the different variables.

Contrary to the estimation of mean ED or ER, no confidence interval can be computed on the estimation of M_{e}. The estimated value $\tilde{M}$ corresponds to the value that encompasses M_{e} with a given user-defined probability p. The proposed method is inspired of the dynamic range determination processed for fixed-point arithmetic as presented by Özer et al. [ÖNG08].

To estimate the real value of the maximum ED, the value $\tilde{M} = \overline{G}^{-1}(\alpha_T)$ is computed. So as to compute $\tilde{M}$ for a given probability p, the distribution of the maximum error values has to be studied and identified to a Gumbel distribution.

To derive the maximum error values distribution, T samples are simulated k times. The maximum error value over each sample set of size T is extracted, and the obtained list of maximum error values models the experimental maximum error values distribution. Then, according to the observations of the realizations of the random variable modeling the extreme value, the parameters corresponding to either Gumbel, Weibull or Fréchet distributions are computed (scale and location parameters). In our case, this is a Gumbel distribution. The parameters k and T have an impact on the estimated values of the parameters scale and location, detailed in Equations 5.24 and 5.25, as shown in Figure 5.2(a) for $T = 10$ and Figure 5.2(b) for $T = 1000$. Nevertheless, according to [Kin85], "The grouping of the measurements is usually easy to define because groups are derived from uses of the data". For instance, when studying floods, the measurements are grouped during a year, while chemical measurements are grouped during a day. In this case, the parameters (k, T) do not have an impact on the type of extreme value distribution followed, but on the parameters of the distribution. This is the reason why, in the proposed experiments, the parameters k and T are varied to show the impact of the parameters on the accuracy of estimation.

As shown in Figure 5.2(a) for the ACA_8, when T is small, the approximation by the limit may give poor results.

From the experiments, the obtained distribution of extremum values can be identified to a Gumbel distribution defined hereafter by its density function g in Equation 5.22 and its CDF G in Equation 5.23:

$$g(x) = \frac{1}{\sigma}\exp\left(-\frac{(x-\mu_G)}{\sigma_G}\right)\exp\left(-\exp\left(-\frac{-(x-\mu_G)}{\sigma_G}\right)\right) \tag{5.22}$$

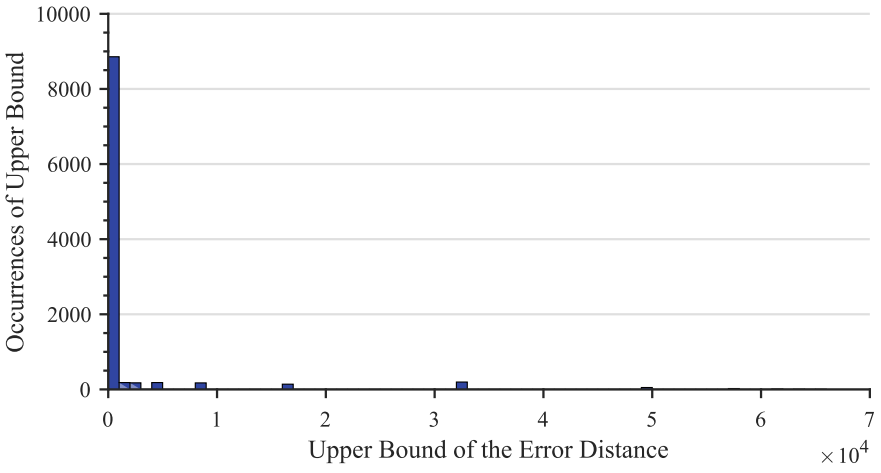
$$G(x) = \exp\left(-\exp\left(-\frac{(x-\mu_G)}{\sigma_G}\right)\right) \tag{5.23}$$

The parameters (σ_G, μ_G) are used to fit the Gumbel distribution to the experimental distribution of maximum error values. The term σ_G is called the scale parameter and is used to stretch or shrink the distribution. The term μ_G is called the location parameter and is used to shift the distribution on the horizontal axis. The computation of (σ_G, μ_G) is detailed in Equations 5.24, 5.25:

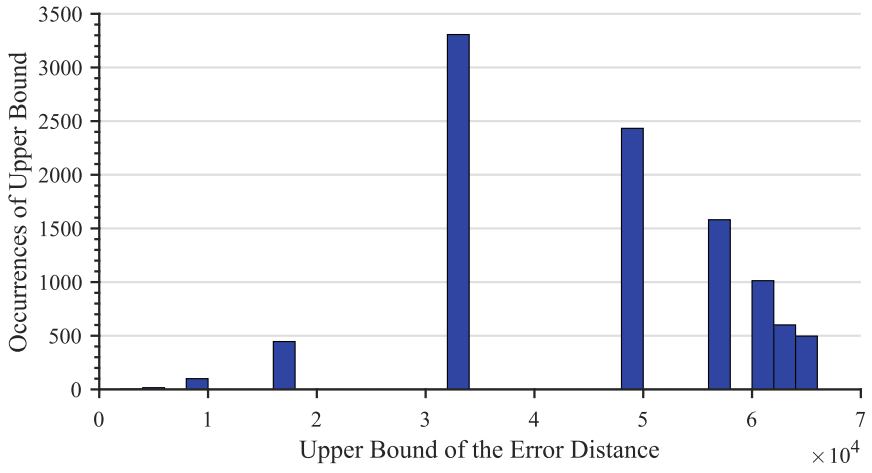
$$\sigma_G = \frac{1}{\pi} \cdot \sqrt{6}\tilde{S}_G \tag{5.24}$$

$$\mu_G = \overline{\mu}_G - \sigma_G \cdot \lambda \tag{5.25}$$

where $\tilde{S}_G$ is the empirical standard deviation of the experimental maximum error values, $\overline{\mu}_G$ is the empirical mean of the experimental maximum error values, and λ is the Euler constant. The parameters (σ_G, μ_G) completely define the Gumbel distribution fitting the maximum error values distribution.



(a) 16-bit ACA_8, $T = 10$, $k = 10000$.



(b) 16-bit ACA_8, $T = 1000$, $k = 10000$

Figure 5.2 – *Distribution of Upper Bound on the Error Distance for two carry chain length values and 16-bit ACA.*

Once the distribution of maximum error values has completely been defined, the goal of our proposed method is to compute the value $\tilde{M}$ such that $G(\tilde{M}) = P(X \leq \tilde{M}) = p$ where p is the in-range probability. Equations 5.26, 5.27 can then be derived.

$$p = P(X \leq \tilde{M}) \quad = \exp(-\exp - \tfrac{(\tilde{M}-\mu_G)}{\sigma_G}) \tag{5.26}$$

$$\tilde{M} \quad = \mu_G - \sigma_G \cdot \ln(\ln(\tfrac{1}{p})) \tag{5.27}$$

Proposed Algorithm

Algorithm 1 presents the estimation of the mean ED and the Error Rate ER with a fair number of samples. From the simulated samples, the maximum ED is also estimated. The population on which inferential statistics are applied is the set $\mathcal{E} = \{e_i / i \in \mathcal{I}\}$. The statistical variables mean ED μ_e, the Error Rate ER and the Standard Deviation (STD)

σ_e are describing the population $\mathcal{E}$ and are consequently characterized by probability laws. To sample the population $\mathcal{E}$, a random sampling method without replacement is used. So that the exhaustive sampling behaves like a non exhaustive sampling, T, the initial number of simulated samples, is taken higher or equal to 30 [Low14].

To characterize an inexact arithmetic operator, the user provides the following information: the desired accuracy on the estimation h, the probability p that the estimated interval contains the real value for μ_e and f, and that the real maximum M_e is lower than the estimated maximum $\tilde{M}$, and the refreshment period T. T is used to refine the number of samples required. The population $\mathcal{E}$ is sampled and T samples are extracted (line 5). The empirical mean $\overline{\mu_e}$, standard deviation $\tilde{S}^2$ and empirical error rate f_e are computed on these samples (lines 9-11). The maximum of the extracted samples is also appended to set J (line 8). From the empirical mean, standard deviation and empirical error rate, the theoretical minimal numbers of samples to compute to estimate μ_e and f according to the user's accuracy constraints is obtained (lines 12-13).

Then, to estimate μ_e and f, the empirical standard deviation $\tilde{S}$, empirical mean $\overline{\mu_e}$ and error rate f_e of the samples are used. Those three estimators are computed to derive the theoretical numbers of samples to simulate to estimate μ_e and f, N_{μ_e} and N_f respectively. The maximum of these two values, N, is taken as the reference number of samples to simulate (line 14). The same process is repeated until the number of simulated samples n exceeds N. The estimated values are consequently refined every T samples to converge towards a minimized value of N, and every T samples, the maximum error value is extracted and appended to the set J. Consequently, the higher T, the more the computations of N_{μ_e} and N_f are accurate. If N is higher than $N_{\max}$, $N_{\max}$ points are simulated but the estimated results are not fulfilling the accuracy requirement, embodied by h. In this case, the obtained accuracy h can be computed depending on $N_{\max}$ as:

$$h = \frac{z_\alpha \cdot \tilde{S}}{\sqrt{N_{\max} - 1}} \tag{5.28}$$

Once the N points have been simulated, the set of maximum error values J is used to identify the obtained distribution of maximum error values to a Gumbel distribution. The parameters σ_G and μ_G are computed (lines 20-21) and used to compute the estimation of the maximum ED according to the in-range probability p (line 22).

5.3 Experimental study

For this experimental study, inexact adders have been selected among three major kinds of topology explored in the literature: timing-starved adders with the Almost Correct Adder (ACA) [VBI08], speculative adders with the Inexact Speculative Adder (ISA) [CSE15, ZGY09] and carry cut-back adders [CSE16]. The important error characteristics when implementing inexact operators are the mean ED, ER and maximum ED.

5.3.1 Estimation of the mean ED and ER

The proposed experimental study aims at showing that:

1. The proposed method correctly estimates error characteristics mean ED μ_e and ER f of circuits for various bitwidths.
2. This estimation remains consistent for higher bitwidths where exhaustive simulation is not possible.

Algorithm 1 Characterization of μ_e, ER and M_e of population $\mathcal{E}$

procedure CHARACTERIZE$\mu_e, f, M_\mathrm{E}(\mathcal{E}, h, p, T, N_\mathrm{max})$
  $\alpha = 1 - p$, $n = 0$
  $J \leftarrow \emptyset$, $E \leftarrow \emptyset$
  repeat
    $E \leftarrow E \cup \mathrm{sampling}(\mathcal{E}, T)$
    $n \leftarrow n + T$
    $M = \max(E)$
    $J = J \cup M$
    $\overline{\mu_e} = \mathrm{computeMean}(E, n)$ ▷ Equation 5.6
    $\tilde{S}^2 = \mathrm{computeSD}(E, n, \overline{\mu_e})$ ▷ Equation 5.7
    $f_e = \mathrm{computeFreq}(E, n)$ ▷ Equation 5.3
    $N_{\mu_e} = \mathrm{computeNMean}(\tilde{S}^2, h)$ ▷ Equation 5.12
    $N_f = \mathrm{computeNFreq}(f_e, h)$ ▷ Equation 5.21
    $N = \max(N_{\mu_e}, N_f)$
    $\mathcal{E} = \mathcal{E} \backslash E$
    if $N \geq N_\mathrm{max}$ **then**
      $N = N_\mathrm{max}$
    end if
  until $n \geq N$
  $\sigma_G = \mathrm{computeScale}(J, n)$ ▷ Equation 5.24
  $\mu_G = \mathrm{computeLocation}(J, n, \sigma_G)$ ▷ Equation 5.25
  $\tilde{M} = \mathrm{computeMax}(\mu_G, \sigma_G, p)$ ▷ Equation 5.27
end procedure

3. For the majority of inexact adders, the proposed method outperforms naive stochastic simulation with a Fixed-Number of Samples (FNS).

In the experiments, two cases are shown: the proposed method requires less samples and thus converges faster towards an accurate error estimation, or it requires more samples than the traditional FNS simulation which is, in this case, not accurate enough.

Each above-mentioned adder architecture have been synthesized, with different bit-widths, from 8 to 32 bits, and varying main design parameters, in order to cover a large spectrum of error behaviors. The proposed characterizations have been completed with $h = 5\,\%$ and $p = 95\,\%$ on an Intel Core i7-6700 processor. The consistency of the error characterization remains the same even with various h and p. The higher p and the lower h, the larger the sample set to simulate.

1) Quality of the estimation for small word-lengths

To first check the quality of the proposed method, small bit-width inexact adders have been characterized with our method, as well as with an exhaustive characterization using BALL simulations to obtain their real error characteristics. The ED values generated by the inexact operators under consideration have been computed for all their possible input values. For instance, for 16-bit inexact operators, the exhaustive characterization requires the simulation of 2^{32} operations.

Table 5.2 reports the confidence intervals on mean ED μ_e and ER f obtained by the proposed method, compared to their real values, and the numbers of samples N used for

the proposed characterization. The ratio N_{ratio} between the number of simulations N done using the proposed characterization method, and the number of simulations done when using an exhaustive characterization is indicated. For 8-bit operators, the number of simulations when using an exhaustive characterization is 2^{16}, while for 16-bit operators, the number of simulations when using an exhaustive characterization is 2^{32}. N_{ratio} is computed as:

$$N_{\text{ratio}} = \begin{cases} \frac{N}{2^{16}} \text{ for 8-bit operators} \\ \frac{N}{2^{32}} \text{ for 16-bit operators} \end{cases} \tag{5.29}$$

For both 8-bit and 16-bit adders, the obtained confidence intervals contain the real values 87.5% of the time, demonstrating that our method is accurate. The 16-bit ACA_8 is the only design for which the obtained confidence intervals do not contain the real values (c.f. bold numbers), but the relative error between confidence interval bound and real value is extremely small. This is coherent, as by user decision, the confidence interval has only 95 % chance to contain the real value.

For most operators, only a few tens of thousands of simulated samples were required to get precise error characteristics. For both 16-bit ACA_{12} and ACA_8, the number of simulated samples has been saturated with $N_{\text{max}} = 25$ millions (c.f. bold sample number). This is due to the fact that ACA adders have a large standard deviation in error values. Though, the proposed method outputs very accurate estimated values of ER and mean ED. The largest relative error on the estimated values compared to the exhaustive characterization is on the estimation of the ER of the operator ACA_8, and is equal to 1.27%.

$\mathbf{N_{bits}}$	Op. type	Name	$\mathbf{IC_{\mu_e}}$		μ_e	$\mathbf{IC_f}$		f	N	$\mathbf{N_{ratio}}$
8	ISA	$ISA_{2,2}$	$8.63{\cdot}10^{-1}$	$9.55{\cdot}10^{-1}$	$8.75{\cdot}10^{-1}$	$1.08{\cdot}10^{-1}$	$1.19{\cdot}10^{-1}$	$1.09{\cdot}10^{-1}$	11,765	0.180
		$ISA_{2,4}$	$4.16{\cdot}10^{-2}$	$1.38{\cdot}10^{-1}$	$9.38{\cdot}10^{-2}$	$1.04{\cdot}10^{-2}$	$3.46{\cdot}10^{-2}$	$2.34{\cdot}10^{-2}$	578	0.009
	ACA	ACA_6	1.67	1.99	1.75	$1.51{\cdot}10^{-2}$	$1.77{\cdot}10^{-2}$	$1.56{\cdot}10^{-2}$	35,873	0.547
16	CCBA	$CCBA_{1,6}$	$7.30{\cdot}10^{-1}$	$8.18{\cdot}10^{-1}$	$7.50{\cdot}10^{-1}$	$1.83{\cdot}10^{-1}$	$2.04{\cdot}10^{-1}$	$1.88{\cdot}10^{-1}$	5041	$1.175{\cdot}10^{-6}$
	ISA	$ISA_{2,4}$	1.95	2.06	1.97	$3.05{\cdot}10^{-2}$	$3.21{\cdot}10^{-2}$	$3.08{\cdot}10^{-2}$	178,930	$4.166{\cdot}10^{-5}$
		$ISA_{2,6}$	$1.73{\cdot}10^{-1}$	$2.69{\cdot}10^{-1}$	$2.42{\cdot}10^{-1}$	$5.40{\cdot}10^{-3}$	$8.40{\cdot}10^{-3}$	$7.60{\cdot}10^{-3}$	11,602	$2.701{\cdot}10^{-6}$
	ACA	ACA_{12}	9.50	9.94	9.69	$4.86{\cdot}10^{-4}$	$4.91{\cdot}10^{-4}$	$4.88{\cdot}10^{-4}$	**25M**	0.006
		ACA_8	$1.71{\cdot}10^{2}$	$1.72{\cdot}10^{2}$	$\mathbf{1.70 \cdot 10^{2}}$	$1.57{\cdot}10^{-2}$	$1.58{\cdot}10^{-2}$	$\mathbf{1.56 \cdot 10^{-2}}$	**25M**	0.006

Table 5.2 – *Estimation results and comparison with exhaustive characterization for operators of small word-lengths (bold numbers if confidence intervals do not contain the real values).*

Op. type	Name	$\mathbf{IC_{\mu_e}}$		$\overline{\mu_e}$ 5M	$\mathbf{IC_f}$		$\overline{f}$ 5M	N	$\mathbf{N_{ratio}}$
CCBA	$CCBA_{1,5}$	$1.564{\cdot}10^{1}$	$1.574{\cdot}10^{1}$	$\mathbf{1.576 \cdot 10^{1}}$	$1.222{\cdot}10^{-1}$	$1.230{\cdot}10^{-1}$	$\mathbf{1.231 \cdot 10^{-1}}$	2,792,512	10^{-13}
	$CCBA_{1,6}$	$1.877{\cdot}10^{1}$	$\mathbf{1.889 \cdot 10^{1}}$	$1.897{\cdot}10^{1}$	$\mathbf{2.867 \cdot 10^{-2}}$	$2.880 \cdot 10^{-2}$	$2.860{\cdot}10^{-2}$	17,008,400	10^{-12}
	$CCBA_{1,7}$	$2.132{\cdot}10^{-1}$	$2.613{\cdot}10^{-1}$	$2.420{\cdot}10^{-1}$	$6.700{\cdot}10^{-3}$	$8.200{\cdot}10^{-3}$	$7.600{\cdot}10^{-3}$	50,176	10^{-15}
	$CCBA_{1,9}$	$4.421{\cdot}10^{-1}$	$5.482{\cdot}10^{-1}$	$5.017{\cdot}10^{-1}$	$1.700{\cdot}10^{-3}$	$2.100{\cdot}10^{-3}$	$2.000{\cdot}10^{-3}$	172,676	10^{-14}
ISA	$ISA_{2,2}$	$8.166{\cdot}10^{3}$	$\mathbf{8.183 \cdot 10^{3}}$	$8.189{\cdot}10^{3}$	$1.246{\cdot}10^{-1}$	$\mathbf{1.249 \cdot 10^{-1}}$	$1.250{\cdot}10^{-1}$	**25M**	10^{-12}
	$ISA_{2,8}$	3.826	3.933	**3.763**	$7.505{\cdot}10^{-3}$	$7.698{\cdot}10^{-3}$	$7.600{\cdot}10^{-3}$	3,130,201	10^{-13}
	$ISA_{2,10}$	$9.125{\cdot}10^{-1}$	1.012	1.003	$4.566{\cdot}10^{-4}$	$5.104{\cdot}10^{-4}$	$4.954{\cdot}10^{-4}$	3,084,740	10^{-13}
ACA	ACA_{17}	$\mathbf{1.433 \cdot 10^{4}}$	$1.812{\cdot}10^{4}$	$1.391{\cdot}10^{4}$	$4.999{\cdot}10^{-5}$	$5.004{\cdot}10^{-5}$	$5.002{\cdot}10^{-5}$	**25M**	10^{-12}

Table 5.3 – *Estimation results and comparison with 5-million BALL simulations for 32-bit operators (bold numbers if confidence intervals do not contain the FNS values).*

2) Consistency of the estimation for 32-bit operators

To check the consistency of the proposed method for 32-bit operators, the proposed characterization has been compared to random FNS simulation with 5 million samples from [CCSE18], which is the typical inexact circuit characterization method as exhaustive simulation is not feasible. The chosen Carry-Cut Back Adder (CCBA) and ACA adders are Pareto-optimal designs shown in the comparative study of [CCSE18]. Those adders are realistic designs to be implemented, and thus represent ideal subjects for the proposed characterization.

Table 5.3 reports the results for 32-bit inexact adder characterization. In the case of 32-bit operators, it is to be noted that both characterizations, the proposed characterization and the one obtained with FNS simulation, are statistical estimates. In case the two methods do not converge towards the same estimation, bold numbers represent values obtained with higher amount of samples, assumed more accurate. The ratio N_{ratio} between the number of simulations N done using the proposed characterization method, and the total number of simulations for an exhaustive characterization (2^{64}) is indicated.

For 2 out of 8 designs ($\text{CCBA}_{1,5}$ and $\text{ISA}_{2,8}$), the obtained confidence intervals obtained with less simulation samples than the FNS simulation do not contain the error values from this latter. Nevertheless, the obtained estimated values of ER and mean ED are very close from the random characterization. Inversely, for 3 of them ($\text{CCBA}_{1,6}$, $\text{ISA}_{2,2}$ and ACA_{17}), the proposed method has converged into different confidence intervals than the BALL simulation, as it has determined that more samples were required for safe estimation. This is coherent, as by user decision, the confidence interval has only 95 % chance to contain the real value. The most critical case concerns ACA_{17}. For this characterization, naive BALL simulation has dangerously underestimated the mean ED compared to the proposed method. This is due to the very low error rate of the 32-bit ACA, for which 5 million samples are insufficient to make good statistics on errors.

3) Number of simulations required for accurate estimation

Algorithm 1 refines the estimation of mean ED and ER given a refreshment period T. Figures 5.3 and 5.4 illustrates the convergence of the estimation on the ER f and the mean ED μ_e respectively. The different curves, corresponding to the different operators, have different starting points depending on the chosen refreshment period T. The relative error of estimation of mean ED and ER depending on the simulation length are represented. To compute the relative error of estimation ϵ of the confidence interval on the mean ED, $\text{IC}_{\mu_e} = [a; b]$, the computation of the center of the estimated interval μ_e is required and is computed as:

$$\mu_e = a + \frac{b - a}{2} \tag{5.30}$$

Finally, the center of the estimated interval μ_e is compared to the FNS value obtained with 5-million BALL simulation $\overline{\mu_{e,5M}}$ as:

$$\epsilon = \frac{|\mu_e - \overline{\mu_{e,5M}}|}{\overline{\mu_{e,5M}}} \tag{5.31}$$

The same process is applied to compute the relative error of estimation of ER.

The final estimated values are all very accurate since the relative error of estimation is always lower than 0.1 %. Small bumps can be noted in the convergence of the estimated values due to the random sampling processed in each iteration of the algorithm. Besides,

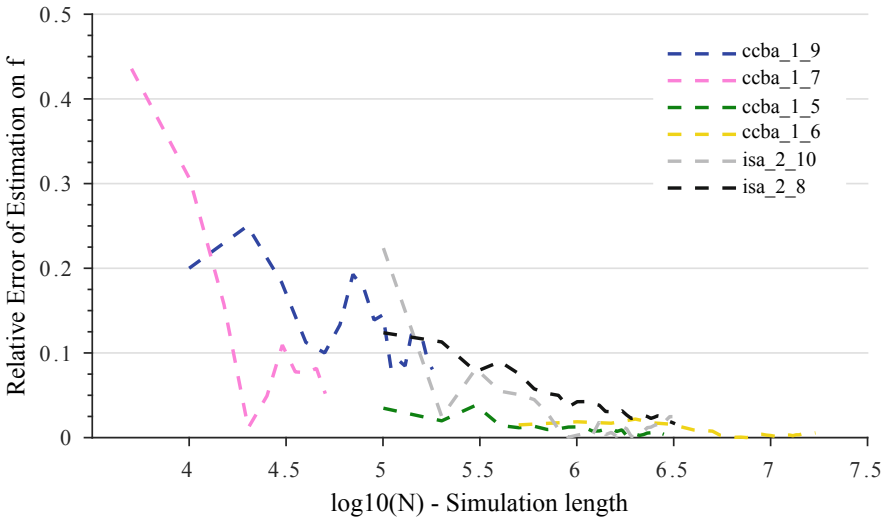


Figure 5.3 – *Convergence of the proposed estimation for the ER f_e with the number of simulated samples N, $p = 95\%$ and $h = 0.05$ for different 32-bit adders.*

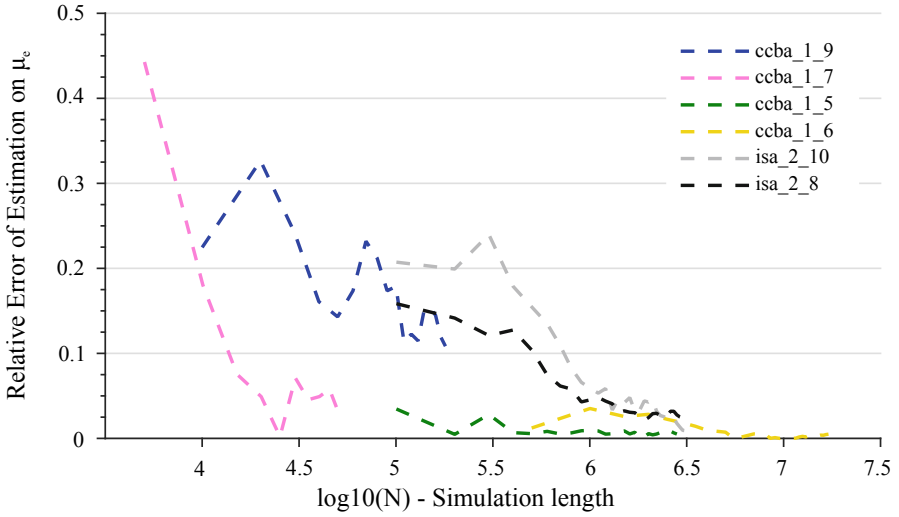


Figure 5.4 – *Convergence of the proposed estimation for the mean ED $\overline{\mu_e}$ with the number of simulated samples N, $p = 95\%$ and $h = 0.05$ for different 32-bit adders.*

the speed of convergence strongly varies depending of the chosen operator. This is why the proposed method, which is an adaptive sample-size method, better fits any operator rather than naive FNS simulations.

5.3.2 Estimation of the maximum ED

The proposed experimental study aims at showing that:

1. The proposed method correctly estimates the maximum ED of circuits for various bitwidths.

2. The quality of the estimated maximum error value is configurable depending on the in-range probability p, the size of the sample sets T and the number of times the T samples are simulated, k.

The total number of simulated points is then $k \times T$. Two cases are shown: the dependency of the quality of the estimation on the total number of simulated samples $k \times T$ and on the in-range probability p. The estimations of the maximum ED have been completed with varying p, T and k on an Intel Core i7-6700 processor.

1) Quality of the estimation for small bit-widths

To first check the quality of the proposed estimation method, the maximum ED of small bit-width inexact adders has been compared to an exhaustive characterization using BALL simulations which shows the real maximum error distance characteristics. Table 5.4 reports the estimated values $\tilde{M}$ of the maximum ED obtained by the proposed method, compared to their real values M_e, depending on the parameters (k, T, p).

$\mathbf{N_{bits}}$	**Op. type**	**Name**	p	k	T	$\tilde{M}$	M_e
8	ISA	$ISA_{2,2}$	90	10	100	8	8
			95	10	100	8	8
			98	10	100	8	8
		$ISA_{2,4}$	90	10	100	4	4
			95	10	100	6	4
			98	10	100	7	4
	ACA	ACA_6	90	10	100	**151**	192
			90	10	100	199	192
			95	10	100	210	192
			98	10	100	249	192
16	CCBA	$CCBA_{1,6}$	90	10	1000	4	4
			95	10	1000	4	4
			98	10	1000	4	4
	ISA	$ISA_{2,4}$	90	10	1000	64	64
			95	10	1000	64	64
			98	10	1000	64	64
		$ISA_{2,6}$	90	10	100	32	32
			95	10	1000	32	32
			98	10	1000	32	32
	ACA	ACA_{12}	90	10	1000	**28467**	61440
			95	10	1000	**41646**	61440
			98	10	1000	66068	61440
		ACA_8	90	10	1000	**63305**	65280
			95	10	1000	67008	65280
			98	10	1000	84188	65280

Table 5.4 – *Estimation results of the maximum ED and comparison with exhaustive characterization for operators of small word-lengths (bold numbers if $\tilde{M} < M_e$).*

For most 8-bit adders, only $k \times T = 1000$ simulations are required to correctly estimate the maximum ED encompassing the real value M_e. This corresponds to the simulation of 1.53% of the exhaustive input data set composed of 2^{16} values. The in-range probability p can be used to be more or less conservative on the estimation. For the example of the ACA_6, the in-range probability can also be used to adjust the accuracy of the estimation. If p is lower than 95%, the obtained estimation $\tilde{M}$ does not always encompass the real maximum M_e. For an in-range probability $p = 95\%$, the estimated maximum value always encompasses the real maximum M_e, demonstrating that the proposed estimation is conservative.

For most 16-bit adders, the estimation of the maximum ED is accurate with only $k \times T = 10^4$ simulations. This corresponds to the simulation of $2.33 \cdot 10^{-4}\%$ of the exhaustive input data set composed of 2^{32} values. The ACA_8 still requires an in-range probability of 95% to encompass the real maximum value M_e. Nevertheless, the ACA_{12} is the only design for which the estimation is accurate only for $p > 98\%$. This operator has very scattered error values and the chance to catch the real maximum during the determination of the error distance values distribution is lower than for the other inexact operators. This characteristic makes the ACA_{12} hardly usable in an application.

2) Consistency of the estimation for 32-bit operators

Table 5.5 reports the results for 32-bit inexact adder maximum ED estimation. To check the consistency of the proposed estimation method for this larger bit-width, the obtained estimations have been compared to random BALL simulation with 5 million samples from [CCSE18].

In the case of 32-bit operators, it is to be noted that both obtained values $\tilde{M}$ and M_e (5**M**) are estimates. No analytical derivation of the maximum ED has been given by the inexact operator designer. For most 32-bit adders excepted the $CCBA_{1,6}$ and $ISA_{2,10}$, the proposed method gives conservative estimates even with an in-range probability of 90%. For the operator $ISA_{2,10}$, the in-range probability has to be greater or equal to 95% to obtain a correct estimation. However, the $CCBA_{1,6}$ requires to set the in-range probability up to 99.9% to encompass the maximum error distance estimated with 5 million samples.

3) Accuracy of the estimation depending on the in-range probability

The in-range probability allows being more or less conservative on the estimate of M_e. Figures 5.5 and 5.6 depict the link between the parameter p and the accuracy of estimation. Indeed, when implementing an AC technique, the maximum ED must not be underestimated. However, if the obtained value overshoots the real maximum ED, the application designer may wrongly discard a technique, hence the importance of adjusting the parameter p.

The estimated maximum error distance values $\tilde{M}$ are represented as a percentage of the accurate maximum ED values M_e for each inexact adder in Figures 5.5 and 5.6. Vertical lines indicate for each inexact adder, when p is high enough to accurately estimate $\tilde{M}$ ($\tilde{M} = M_e$). The proposed method correctly estimates the maximum ED for both inexact adders 8-bit $ISA_{2,2}$ and 16-bit $CCBA_{1,6}$ and for p varying from 25% to 100% since these adders frequently generate the maximum error distance.

For the 8-bit $ISA_{2,4}$, $\tilde{M}$ encompasses M_e when $p \geq 72\%$ and for the 16-bit $ISA_{2,6}$, when $p \geq 68$. Nevertheless, for the $ISA_{2,6}$, ACA_6 and ACA_8, the in-range probability p has to be very high to encompass the real maximum error distance (higher than 85%, 88% and 97% respetively). Indeed, the generated errors are scattered and local maxima may

N_{bits}	Op. type	Name	p	k	T	$\tilde{M}$	M_e (5M)
32	CCBA	$CCBA_{1,5}$	90	500	1000	128	128
			95	500	1000	128	128
			98	500	1000	128	128
		$CCBA_{1,6}$	90	500	1000	**1349**	1641.6
			95	500	1000	**1409**	1641.6
			98	500	1000	**1491**	1641.6
			99.9	500	1000	1743	1641.6
		$CCBA_{1,7}$	90	500	1000	32	32
			95	500	1000	32	32
			98	500	1000	32	32
		$CCBA_{1,9}$	90	500	1000	325	256
			95	500	1000	373	256
			98	500	1000	439	256
	ISA	$ISA_{2,2}$	90	500	1000	65536	65536
			95	500	1000	65536	65536
			98	500	1000	65536	65536
		$ISA_{2,8}$	90	500	1000	16384	16384
			95	500	1000	16384	16384
			98	500	1000	16384	16384
		$ISA_{2,10}$	90	500	1000	**1945**	2048
			95	500	1000	2614	2048
			98	500	1000	3289	2048
	ACA	ACA_{17}	90	500	1000	$4{\cdot}10^9$	$2{\cdot}10^9$
			95	500	1000	$4{\cdot}10^9$	$2{\cdot}10^9$
			98	500	1000	$4{\cdot}10^9$	$2{\cdot}10^9$

Table 5.5 – *Estimation results of the maximum ED and comparison with Monte-Carlo characterization (5M) for 32-bit operators (bold numbers if $\tilde{M} < M_e$).*

be found in the different samples, leading to a lower value of $\tilde{M}$. When p increases, the estimated maximum ED becomes very conservative. Small bumps can be observed for the ACA_6 and ACA_8, also caused by the large standard deviation generated by this type of inexact adder. It is still to be noted that for 8-bit and 16-bit estimations, the number of simulated samples is small, since equal to 1000 samples which represents 1.5% of the whole input space for 8-bit operators, and only $2.3 \cdot 10^{-5}$% of the whole input space for 16-bit operators.

4) Accuracy of the estimation depending on the number of simulated samples

The accuracy of the estimation can be controlled with the total number of simulated points $k \times T$ taken to derive the distribution of the maximum error distance values. Figures 5.7 and 5.8 represent the evolution of the accuracy of estimation depending on the size of the simulated samples T, with k set to 10, for different 8-bit and 16-bit adders, respectively.

Figures 5.7 and 5.8 represent the estimated value $\tilde{M}$ as a percentage of M_e depending on T. In this case, T samples are simulated and their maximum is extracted. This

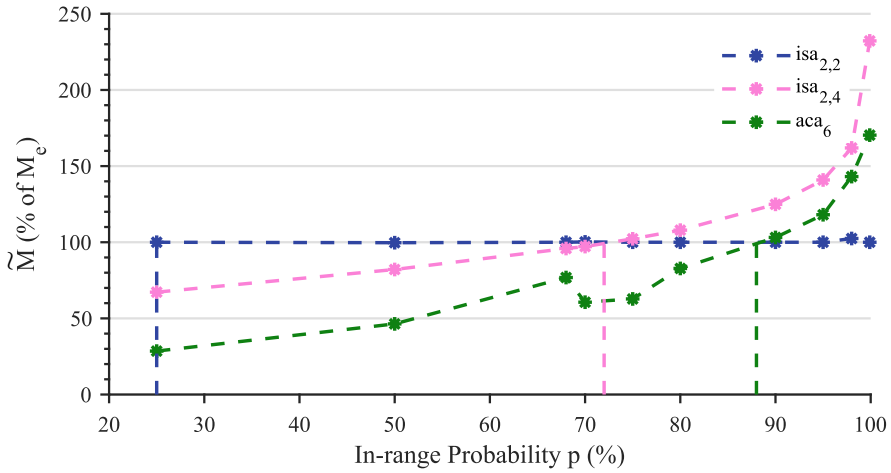


Figure 5.5 – *Estimation of maximum ED as a percentage of M_e depending on the in-range probability p for a fixed number of simulated samples $k \times T$, $k = 10, T = 100$, for 8-bit adders. Vertical lines indicate $\tilde{M} = M_e$.*

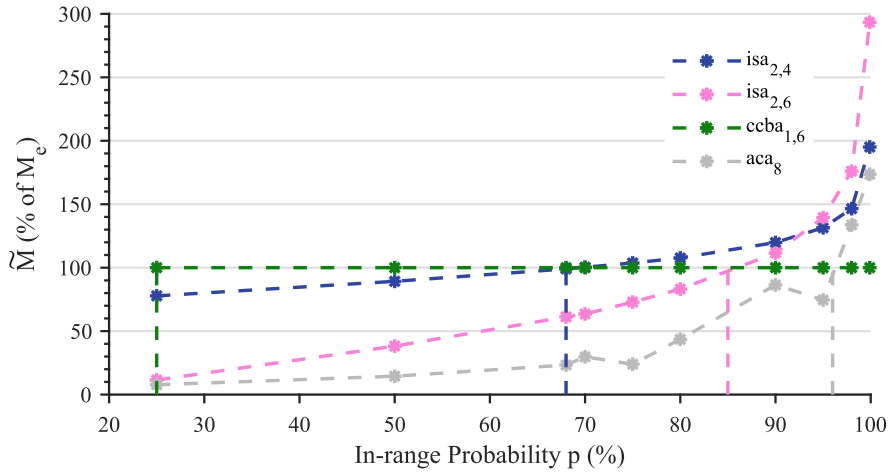


Figure 5.6 – *Estimation of maximum ED as a percentage of M_e depending on the in-range probability p for a fixed number of simulated samples $k \times T$, $k = 10, T = 100$, for 16-bit adders. Vertical lines indicate $\tilde{M} = M_e$.*

operation is done $k = 10$ times. For 8-bit adders, the estimates converge towards a value for 8-bit $\text{ISA}_{2,2}$ and $\text{ISA}_{2,4}$ as soon as $T \geq 25$. For 16-bit adders, the estimates converge towards a value for $\text{ISA}_{2,4}$, $\text{ISA}_{2,6}$ as soon as $T \geq 250$ and for $\text{CCBA}_{1,6}$ as soon as $T \geq 50$. As soon as the size of the samples exceeds 25 for 8-bit adders, and 250 for 16-bit adders, simulating additional samples does not impact the estimated maximum value $\tilde{M}$. The adders $\text{ISA}_{2,2}$, 16-bit $\text{ISA}_{2,4}$, $\text{ISA}_{2,6}$ and $\text{CCBA}_{1,6}$ are converging towards the real value M_e when $T \geq 25$ for the 8-bit adder and $T \geq 250$ for the 16-bit adders. For the 8-bit adder $\text{ISA}_{2,4}$, the estimation is conservative since $M_e = 4$ and the estimate converges towards $\tilde{M} = 5$. This case is not problematic since the relative error of estimation is equal to 25%.

Nevertheless, to estimate correctly $\tilde{M}$ for the ACA_6 and ACA_8, the size of the simulated samples has to be really high compared to the other considered inexact operators, 72 and 870 respectively. Indeed, as shown in Figure 5.5 and 5.6, for a fixed sample size $T = 100$,

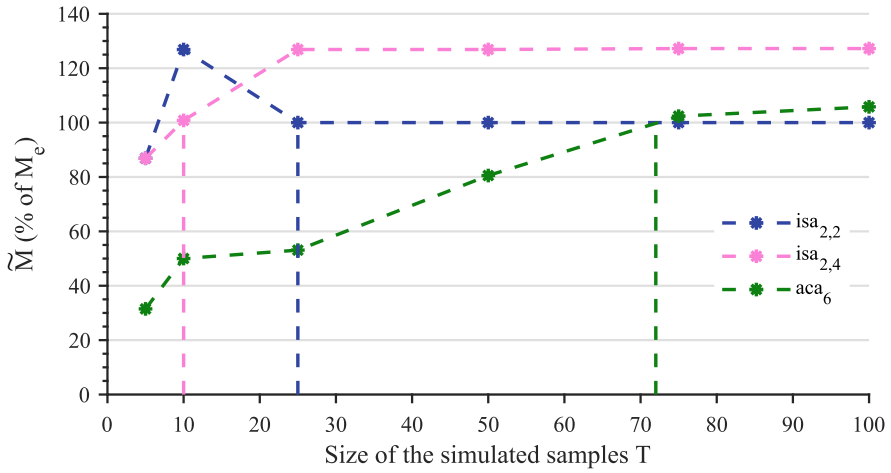


Figure 5.7 – *Estimation of maximum ED as a percentage of M_e depending on the T with $k = 10$, $p = 90\%$, for 8-bit adders. Vertical lines indicate $\tilde{M} = M_e$.*

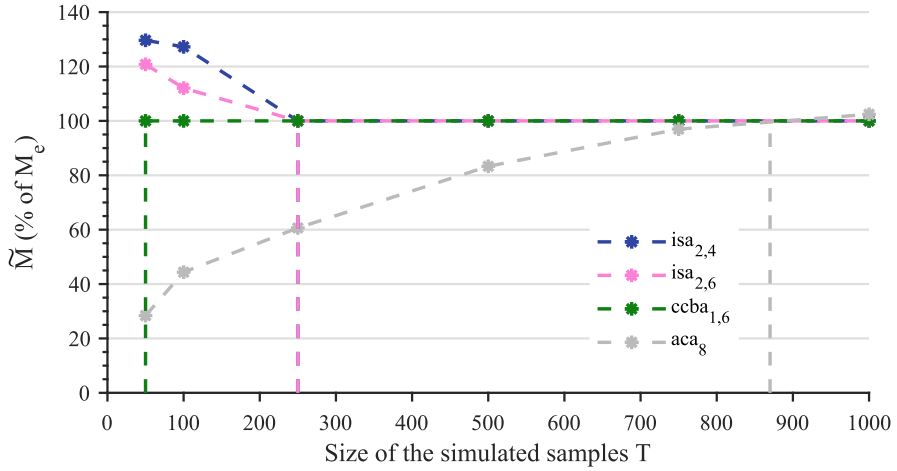


Figure 5.8 – *Estimation of maximum ED as a percentage of M_e depending on the T with $k = 10$, $p = 90\%$, for 16-bit adders. Vertical lines indicate $\tilde{M} = M_e$.*

the estimation of $\tilde{M}$ for both inexact operators ACA_6 and ACA_8 needs a high in-range probability to reach the accurate value M_e.

Figures 5.9 and 5.10 represent the estimated value $\tilde{M}$ as a percentage of M_e depending on k. In this case, T samples ($T = 100$ for 8-bit adders, $T = 250$ for 16-bit adders) and their maximum is extracted. This operation is done a varying number of times k. For 8-bit adders, the $\text{ISA}_{2,2}$ and $\text{ISA}_{2,4}$ converge towards a value as soon as $k = 5$. As shown in Figure 5.7, the ACA_6 would require more simulations to converge. Nevertheless, contrary to the impact of T on the quality of the estimation, in this case, a single adder ($\text{ISA}_{2,2}$) has converged towards the exact value M_e. This is due to the frequent generation of the maximum error value with this inexact adder. For the ACA_6, the estimated maximum $\tilde{M}$ is underestimated. Indeed, if the maximum extracted in the samples of size T is a local maximum very far from the real maximum value, it induces parasite results when computing the Gumbel distribution. For the $\text{ISA}_{2,4}$, the estimated maximum $\tilde{M}$ is overestimated, with a relative error of estimation of 25%. For 16-bit adders presented in Figure 5.10, the estimates converge towards a value as soon as $k = 25$. The curve representing the $\text{ISA}_{2,4}$ is

overlapping the curve representing the $CCBA_{1,6}$. In this case, for both adders $ISA_{2,4}$ and $CCBA_{1,6}$, only $k = 5$ simulations are required to correctly estimate the value $\tilde{M}$. For the $ISA_{2,6}$, the value $\tilde{M}$ is slightly overestimated. Finally, for the same reasons as for the 8-bit ACA_6, the ACA_8 underestimate the value $\tilde{M}$. However, as stated for the estimation of the ER and mean ED, inexact operators with a large standard deviation renders circuits with poor interest.

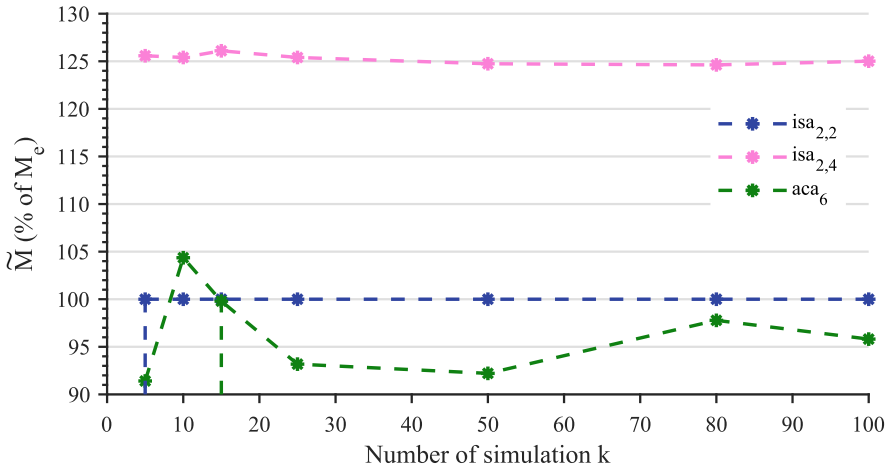


Figure 5.9 – *Estimation of maximum ED as a percentage of M_e depending on k, $p = 90\%$, 8-bit adders, $T = 100$. Vertical lines indicate $\tilde{M} = M_e$.*

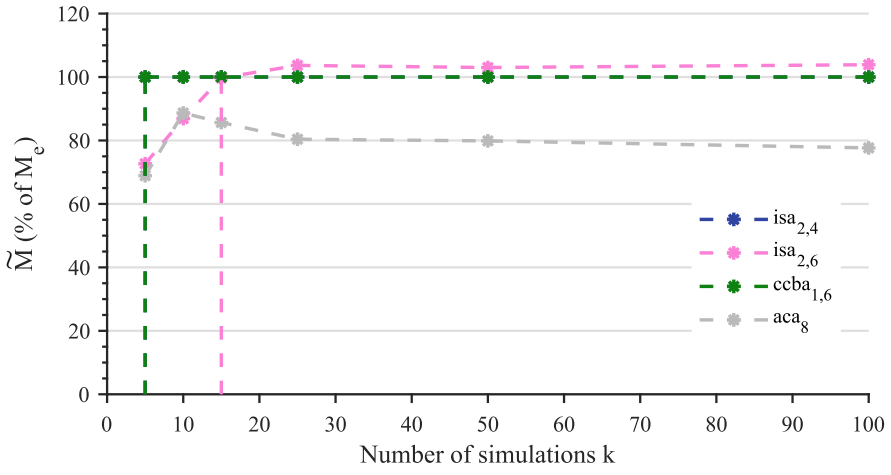


Figure 5.10 – *Estimation of maximum ED as a percentage of M_e depending on k, $p = 90\%$, 16-bit adders, $T = 250$. Vertical lines indicate $\tilde{M} = M_e$.*

To draw a conclusion, to correctly estimate the maximum error distance for an inexact adder, the user will mainly modify the in-range probability p, allowing to be more or less conservative on the estimation without simulating additional samples, or the size of the samples T to ensure to converge towards the real maximum value and not towards an underestimated maximum value to derive the Gumbel distribution.

5.4 Conclusion

In this contribution presented in the international conference EUSIPCO 2018 [BCDM18] and in the journal Microelectronics Reliability [BCDM19], we have proposed a characterization method of the approximation error induced by inexact arithmetic circuits, that exploits the statistical properties of the error. The benefits of the proposed method have been demonstrated on different inexact arithmetic adders (ACA, ISA and CCBA) and the mean error distance, error rate and maximum error distance have been estimated. From user-defined confidence requirements, the proposed method automatically adjusts the number of simulations required by using statistical properties of the approximation error. Validated by its accurate estimation of error characteristics on 8 to 16-bit circuits, the proposed method has been shown coherence and consistency on larger bit-widths, with 32-bit circuits, where exhaustive simulation is not feasible. This experimental study has shown that the proposed method outperforms naive stochastic BALL simulations with a fixed number of samples, either by converging towards a more accurate characterization, or by drastically reducing the amount of samples required for an accurate estimation, saving time and resources. As a future work, the proposed characterization framework could be tested on other types of inexact arithmetic operators as multipliers.

CHAPTER 6

Application Quality Metric: Application to Inexact Operators

"If we didn't have genetic mutations, we wouldn't have us. You need error to open the door to the adjacent possible."

Steven Johnson

Inexact circuits can be characterized in terms of approximation errors. Analytical techniques, exhaustive or Fixed-Number of Samples (FNS) simulations have been proposed in the literature. In Chapter 5, an efficient simulation-based method for a fast characterization of the error induced by inexact operators has been proposed. To link the approximation error induced by inexact circuits to the application quality of service (QoS), the operators have commonly to be emulated at gate-level within the application. As previously shown, this leads to long simulation time. To reproduce the modifications of the internal structure of the operator, Bit-Accurate Logic-Level (BALL) simulations are used. In this chapter, the error injection technique is exploited. A simulator has been proposed to replace the slow BALL simulation of an inexact operator $\widehat{\diamond}$ so as to fasten its simulation and be able to link the circuit error metrics to the application quality metric. The BALL simulation is replaced by the simulation of the exact operator $\diamond$ plus a Pseudo-Random Variable (PRV) modeling the approximation error. The modelization of the error induced by the inexact operator by a stochastic process is described in Section 6.2. In this Section, two versions of the proposed simulator, called the Fast and Fuzzy (FnF) simulator, are described. The first version of the FnF simulator (FnF_i), has been presented in the international conference GLSVLSI 2018 [BDPM18]. This version of the simulator takes into account the correlation between the input values of the operator and the errors of approximation. The second version of the FnF simulator (FnF_o), has been presented in the international conference ISCAS 2018 [BDM18a]. This version of the simulator takes into account the correlation between the output values of the operator and the errors of approximation. The construction of the simulator is described in Section 6.3. A comparison of both simulators is provided in Section 6.4.

6.1 Introduction

In Chapter 5, a characterization of the error profile of inexact operators has been proposed. This method allows deriving circuit error metrics to feed the application quality metric evaluation block (Block 2) in Figure 5.1. Our contribution in Chapter 5 allows determining metrics as the Mean Error Distance (mean ED), the Error Rate (ER) or the Maximum Error Distance (maximum ED). The aim of the contribution presented in this chapter, is to use these circuit error metrics to predict the impact of approximation errors on the application quality metric. Indeed, the impact of the induced errors on the quality at the output of the application has to be quantified and framed to ensure its behavior despite the induced approximations.

The challenge when including inexact operators in an application, is to evaluate the impact of the approximation on the QoS λ at the output of the application. Potential approximations have to be evaluated to choose the best inexact operators with respect to the different constraints on the application implementation. The Approximation Design Space Exploration (ADSE) is large and requires a fast simulation to evaluate the approximation impact on QoS. Indeed, several parameters can vary when selecting an approximation perspective, as for instance the operation replaced, the operator chosen, as well as parameters set to adjust the precision of the simulation. For instance, the carry chain length is a user-defined parameter for the Almost Correct Adder (ACA), and has an impact on the quality and the energy savings of the approximation. For the Carry-Cut Back Adder (CCBA) presented in [CCSE18], the positions of the cuts are also user-defined parameters to trade-off the quality for the delay. Consequently, to be able to explore the whole design space to fix the different parameters of the inexact operators and find which operations should be approximated, a fast evaluation of inexact operators, that directly links the approximation and the application quality metric, is needed.

As presented in Chapter 3, two families of state-of-the-art approaches exist to link the errors induced by inexact operators and the application quality metric, analytical and simulation-based techniques. Analytical methods provide mathematical expressions of an error metric (mean ED or ER) on inexact operators. The chosen error metric can then be quickly evaluated. For instance, when implementing an approximation based on fixed-point coding, the error metric is the error power. To derive the error power, perturbation theory is used [SB04]. Perturbation theory is however not applicable to inexact operators because this theory builds on the hypothesis that errors are small compared to signal, which is not the case in most inexact operators. Several analytical methods have thus been proposed to analyze the impact of an inexact operator on an application, as Modified Interval Arithmetic (MIA) or Modified Affine Arithmetic (MAA) [HLR11]. However, the number of terms required to propagate the probability mass function (PMF) of an inexact operator suffers from range explosion. For instance, to propagate the PMF through 4 Multiply-accumulate (MAC) operations, 8 million terms are needed [HLR11]. Analytical techniques are completely describing the PMF of an inexact operator but do not give a direct link between the error metric and λ, the application quality metric.

The straightforward method to determine the impact of the error on λ is to perform a functional simulation of the application. A functional simulation of an inexact operator simulates the operator at the logic level. Thus, the simulation time is significantly higher than the one obtained with a classical execution using Central Processing Unit (CPU) native data types, as presented in Chapter 3.

Nevertheless, the strength of simulation-based method is that they are generic, and directly link the approximation to the application output quality. To the best of our

knowledge, no method has been proposed to accelerate the simulation of inexact operators by combining exact computation and statistical methods.

6.2 Modelization of the inexact operator error by a stochastic process

Our proposed simulation technique for inexact operators simplifies the modeling of the approximation error to fasten the simulation of inexact operators within the targeted application. The proposed FnF simulators are set in the design flow presented in Figure 6.1. The ADSE, corresponding to Block 4 in Figure 6.1, aims at finding the best inexact operator for each operation in the application as well as the best parameters for their tuning. For each test of a configuration $\mathcal{C}$, λ the QoS, is evaluated from the results of the fast simulation carried-out with the proposed FnF simulator (Block 3 in Figure 6.1).

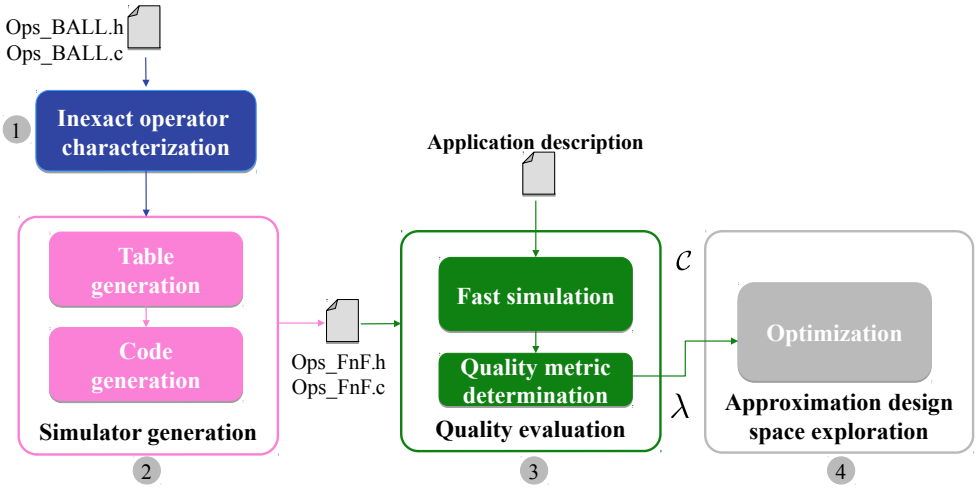


Figure 6.1 – *ADSE flow integrating the proposed FnF simulator to evaluate the QoS of an application implementing inexact operators.*

To take into account the correlation between input values for the FnF_i simulator, output values for the FnF_o simulator, and the approximation errors, the input operand/output set is decomposed into subspaces and a different Pseudo-Random Variable (PRV) is associated to each subspace. Each PRV is defined to mimic the error in terms of ER and Error Distance (ED) generated by the approximation. The proposed simulator is designed to be operator agnostic and is intended to be used during the ADSE process. The FnF simulators are designed to quickly evaluate the impact of different approximations at the hardware level on the QoS of an application.

The FnF simulators use pre-computed tables built in Block 2. The generation of the tables requires the knowledge of the approximation error statistics. These statistics are provided by the characterization process corresponding to Block 1 in Figure 6.1. Analytical techniques [MHH+17], Monte-Carlo simulations [HL11], exhaustive simulations or the characterization method proposed in Chapter 5 are the different alternatives to obtain

the statistics required in Block 1. In the rest of the Chapter, the statistics on the error have been derived from exhaustive simulations for accuracy and generic purpose. The required statistics on the error as the ER, mean ED or maximum ED have been computed by simulating all the possible input values of the inexact operator.

Let's consider an inexact operator $\hat{\diamond}$ whose input operands $x \in I_x = [\underline{x}; \overline{x}]$ and $y \in I_y = [\underline{y}; \overline{y}]$ are encoded on N_x and N_y bits respectively. $\underline{x}$ and $\overline{x}$ represent the minimum and maximum values of x (same for y). The accurate original operator is $\diamond$. The set of all the possible input values for the operators $\diamond$ and $\hat{\diamond}$ is $\mathcal{I} = I_x \times I_y$ and is composed of $2^{N_x+N_y}$ elements. The output of the accurate operator $\diamond$, $z = x \diamond y$ is encoded on N_z bits. The set of all the possible output values for the accurate operator $\diamond$ is $\mathcal{O}$. For an addition or subtraction, the output set $\mathcal{O}$ is composed of $2^{\max(N_x,N_y)+1}$ values and for a multiplier $\mathcal{O}$ is composed of $2^{N_x+N_y}$ values.

Our method reduces the software simulation time of an inexact operator by replacing the BALL simulation of $\hat{\diamond}$ by the accurate version of the operator $\diamond$ plus a PRV $\tilde{e}$ whose statistical characteristics are computed from the error generated by $\hat{\diamond}$, as presented in Equation 6.1 and in Figure 6.2. According to the ER f determined during the operator characterization phase, 0 or the error e is added to $x \diamond y$.

$$x\hat{\diamond}y \Leftrightarrow x \diamond y + \tilde{e} \tag{6.1}$$

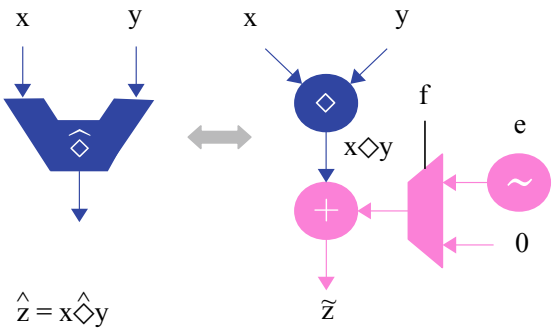


Figure 6.2 – *Statistical equivalence between BALL and FnF simulation of* $x\hat{\diamond}y$.

In the proposed approach, the error due to the approximation is modeled by a stochastic process whose features are determined with an operator characterization phase. The operator characterization phase differs whether the simulator is built on the input or on the output values of the operator.

To determine the accurate error of approximation $\hat{e} = x \diamond y - x\hat{\diamond}y$, a table addressed by x and y can be considered to reproduce the exact behavior of the inexact operator. Nevertheless, this table is composed of $2^{N_x+N_y}$ elements. The amount of memory to store this table is prohibitively large even for small values of N_x and N_y. The proposed simulators are using smaller tables storing the characteristics of the approximation error modeled by a PRV $\tilde{e}$.

6.2.1 PRV to model the error for the input-based FnF_i simulator

To reduce the size of the table storing the error characteristics while avoiding a coarse error modeling, the input set $\mathcal{I} = I_x \times I_y$ is decomposed in subspaces $S_{ij} = I_{xi} \times I_{yj}$ such that $\bigcup_{i,j} S_{ij} = \mathcal{I}$, the input space is entirely covered by the subspaces. Since the ER is not equal

to 1 in each subspace S_{ij}, our method determines a pseudo-random variable PRV $\tilde{e}_{ij}$ for each S_{ij} with statistical characteristics provided by the inexact operator characterization phase (Block 1 in Figure 6.1). It is important to note that our FnF simulator can take as input the statistics on the error provided by an analytical analysis as the one proposed by Mazahir et al. [MHH+17], by exhaustive or by Monte-Carlo simulations. Our simulator does not output the exact value of the approximate operation $x \diamond y + \widehat{e}$ but generates the error $\tilde{e}$ with the same statistical characteristics as the error at the output of the approximate operator. Depending on the ER f_{ij} in S_{ij}, either 0 or e_{ij} is added to $x \diamond y$.

The size of the subspaces S_{ij} controls the modeling grain and is embodied by the user-defined fuzziness degree F. F has an impact on the accuracy of our FnF simulator. In the case of the FnF simulator built on the input values of the inexact operator, the input operands sharing the same $N - F$ Most Significant Bits (MSBs) are grouped in the same subspace S_{ij}, and are then associated to the same PRV $\tilde{e}_{ij}$, hence the error on the result of the simulation. The larger F is, the more different outputs of the simulated operator are summarized with the same PRV $\tilde{e}_{ij}$, thus increasing the simulation inaccuracy.

Algorithm 2 FnF$_i$ Simulation of $x\widehat{\diamond}y$

```
1:  procedure FnF_◇̂(x, y, N, F)
2:      x_0 = x >> F                                  ▷ Pre-process operands
3:      y_0 = y >> F
4:      k = T_idx[x_0][y_0]
5:      if k == 0 then                                ▷ Error-free test
6:          return x ◇ y
7:      else                                          ▷ Pseudo-random number generation
8:          M_1 = 2^F − 1
9:          (a_ij, b_ij, f_ij) = T_err[k]
10:         u_ij = generatePRNumber(M_1, x, y)
11:         e_ij = generateError(u_ij, f_ij, a_ij, b_ij)
12:         return x ◇ y + e_ij
13:     end if
14: end procedure
```

Algorithm 2 details the proposed simulation process for $N_x = N_y = N$ and Figure 6.3 illustrates the flow of our simulation. The first step of the FnF$_i$ simulation is to *pre-process* the input operands x and y by extracting their $N - F$ Most Significant Bits (MSBs) (lines 2-3). This leads to the values x_0 and y_0 respectively. x_0 and y_0 indicate to which subspace S_{ij} x and y belong to. The index $k = \mathcal{T}_{idx}[x_0][y_0]$ indicates if $x\widehat{\diamond}y$ may generate an error. $\mathcal{T}_{idx}$ is a precomputed table that indicates if an error may occur in S_{ij}. If k is equal to zero, no error is generated and the accurate version of $x \diamond y$ is directly returned, thus avoiding any simulation time overhead. If k is different from zero, a set of statistical characteristics (a_{ij}, b_{ij}, f_{ij}) is retrieved from table $\mathcal{T}_{err}$ (line 10) and describes the PRV $\tilde{e}_{ij}$. According to the parameter f_{ij}, the error value e_{ij} is finally generated. The pre-computed table $\mathcal{T}_{err}[k]$ stores the sets of statistical characteristics for each S_{ij} where an error occurs. Section 6.3 presents the generation of the two abovementioned tables. The error value e_{ij} is computed with the following expression:

$$e_{ij} = p_{ij} \cdot (a_{ij} \cdot u_{ij} + b_{ij}) \tag{6.2}$$

with u_{ij} a uniform random variable and p_{ij} a random boolean variable whose distribution follows a Bernouilli law.

Pseudo-random variable generation The random variable p_{ij} is equal to 1 with a probability f_{ij} and to 0 with a probability $1 - f_{ij}$. The random variable p_{ij} is obtained from the random variable u'_{ij} which corresponds to variable u_{ij} uniformly distributed in the interval $[0, 1]$ presented in Equation 6.3, after scrambling (operator S).

$$p_{ij} = \begin{cases} 1 & \text{if } u'_{ij} < f_{ij} \\ 0 & \text{else.} \end{cases} \tag{6.3}$$

In the FnF$_i$ simulator, p_{ij}, is generated from the F Least Significant Bits (LSBs) of the input operands. Indeed, the LSBs of a signal can be considered as a white random additive noise non-correlated with the input signal as derived by Widrow [WK08]. The F LSBs of x and y are concatenated and finally scrambled by a `xor` operation with a constant K. The purpose of these operations is to map the input operands $(x, y) \in I_x \times I_y$ to $p_{ij} \in [0; 2^{2F}[$ in a bijective way.

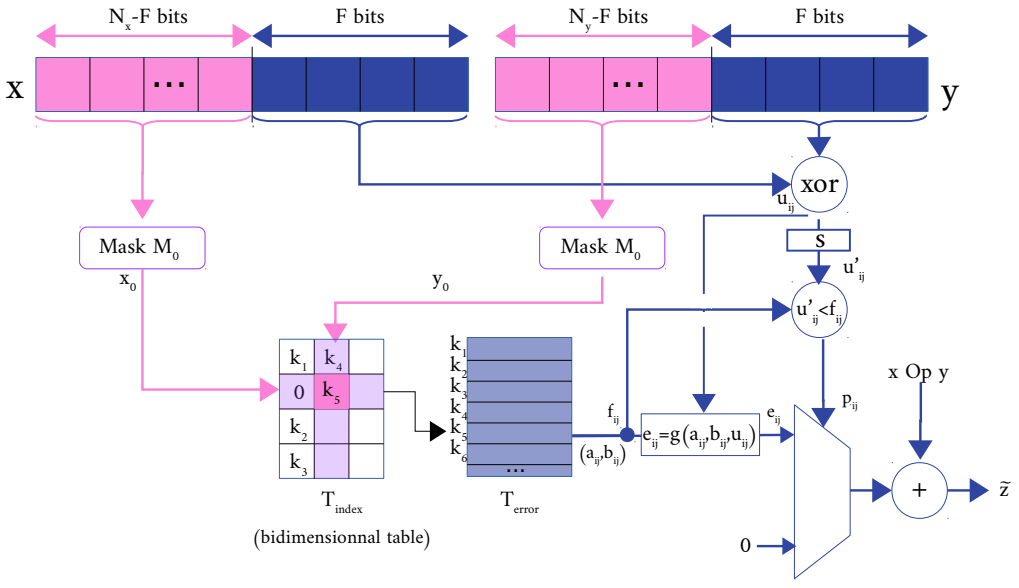


Figure 6.3 – *Graph flow of the proposed FnF$_i$ simulation.*

Finally, an asset of our simulator is that the execution time depends on the number of errors committed by the original inexact operator. The less errors an operator generates, the faster our FnF simulator is, because lines 8-11 of Algorithm 2 can be skipped.

6.2.2 PRV to model the error from the output-based FnF$_o$ simulator

In the case of the FnF simulator built on the output values of the inexact operator, the output set $\mathcal{O}$ is decomposed in 2^{N_z-F} subspaces $\mathcal{O}_i$. The fuzziness degree F also impacts the accuracy of the modelization, the size of the subspaces and the simulation time. For each subspace $\mathcal{O}_i$, a PRV $\tilde{e}_i$ is used to model the error within the subspace $\mathcal{O}_i$. Each subspace $\mathcal{O}_i$ contains the output values $z = x \diamond y$ sharing the same N_z-F MSBs. These output values are modeled by the same PRV $\tilde{e}_i$, as presented in Figure 6.4. The bigger F is, the bigger the subspaces $\mathcal{O}_i$ are and consequently the more information are summarized within a single PRV $\tilde{e}_i$.

The statistical characteristics of the PRV $\tilde{e}_i$ are stored in a table $\mathcal{T}_{err}$ and the N_z-F MSBs of the variable z are used to address $\mathcal{T}_{err}$ as presented in Figure 6.4. Rather

Algorithm 3 FnF$_o$ Simulation of $x\widehat{\diamond}y$

1: **procedure** $FnF_{\widehat{\diamond}}(x, y, N, F)$
2: $z = x \diamond y$ ▷ Compute accurate operation
3: $z_0 = z >> F$ ▷ Pre-process output
4: $k = \mathcal{T}_{idx}[z_0]$
5: **if** $k == 0$ **then** ▷ Error-free test
6: **return** z
7: **else** ▷ Pseudo-random number generation
8: $M_1 = 2^F - 1$
9: $(a_i, b_i, f_i) = \mathcal{T}_{err}[k]$
10: $u_i = \text{generatePRNumber}(M_1, z)$
11: $e_i = \text{generateError}(u_i, f_i, a_i, b_i)$
12: **return** $x \diamond y + e_i$
13: **end if**
14: **end procedure**

than indexing this table with the operator inputs, as in the FnF$_i$ simulator, the proposed approach uses the output values to index the table $\mathcal{T}_{err}$. Contrary to the FnF$_i$ simulator, the table storing the error characteristics is a 1-dimension table since it is only indexed with the output values z.

Numerous input combinations do not generate an error at the output of the approximate operator $\widehat{\diamond}$. Let f_i be the frequency of error occurrence in the subspace $\mathcal{O}_i$. The accurate error $\widehat{e}$ in $\mathcal{O}_i$ is equal to 0 with a probability of $1 - f_i$. To model the error committed in the subspace $\mathcal{O}_i$, the error value e_i is generated with the Equation 6.4.

$$e_i = p_i.\,(a_i.u_i + b_i) \tag{6.4}$$

In Equation 6.4, u_i represents a uniform random variable and p_i a random boolean variable whose distribution follows a Bernoulli law. The random variable p_i is equal to 1 with a probability f_i and to 0 with a probability $1 - f_i$. The random variable p_i is obtained from the random variable u'_i which corresponds to variable u_i uniformly distributed in the interval $[0, 1]$ presented in Equation 6.5, after scrambling (operator S). The variables (a_i, b_i) are the coefficient of the affine form used to compute an error value with a pre-computed amplitude.

$$p_i = \begin{cases} 1 & \text{if } u'_i < f_i \\ 0 & \text{else.} \end{cases} \tag{6.5}$$

During the inexact operator error characterization phase to build the proposed simulator, for each subspace $\mathcal{O}_i$, the characteristics of the accurate error of approximation $\widehat{e}_i$ generated by the inexact operator are extracted. For each $\mathcal{O}_i$ in $\mathcal{O}$, the error values $\widehat{e}_i$ are computed for the input combinations (x, y) such that $z = x \diamond y \in \mathcal{O}_i$. The triplet (a_i, b_i, f_i) describes the PRV $\tilde{e}_i$ and is used to compute the error e_i. Then, from these error values in $\mathcal{O}_i$, the error amplitude represented by (a_i, b_i) and the threshold f_i to generate an error with the same frequency of error occurrence are computed and stored in the table $\mathcal{T}_{err}$.

Algorithm 3 presents the simulation process of $x\widehat{\diamond}y$ and Figure 6.4 illustrates the flow of the FnF$_o$ simulation. Two pre-computed tables are required, $\mathcal{T}_{idx}$ and $\mathcal{T}_{err}$. $\mathcal{T}_{idx}$, of size $2^{N_z - F}$, is used to know if an error occurs in $\mathcal{O}_i$. The table $\mathcal{T}_{err}$ stores the statistical characteristics (a_i, b_i, f_i) of the different PRV $\tilde{e}_i$. The first step to simulate $x\widehat{\diamond}y$ with the FnF simulator built on the output values is to compute the exact value $z = x \diamond y$ (line 2).

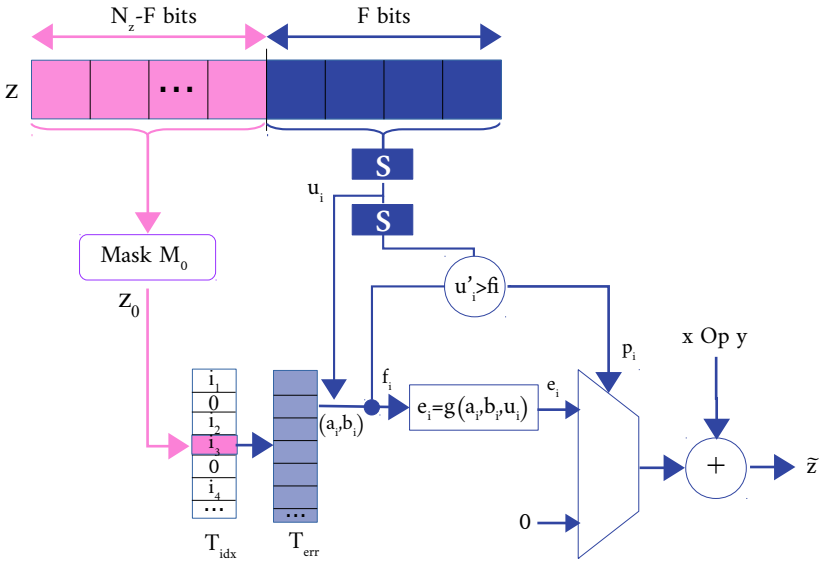


Figure 6.4 – *Graph flow of the proposed FnF_o simulation.*

Then, the $N_z - F$ MSBs of z are extracted, leading to the value z_0 (line 3), addressing table T_{idx}. The value z_0 indicates which subspace $\mathcal{O}_i$ the output value z belongs to (line 4).

The value $\mathcal{T}_{idx}[z_0]$ indicates if $x\hat{\diamond}y$ may generate an error. If $\mathcal{T}_{idx}[z_0]$ is equal to zero, no error is generated and the exact version of the arithmetic operator already computed z is used, thus avoiding any supplementary processing (lines 5-6). Otherwise, $\mathcal{T}_{idx}[z_0]$ gives the index k to address the second table $\mathcal{T}_{err}$ and allows retrieving the parameters a_i, b_i and f_i of the PRV $\tilde{e}_i$ (line 9). The error e_i is finally generated from the Equation 6.4 (line 11).

Like in the FnF_i simulator, to generate the uniform random variable u_i used to compute the error $\tilde{e}_i$, the LSBs of the accurate output value z are considered. The LSBs of z are then `xor`ed with a constant K to scramble it. The obtained result is the uniform random variable u_i (line 10).

The C code developed to implement the proposed approach has been optimized to waste the least cycles possible when simulating operands that do not generate any error.

6.3 Automated simulator construction

Our main contribution is to reduce the simulation time to get the value $x\hat{\diamond}y$ with the two error models detailed in Section 6.2. The error modeling is simplified using the tables $\mathcal{T}_{idx}$ and $\mathcal{T}_{err}$ to store the characteristics of the Pseudo-Random Variables (PRVs) used to compute the output value. The generation of these tables is done only once for each operator and off-line.

For both simulators, Table $\mathcal{T}_{idx}$ indicates if the inexact operation does or not generate an error. For FnF_i, the size of $\mathcal{T}_{idx}$ is $2^{N_x+N_y-2F}$. The $N_x - F$ and $N_y - F$ MSBs of the operands are extracted and operands obtaining the same extracted value are mapped to the same subspace S_{ij} in $\mathcal{T}_{idx}$. Equation 6.6 presents the computation of $\mathcal{T}_{idx}$.

$$\mathcal{T}_{idx}[i,j] = \begin{cases} 0 & \text{if } \forall x,y \in S_{ij},\ x\hat{\diamond}y = x \diamond y \\ k & \text{else.} \end{cases} \tag{6.6}$$

For FnF$_o$, the size of $\mathcal{T}_{idx}$ is 2^{N_z-F}. The $N_z - F$ MSBs of the output of the accurate operation are extracted and outputs obtaining the same extracted value are mapped to the same subspace S_i in $\mathcal{T}_{idx}$. Contrary to FnF$_i$, this table is only indexed with z, nevertheless the computation of $\mathcal{T}_{idx}$ for FnF$_o$ is similar to Equation 6.6.

For both simulators, $\mathcal{T}_{err}$ stores the ED characteristics embodied by the affine form (a_{ij}, b_{ij}) and the value f_{ij} corresponding to the ER in the considered subspace. The size of $\mathcal{T}_{err}$ then depends on the number of subspaces where an error occurs.

To generate an error with the same statistical characteristics as the inexact operator, a system with two equations for three unknowns is set. The system has been detailed for the FnF simulator built on the input values, but is similar for the system on the output values. The variables would just be indexed by a single variable i since the tables are 1-dimension tables.

Affine form modeling To allow solving the system, the value p_{ij} computed from the ER f_{ij}, is computed to be equal to the number of errors to generate. We then have the system $\mathcal{S}$ in Equation 6.7 to solve, with f_{ij} representing the ER in S_{ij}, A_{ij}, the maximum ED in S_{ij}, and μ_{ij} the mean ED in S_{ij}.

$$\mathcal{S}: \begin{cases} p_{ij} & = 2^{2*F}(1 - f_{ij}) \\ A_{ij} & = a_{ij} + b_{ij} * \max\limits_{i,j}(p_{ij}) \\ \\ \mu_{ij} * f_{ij} & = \frac{1}{2^{2*F}} \cdot (\sum_{p=0}^{2^{2*F}-1} a_{ij} + b_{ij} * p) \end{cases} \tag{6.7}$$

$\mathcal{S}$ is developed to extract a_{ij} and b_{ij} in Equation 6.8. Indeed, the maximum value of the pseudo-random number p_{ij} is $2^{2F} - 1$ and the third equation of the system can be separated to highlight an arithmetic sequence whose first term is p_{ij} and common term 1.

$$\begin{cases} b_{ij} & = \frac{2^{2F+1}\mu_{ij} f_{ij} - 2A_{ij}(2^{2F} - 1 - p_{ij})}{(2^{2F} - 1 - p_{ij})(2^{2F} - 3 + p_{ij} - 2^{2F+1})} \\ \\ a_{ij} & = A_{ij} - b_{ij}(2^{2F} - 1) \end{cases} \tag{6.8}$$

Constant form modeling The error e_{ij} can also be modeled by only a constant value if b_{ij} is set to zero. This reduces the complexity at the expanse of a loss in the accuracy of our simulator. In this case, the value p_{ij} is computed for each S_{ij} by equalizing the mean ED of $\hat{\diamond}$ with the mean ED generated by the FnF simulation of $\hat{\diamond}$, as presented in Equation 6.9.

$$\mu_{ij} * f_{ij} = a_{ij} * (1 - \frac{p_{ij}}{2^{2*F}}) \tag{6.9}$$

Thus, the value p_{ij} is equal to:

$$p_{ij} = 2^{2*F} \cdot (1 - \frac{\mu_{ij} \cdot f_{ij}}{a_{ij}}) \tag{6.10}$$

Besides, an inexact operator can be designed to generate errors always lower or equal to $x \diamond y$. In this case, the error e_{ij} cannot be greater than the value of the operation $x \diamond y$, since the output of the simulation is $x \diamond y + e_{ij}$. The generated error e_{ij} is then equal to $\max(-\min\limits_{x,y \in S_{ij}}(x \diamond y), A_{ij})$. The maximum ED is not always generated. Consequently, to keep the same mean ED, the threshold p_{ij} has to be lower to generate errors more often.

The worst possible case is obtained when e_{ij} is always equal to $x \diamond y$ in S_{ij}: in this case, the value of t_{ij} is presented in Equation 6.11. Else, e_{ij} is always equal to A_{ij}, the maximum ED in S_{ij}.

$$p_{ij} = 2^{2*F} \cdot (1 - \frac{\mu_{ij} \cdot f_{ij}}{-\min\limits_{x,y \in S_{ij}} (x \diamond y)}) \tag{6.11}$$

Example on 3-bit operator The construction of the simulator from the table of error amplitude values is illustrated for FnF$_i$ and $b_{ij} = 0$ in Tables 6.1 and 6.2. Table 6.1 indicates the error amplitude values, that is to say $z - \widehat{z}$, for an illustrative inexact arithmetic 3-bit operator depending on the input operands x and y. If the number of Fuzzy bits is equal to 1, the input operands sharing the same $3 - 1 = 2$ MSBs are mapped to the same subspace, as indicated with colors in Table 6.2.

		x **operand values**							
	$\widehat{e}$	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
y **operand values**	**0**	0	0	0	0	0	0	0	0
	1	0	0	0	-4	0	0	0	-4
	2	0	0	0	0	0	0	-8	-8
	3	0	-4	0	0	0	-4	-8	-8
	4	0	0	0	0	-8	-8	-8	-8
	5	0	0	0	-4	-8	-8	-8	-12
	6	0	0	-8	-8	-8	-8	-8	-8
	7	0	-4	-8	-8	-8	-12	-8	-8

Table 6.1 – *Illustration of the proposed method: table of error amplitude values* $\widehat{e} = x\widehat{\diamond}y - x \diamond y$.

		x_0 **values**							
	$\widehat{e}$	**0**		**1**		**2**		**3**	
y_0 **values**	**0**	0	0	0	0	0	0	0	0
		0	0	0	-4	0	0	0	-4
	1	0	0	0	0	0	0	-8	-8
		0	-4	0	0	0	-4	-8	-8
	2	0	0	0	0	-8	-8	-8	-8
		0	0	0	-4	-8	-8	-8	-12
	3	0	0	-8	-8	-8	-8	-8	-8
		0	-4	-8	-8	-8	-12	-8	-8

Table 6.2 – *Illustration of the proposed method: table of subspaces.*

To build the table $\mathcal{T}_{idx}$, each subspace is analyzed and if all the error amplitude values in the subspace are equal to zero, the corresponding value in $\mathcal{T}_{idx}$ is zero, else it contains the index to retrieve the subspace information in $\mathcal{T}_{err}$. For instance, the value of $\mathcal{T}_{idx}$ for the subspace $\mathcal{S}_{00}$ is zero, while for the subspace $\mathcal{S}_{23}$ it is 8. To build the table $\mathcal{T}_{err}$, the ER f_{ij}, maximum error amplitude A_{ij} and mean error amplitude μ_{ij} are computed for each subspace in which an error may occur. For instance, for $\mathcal{S}_{23}$, the ER is equal to 1, the

maximum error amplitude is equal to -12 and the mean error amplitude is equal to -9. The corresponding tables $\mathcal{T}_{idx}$ and $\mathcal{T}_{err}$ are indicated in Tables 6.3 and 6.4 respectively.

k	**0**	**1**	**2**	**3**
0	0	1	0	2
1	3	0	4	5
2	0	6	7	8
3	9	10	11	12

Table 6.3 – *Illustration of the proposed method: table $\mathcal{T}_{idx}$.*

k	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
$\boldsymbol{\mu_{ij}}$	-4	-4	-4	-4	-8	-4	-8	-9	-4	-8	-9	-8
$\mathbf{A_{ij}}$	-4	-4	-4	-4	-8	-4	-8	-12	-4	-8	-12	-8
$\mathbf{f_{ij}}$	0.25	0.25	0.25	0.25	1	0.25	1	1	0.25	1	1	1

Table 6.4 – *Illustration of the proposed method: table $\mathcal{T}_{err}$.*

6.4 Experimental study

The FnF simulators propose to simulate an inexact operator using a simple error model. To evaluate the impact on the simulation of the proposed method, four points have to be highlighted

1. The time savings offered by the simulators for the simulation of a single operation presented in Section 6.4.1.
2. The simulation time/accuracy trade-off of the proposed simulation depending on the fuzzyness degree F, detailed in Section 6.4.2.
3. The overhead in terms of computation time and memory footprint due to the approximate operator characterization phase, presented in Section 6.4.3.
4. The accuracy of the QoS evaluation compared to the time savings in an ADSE process, extended in Section 6.4.4.

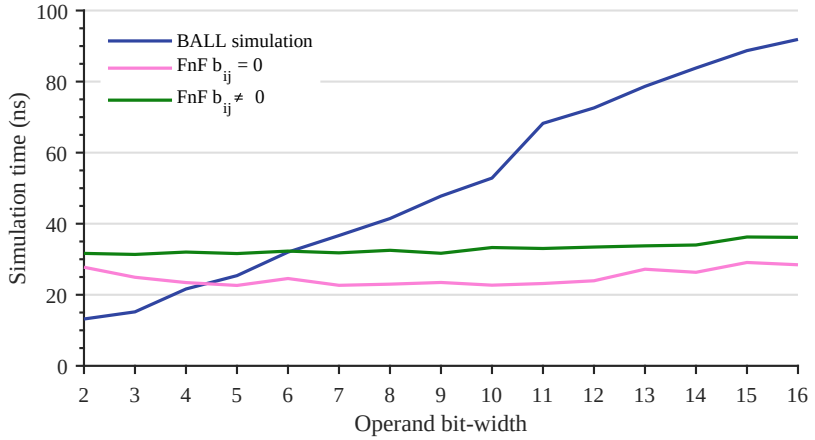
The results have been obtained on a processor Intel i7-6700 with 32GBytes of RAM.

6.4.1 Simulation time savings for the FnF simulators

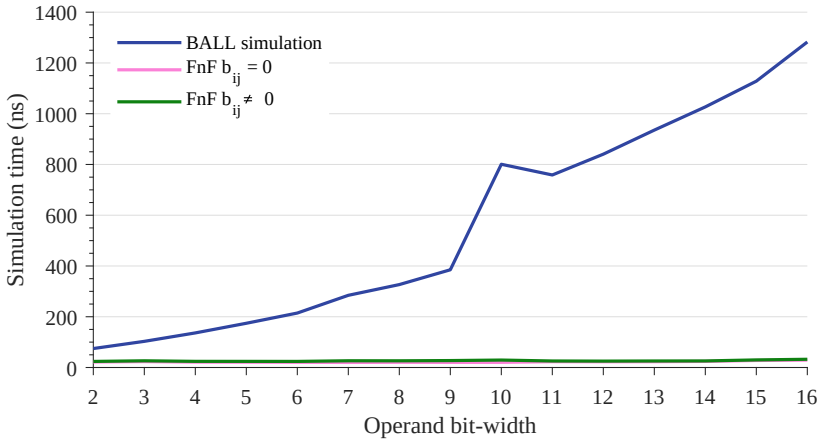
To quickly test the impact of an inexact operator in an application, the easiest solution is to simulate the application in software with BALL simulation. The simulation time of our FnF simulators is compared with the software BALL simulation time obtained with the C code from the App Test framework [BSM17] for a single operation and various input operand bit-width. The FnF simulators have been built if the error is modeled by an affine form or a constant value, which impacts the accuracy of the simulation. The results are presented for two types of inexact operators: the ACA and the Lower-Error Fixed-Width Multiplier, called the Approximate Array Multiplier (AAM) in the rest of the Chapter and presented in [VWF00]. Those operators have been chosen for the experimental study since they are between the extremum presented in Table 3.2 in Chapter 3.

FnF simulator built on the input values

Figure 6.5(a) represents the obtained simulation times for the ACA and Figure 6.5(b) for the AAM. The time for simulating a single operation is represented depending on the input operand bit-width. The obtained simulation times have been **averaged** for all possible configurations of carry-chain length (for the ACA) and all possible values of F for both operators. A BALL simulation of the ACA on 16-bit takes 300 more time than classical execution of the exact operator with native data types, and the BALL simulation of the AAM, for the multiplication of two 16-bit operands takes 4200 more time than a simulation with native data types.



(a) ACA inexact operator.



(b) AAM inexact operator.

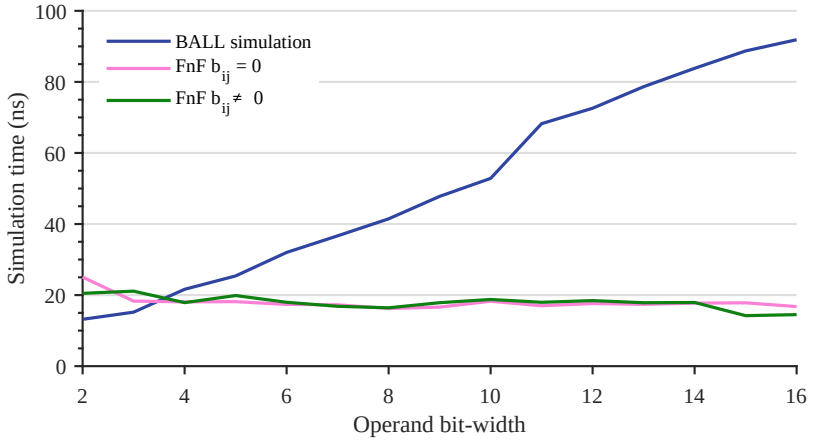
Figure 6.5 – *Simulation time for the FnF$_i$ and the BALL simulation of one operation for two inexact operators.*

The simulation times for the FnF$_i$ simulator are represented for $b_{ij} \neq 0$ and $b_{ij} = 0$, the affine modeling being slightly longer to simulate than when $b_{ij} = 0$. For the ACA, the FnF simulation is always faster than the BALL one with operand bit-width greater than 6. On a single 16-bit addition, the BALL simulation takes 3.5 more time than the FnF simulation. The gain for this operator is reasonable since the design of the ACA is quite simple and can easily be reproduced with C-code. However, when it comes to the AAM, whose design is much more complex than the one of the ACA, the FnF simulation

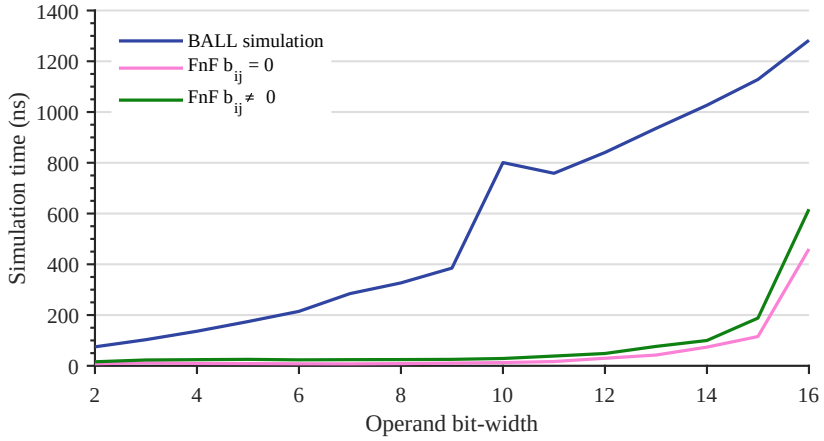
is always faster and the BALL simulation takes 44 more time than the FnF simulation on a single 16-bit operation.

FnF simulator built on the output values

Figure 6.6(a) represents the obtained simulation times for the ACA and Figure 6.6(b) for the AAM. The time for simulating a single operation is represented depending on the input operand bit-width. The obtained simulation times have been **averaged** for all the possible configurations of carry-chain length (for the ACA) and all the possible values of F for both operators.



(a) ACA inexact operator.



(b) AAM inexact operator.

Figure 6.6 – *Simulation time for the FnF_o and the BALL simulation of one operation for two inexact operators.*

The simulation times for the FnF_o simulator are represented for $b_{ij} \neq 0$ and $b_{ij} = 0$. For the ACA, the FnF simulation is always faster than the BALL one with operand bit-width greater than 4. The affine modeling leads to similar simulation times than with the modeling with a constant error value. On a single 16-bit addition, the BALL simulation takes 5.5 more time than the FnF simulation with $b_{ij} = 0$ and 6.3 more time than the FnF simulation with $b_{ij} \neq 0$, which is counter intuitive. The gain for this operator is reasonable since the design of the ACA is quite simple and can easily be reproduced with

C-code. When it comes to the AAM, whose design is much more complex than the one of the ACA, the FnF simulation is always faster than the BALL simulation. Nevertheless, because of the size of the required tables to build the simulator, leading to more cache misses, the simulation times are increasing from $N \geq 10$. The maximum time savings are obtained when $N = 10$ and are equal to 63 if $b_{ij} = 0$ and to 28 if $b_{ij} \neq 0$. According to the simulation time, from $N = 10$ for the AAM operator, the FnF simulator built on the input values offer larger savings than the FnF simulator built on the output values.

Comparison of both simulators

Both simulators offer moderate time gains according to the simulation of the operator ACA since the internal logic structure of the ACA is simple to reproduce with a C code, the approximation simply consisting in cutting the carry-chain propagation in an addition. The simulation time gains are up to 6.3× compared to the BALL simulation. The FnF_o simulator is faster than the FnF_i since the tables to store are much smaller, and less operations are needed when computing a simulation with no errors.

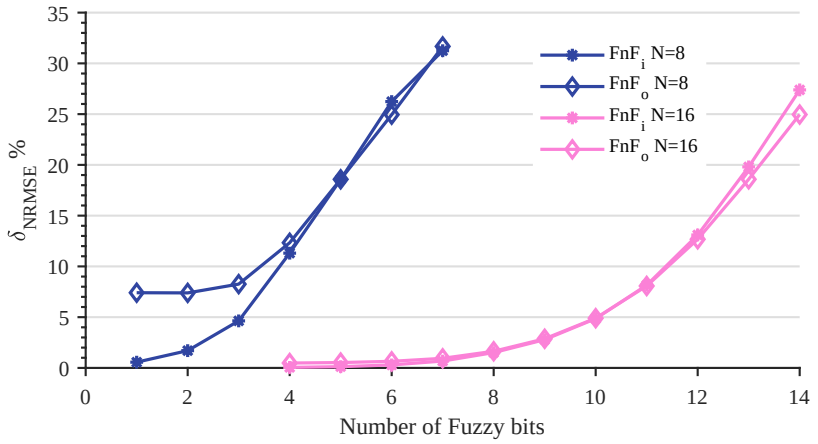
The FnF_i simulator is particularly useful for the Design Space Exploration (DSE) of an algorithm with more complex operators, as for instance the AAM. Indeed, for the simulation of a 16-bit AAM, the FnF_i was 44× faster than the BALL simulation. Nevertheless, the behavior of the FnF_o with a multiplier is more complex, as presented in Figure 6.6(b). With the AAM, the FnF_o simulation time gains are increasing up to 63× on a 10-bit AAM, to then decrease up to 37 on a 16-bit AAM. The FnF_i offers higher gains for multipliers whose input bit-width is greater or equal to 11, but in both cases, the simulation time savings on a 16-bit multiplier are considerable for the DSE of an application.

6.4.2 Trade-off simulation time/quality

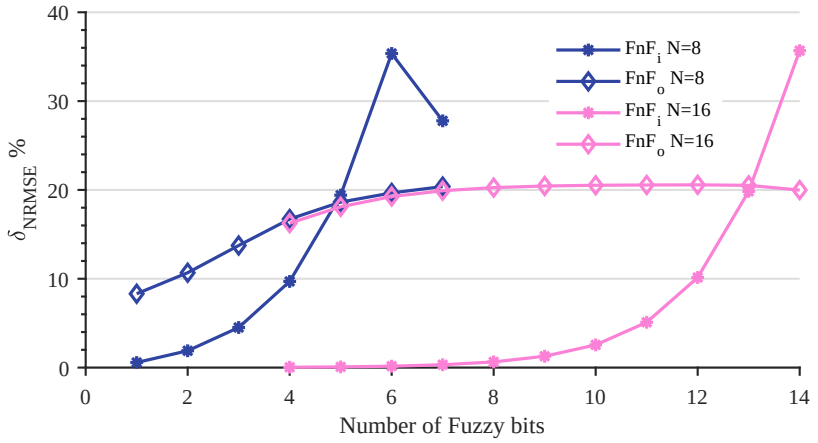
The lower F is, the more accurate the simulation is, and the bigger the tables to store are. The number of Fuzzy bits F embodies a trade-off between the simulation time, the size of the tables to store and the quality of the simulation. The simulation time depends on the number of errors generated by the original approximate operator $\hat{\diamond}$ and on F that impacts the size of the tables and consequently the number of cache misses. The lower F , the slower the FnF simulation. The following results have been obtained with $b_{ij} = 0$ for FnF_i and $b_i = 0$ for FnF_o.

To study the impact of the number of Fuzzy bits F on the simulation output quality, the relative error of Normalized Rooted Mean Squared Error (NRMSE) between the approximation (computed with the BALL simulation) and the FnF_i and FnF_o simulations are presented in Figure 6.7(a) for two operators, the ACA on 8 bits with a carry chain-length cut at 4 and the ACA on 16 bits with a carry chain-length cut at 8. The relative error between both NRMSE is called δ_{NRMSE} and is expressed in percent. For the ACA on 8 bits, δ_{NRMSE} stays under 10% if F is lower or equal to half of the input bit-width. On the 16-bit ACA, the margin is bigger. Indeed, δ_{NRMSE} stays under 10% until F is equal to 12. The supplementary error due to the proposed model is acceptable for a number of Fuzzy bits F between 50 to 75 % of the total operator word-length. As shown in the next section, this leads to small tables to store for our approach.

The relative error of NRMSE between the approximation, computed with the BALL simulation, and the FnF_i and FnF_o simulations are presented in Figure 6.7(b) for two operators, the AAM on 8 bits and the AAM on 16 bits. The relative error between both NRMSE is called δ_{NRMSE} and is expressed in percent. The choice of the simulator has a strong impact on the quality contrary to the case of the operator ACA. For instance,



(a) ACA inexact operator, carry chain-length cut at $N/2$.



(b) AAM inexact operator.

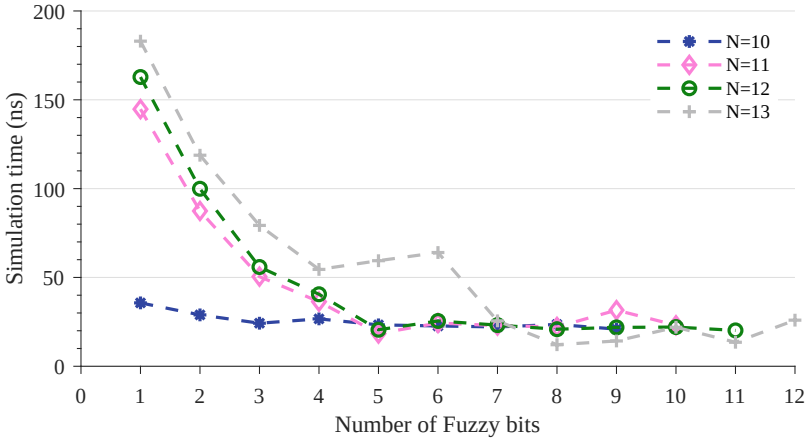
Figure 6.7 – *Simulation time for the FnF_o and the BALL simulation of one operation for two inexact operators.*

for the AAM on 8 bits, δ_{NRMSE} stays under 10% if F is lower or equal to half of the input bit-width for FnF_i. For FnF_o, the induced error evolves in 9 to 18% if F is lower or equal to half of the input bit-width. The simulator built on the input is more interesting in this case. Nevertheless, this tendancy gets reversed from $F = 5$ bits. The behavior of the relative error of NRMSE is similar on the 16-bit AAM. δ_{NRMSE} stays under 10% until F is equal to 12 for the simulator FnF_i, and between 16 and 21% for the FnF_o. The conclusion is similar, it is preferable to use the simulator built on the input values up to $F = 12$. Afterwards the induced error becomes higher for the FnF_i than for the FnF_o.

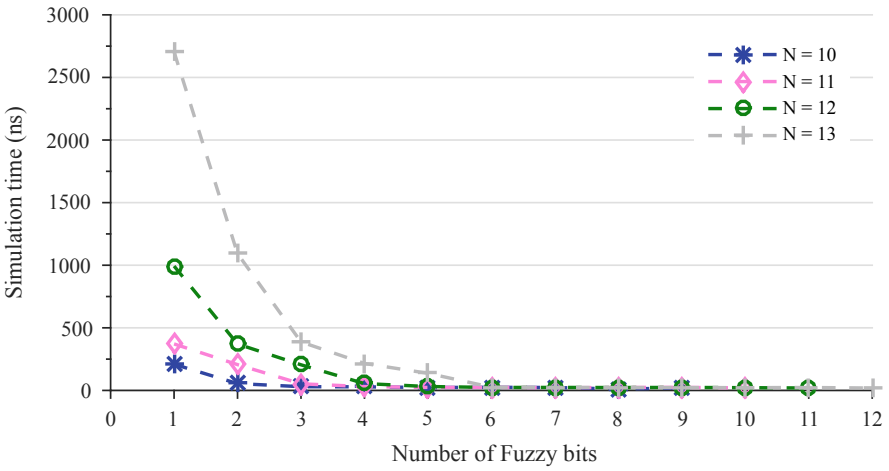
The number of Fuzzy bits F also has an impact on the simulation time as shown in Figure 6.8(a) for the inexact operator ACA and Figure 6.8(b) for the inexact operator AAM. In both cases, the simulator is built on the input values. The FnF simulation time is represented for different input bit-widths, depending on the number of Fuzzy bits F. The highest F, the smallest the tables and the least cache misses.

For instance, for $N = 13$ and for the inexact operator AAM, F has to be at least equal to 3 to be faster than the BALL simulation (simulation time gain of 2.4) and the maximum simulation time gain if $F = N-1$ is equal to 49.2. For $N = 13$ and for the inexact operator

ACA, F has to be at least equal to 4 to be faster than the BALL simulation (simulation time gain of 1.44) and the maximum simulation time gain for $F = N - 2$ is equal to 5.82.



(a) ACA inexact operator.



(b) AAM inexact operator.

Figure 6.8 – *FnF_i simulation time depending on the number of Fuzzy bits F.*

6.4.3 Inexact operator characterization phase

Compared to the BALL simulation, a pre-processing phase is required to build the tables $\mathcal{T}_{err}$ and $\mathcal{T}_{idx}$ for each operator. The inexact operator error $\widehat{e}_{ij}$ or $\widehat{e}_i$ for the FnF_i and FnF_o respectively, is characterized and the statistical characteristics of the PRV $\tilde{e}_{ij}$ or $\tilde{e}_i$ are computed and stored in $\mathcal{T}_{idx}$ and $\mathcal{T}_{err}$. In Table 6.5, the memory footprint to store the tables $\mathcal{T}_{err}$ and $\mathcal{T}_{idx}$ for an average value of F that is to say $F = \lfloor \frac{N}{2} \rfloor$ for FnF_i and $F = \lfloor \frac{N_z}{2} \rfloor$ for FnF_o is provided, as well as the time required to build the tables. The characterization step is done once off-line for each new operator. The characterization process used in this experimental set-up is an exhaustive characterization method and the obtained results are given according to the worst-case characterization in terms of time. To reduce the overhead of this step, Monte-Carlo simulations or the proposed characterization method in Chapter 5 can be used. The FnF_o simulator is longer to build by construction because consecutive output values of an operator are not necessarily belonging to the

same subspace, which induces a bad locality in terms of memory accesses. The results are presented in the case of an exhaustive test of all the value x and y. Nevertheless, for operators with a high operand word-length, an exhaustive test is not possible. Accurate estimation of the statistical parameters a_i, b_i and f_i can be obtained with a limited number of operand values reducing dramatically the time required to build the tables. The main advantage of the FnF$_o$ simulator compared to the FnF$_i$ is the significant reduction of the size of the tables for the adder. For the ACA on 16 bits, the memory footprint is divided by 166 with the FnF$_o$ compared to tables obtained with the FnF$_i$ simulator.

$\hat{\diamond}$	**N**	**t (FnF$_o$)**	**t (FnF$_i$)**	**Mem (FnF$_o$)**	**Mem (FnF$_i$)**
AAM	6	$<$ 1ms	$<$ 1ms	764B	1KiB
	7	$<$ 1ms	0.2s	3KiB	3KiB
	8	$<$ 1ms	0.3s	45KiB	3KiB
	10	0.4s	3.5s	12KiB	12KiB
	16	4h20'	137.5s	768KiB	768KiB
ACA	6	$<$ 1ms	$<$ 1ms	192B	1.3KiB
	7,8	$<$ 1ms	$<$ 1ms	384B	4KiB
	10	37s	2.98s	768B	16KiB
	16	4h15'	62.5s	6 KiB	1MiB

Table 6.5 – *Time and Memory overhead to construct the FnF simulator.*

6.4.4 ADSE for a stereo vision application

When designing an application with Approximate Computing (AC), several parameters have to be tuned during the ADSE. During this phase, the application is run several times to measure the quality of the result at the output of the application with the chosen parameters, to finally select the parameters leading to the best performances with respect to the designer's constraints. To give an idea of the intensiveness of this phase, a non exhaustive list of tunable parameters is given hereafter. The percentage of replaced operators, the replaced operations, the inexact operator that is used (ex: ACA or Error-Tolerant Adder Type II (ETAII)), if the operator is tunable, the parameter to tune (ex: carry-chain length for the ACA) and finally, the set of inputs on which the algorithm has to be tested, are tunable parameters.

To illustrate the DSE phase, the stereomatching algorithm presented in Chapter 4 is considered [MPMN14]. This computer vision algorithm outputs a depth map given two rectified input images. AC is particularly interesting for this algorithm since it is computationally intensive. In this experiment, only the most computation intensive kernel of this algorithm is considered for the ADSE. Only taking into account the percentage of operators as well as the operations to replace, but no tuning of the operators, 158 simulations are needed for the DSE. If the replacement of multiplication operations is solely considered, testing all the possible configurations with the BALL simulation takes **8.82 hours**. The simulation using the FnF simulator built on the input values only requires **12.71 minutes**.

This test has been solely done using an input image from a test set extracted from the Middlebury database [SS03]. To validate the approximation on the stereomatching algorithm, the test needs to be done on a complete data set, hence increasing the number of simulations to perform. Several configurations have been tested and are presented in

Table 6.6. The quality metric used to measure the difference between FnF and the BALL simulation is the Structural Similarity Index Measure (SSIM) presented in [WBSS04]. The terms r_{ADD} and r_{MULT} represent the ratio of addition and multiplication which use an inexact operator. The gain $G_t = \frac{t_{BALL}}{t_{FnF}}$ in terms of simulation time of the proposed approach compared to a BALL simulation is provided. The accuracy degradation of the application QoS evaluation is provided through Δ_a that measures the relative difference between the SSIM of both output images obtained with FnF and BALL simulations. The multiply operations have been replaced by the AAM and the additions have been replaced by the ACA with a carry-chain length equal to 7. Both operators are on 16-bit. The most important accuracy degradation for QoS evaluation depends on which operation is impacted by the approximation, but the degradation is always lower than 6%. The FnF simulator leads to an accurate QoS evaluation and allows saving a non negligible simulation time by dividing the simulation time between 1.38 to 87.59.

F	r_{ADD} (%)	r_{MPY} (%)	G_t	Δ_{acc}
5	0	14.3	65.59	$< 0.1\%$
5	0	14.3	66.01	$< 0.1\%$
5	0	14.3	65.71	$< 0.1\%$
5	0	28.6	84.44	$< 0.1\%$
5	0	28.6	83.37	$< 0.1\%$
5	0	28.6	87.59	$< 0.1\%$
8	20	0	1.38	5.69%
5(AAM) - 8(ACA)	20	14.3	52.24	0.578 %

Table 6.6 – *ADSE for the stereomatching algorithm. Simulation time gain G_t and quality degradation Δ_{acc} of the proposed approach compared to a BALL simulation for different configurations r_{ADD} and r_{MULT}.*

6.5 Conclusion

When implementing an AC technique, the link between the errors induced by the approximation and the QoS at the output of the application implementing the approximations has to be done. Analytical techniques fail to provide a generic method to compute the QoS and simulation-based techniques can be preferred. To overcome the long simulation time to measure this loss of quality, in this Chapter, a simulation technique based on error injection has been proposed. Our fast simulation technique avoids to simulate the inexact operator within the application at the logic level. Nevertheless, a thorough characterization of the error induced by the inexact operator has to be done to inject an error with the right characteristics. This characterization can be done with exhaustive simulations, analytical techniques or the characterization method proposed in Chapter 5.

This contribution has been presented in the international conferences GLSVLSI 2018 [BDM18c] and ISCAS 2018 [BDM18a]. In these contributions, two different versions of the fast simulator for inexact arithmetic operators have been proposed. The proposed simulators are used for the DSE of an application implementing inexact operators. The approximation error modeling of inexact operators is simplified to fasten the simulation. Both simulators are demonstrated to induce an acceptable loss of simulation accuracy in exchange for large simulation speedups. So as to reduce the simulation time and the memory overhead to store the simulator, a parameter F called Fuzziness degree can be

tuned. Through the experimental study, it has been shown that a value of 4 for F in case of 8-bit inexact operators, and 12 in case of 16-bit operators was a good compromise between the simulation time, memory overhead, and quality of the simulation.

The contributions presented in Chapter 5 and 6 both cover the spectrum of characterization of the error profile and link with the application quality metric for inexact arithmetic operators.

CHAPTER 8

FAKEER: FAst Kriging-based Error Evaluation for fixed-point Refinement

"Pour explorer le champ des possibles, le bricolage est la méthode la plus efficace."

Hubert Reeves

To convert an application to fixed-point arithmetic, an efficient noise power modelization has been proposed in the previous Chapter. The noise power modelization has been exploited in the fixed-point refinement process to reduce the number of simulations done for each noise power evaluation. In the previous Chapters, inferential statistics were used to infer, from a reduced number of samples, metrics on the approximation error. In the case of inexact arithmetic operators, inferential statistics were used to estimate the Mean Error Distance (mean ED) and the Error Rate (ER) of the approximation error. In the case of finite precision and particularly fixed-point arithmetic, inferential statistics were used to estimate the noise power value at the output of the application. The noise power is an intermediate accuracy metric massively used for fixed-point arithmetic. This Chapter is is continuity of Chapter 7. In Chapter 7, the statistical properties of the noise power have been exploited. The noise power is the second order moment of the error of approximation and inferential statistics can be used to reduce the number of points used to estimate it.

In this Chapter, we have proposed a new method for the estimation of an accuracy metric at the output of an application that can be used during fixed-point refinement. The accuracy metric can be the noise power at the output of the application or the application quality of service (QoS) metric. In particular, during the fixed-point refinement process, numerous configurations of the word-lengths are tested, and the chosen accuracy metric at the output of the algorithm is evaluated for each configuration. Classically, the accuracy metric is evaluated by simulations for each configuration, which makes the refinement process a time-consuming task. In Chapter 7, the time for fixed-point refinement was reduced by using a reduced number of samples for the estimation of the intermediate accuracy metric, the noise power. In this Chapter, we have proposed a new method for accuracy and quality metric inference using kriging, a geostatistical method. In this method, the inference is done using the already computed values of the metric. The concept exploited here is to deduce from the behavior of the metric depending on the

already tested configurations, the value of the metric in an unknown configuration. The number of metric evaluations done with simulations is then reduced.

The motivations for using kriging for metric estimation are exposed in Section 8.1. The method to infer the accuracy or quality metric at the output of an application is presented in Section 8.2. The obtained experimental results on several benchmarks are presented in Section 8.3.

8.1 Introduction

When implementing an Approximate Computing (AC) technique in an application, an optimization problem has to be solved. The implementation cost $\mathcal{C}$ of the application has to be minimized subject to a quality constraint λ_{min}, as depicted in Equation 8.1:

$$\min(\mathcal{C}(\mathbf{e})) \quad \text{subject to} \quad \lambda(\mathbf{e}) > \lambda_{min} \tag{8.1}$$

where $\mathbf{e}$ represents a N_v-length vector of the different sources of approximation. For instance, this problem can be solved during a block-based sensitivity analysis as proposed by Parashar et al. [PRMS10]. In this case, the vector $\mathbf{e}$ represents the noise power values tolerated by each block in an application subject to a quality constraint at the output of the application. This technique can be used before applying fixed-point refinement on each individual block in the case of fixed-point arithmetic conversion. The problem depicted in Equation 8.1 can also be solved for word-length optimization when converting an application to fixed-point arithmetic. In this case, the vector $\mathbf{e}$ is the vector of the word-lengths of the internal variables in the application.

Nevertheless, solving this problem is long and complex. This optimization problem is commonly solved using functional simulation techniques with an arbitrary large predefined input data set. For each tested configuration in vector $\mathbf{e}$, the accuracy metric at the output of the application $\lambda(\mathbf{e})$ is evaluated by simulations. For fixed-point arithmetic, classically, instead of measuring the impact of finite precision on the output QoS, an intermediate metric is used. The quantization noise power P [Men08] is a massively used accuracy metric. The quantization noise power measures the loss of accuracy due to finite precision. The accuracy metric can also directly be the application QoS metric, but is generally harder to evaluate.

To measure the impact of the approximation technique on the considered quality or accuracy metric, it has to be emulated. The simulation time overhead due to the emulation of the approximation combined with the great number of samples to simulate and numerous configurations to test lead to long time to evaluate the metric. The total time t_{opt} for solving this optimization problem is expressed in Equation 8.2.

$$t_{\text{opt}} = N_\lambda \cdot N_\text{S} \cdot t_S \tag{8.2}$$

where N_λ is the number of evaluations of the metric, N_S, the number of samples to simulate for each metric evaluation and t_S, the simulation time of a single sample.

For fixed-point arithmetic, the time for solving this optimization problem has been reduced in the literature with efficient emulation methods, reducing the time t_S, as for instance C++ fixed-point libraries proposed by MentorGraphics [Gra16] or by SystemC [GLMS10]. We have proposed a framework based on inferential statistics to reduce the number of samples to simulate N_S for the intermediate accuracy metric, the noise power evaluation in [BDM19a].

In the proposed contribution, we aim at reducing the number of simulation-based metric evaluations N_λ.

8.2 FAKEER methodology

8.2.1 Related works

Interpolation-based methods have already been proposed to solve the optimization problem in Equation 8.1 for fixed-point refinement. Sedano et al. [Sed12] have proposed an interpolation method based on an optimization method proposed by Han et al. [HEKC01], the `preplanned search`. The `preplanned search` is a greedy optimization algorithm similar to the `min + 1 bit` algorithm, but it takes into account the sensitivity of each variable in the competition between the variables. The different steps for the `preplanned search` are as follows:

1. Search for the minimum vector of uniform word-lengths $\mathbf{w}^u$ to meet the accuracy constraint.
2. From the minimum vector of uniform word-lengths $\mathbf{w}^u$, search for the minimum word-length for each variable $\mathbf{w}^{min}$ to meet the accuracy constraint, while the other variables are set to $\mathbf{w}^u$. The word-lengths are decreased by 1 bit at each iteration.
3. During the search for $\mathbf{w}^{min}$, compute and save the sensitivity s_i for each variable such that $s_i = \lambda(\mathbf{w}_i + 1) - \lambda(\mathbf{w}_i)$.
4. Iterative competition guided by the list of decreasing sensitivities until the accuracy constraint is met.

The method proposed by Sedano et al. reduces the number of simulations during the search for $\mathbf{w}^{min}$ and for the sensitivity of each variable by using interpolation. In this method, the vector $\mathbf{w}^{min}$ is obtained with decrements of b bits at each iteration. The accuracy is measured at each iteration, and the intermediate values are interpolated. The measured values are stored in a matrix M in which line i corresponds to variable i and column j to a word-length equal to j. If $\mathcal{S}$ is the set of measured values, the unknown value $M(i,j)$ can be interpolated as:

$$M(i,j) = \frac{M(i,j-p)\cdot 4^p + \frac{M(i,j+q)}{4^q}}{2} \tag{8.3}$$

where $M(i,j-p)$ and $M(i,j+q)$ are the nearest already measured points for variable i. Nevertheless, a drawback of this technique is that the interpolator is based on the theory on linear and smooth non-linear systems [Par07] as mentionned in [Sed12]. Moreover, this interpolation is used during the search for $\mathbf{w}^{min}$, only considering the contribution on the noise power of a single variable. This type of interpolation is not considering a N_v-dimension hypercube allowing taking into account the contributions on the noise power of all the variables at the same time. Finally, this method is specific to the estimation of the intermediate accuracy metric, the noise power.

Contrary to [Sed12], our method reduces N_λ using kriging, which is compatible with non-linear systems and any type of accuracy or quality metric. Kriging allows interpolating the value of the metric λ in a given configuration $\mathbf{e}$, from previously measured values of λ in other configurations.

8.2.2 Kriging-Based Inference

The objectives of the proposed method are:

1. To reduce the number of simulation-based evaluations N_λ of the accuracy or quality metric at the output of the application for solving the optimization problem in Equation 8.1.

2. To provide an interpolation method not limited to linear and smooth non-linear systems.

3. To provide an interpolation method generic for any metric at the output of the application.

To solve the considered optimization problem, the value of the metric $\lambda(\mathbf{e^i})$ depends on the considered configuration of the approximation sources $\mathbf{e^i}$. In the rest of the chapter, the vector of size N_v, $\mathbf{e^i} = (e_0, e_1, ...e_{N_v})$ represents the configuration i of the different approximation sources. To solve the optimization problem, the number of tested configurations is large. For instance, for fixed-point refinement, in gradient ascent-based methods as *min+1 bit* [Ca01], for each tested configuration, the best direction is searched and for each configuration $\mathbf{e^i}$, N_S samples have to be simulated to compute the accuracy metric $\lambda(\mathbf{e^i})$. For the fixed-point refinement of small signal processing kernels as the Finite Impulse Response (FIR) filter or the Infinite Impulse Response (IIR) filter, the accuracy metric being the noise power and a noise constraint of $-60dB$, the number of accuracy metric evaluations N_λ ranges between 145 and 844 but can be larger than 1000 for more complex kernels with numerous variables to optimize.

If the components of vector $\mathbf{e}$ are forming a hypercube, the different values of vector $\mathbf{e_k}$ for each tested configuration are sampling this N_v-dimension hypercube. The proposed method infers the value of the metric λ in a new sample of the hypercube $\mathbf{e^i}$ from the values of the metric λ already measured on the other samples, instead of evaluating by simulations. The inference is done with kriging, a technique to estimate the value of a random field, in this case λ, in an arbitrary sample $\mathbf{e^i}$ depending on the values of λ already measured in the samples $\mathbf{e^j}$, $j \neq i$.

Kriging description

Geostatistics [Wac14] applies the theory of random functions to spatially distributed data. The goal of geostatistics is to model the behavior of a variable that is evolving in space and/or time, and predict its value in unknown parts of space. Geostatistical methods were first developed for mining, and have been used to estimate complex quantities such as confidence intervals. The geostatistical method implemented to estimate the metric is a simple kriging technique. Kriging is a stochastic spatial interpolation technique, that allows predicting a random field $\lambda(.)$, possibly non-linear, in an arbitrary sample $\mathbf{e^i}$ using the already known values of $\lambda(.)$ in the points $\mathbf{e^0}, ..., \mathbf{e^{i-1}}$. The proposed method relies on two steps:

1. The function indicating the correlation between points depending on their distance is identified from the already known values of $\lambda(.)$ in $\mathbf{e^0}, ..., \mathbf{e^{i-1}}$.

2. The obtained model is used to interpolate the value of $\lambda(.)$ in point $\mathbf{e^i}$.

The random field $\lambda(.)$ is modeled by:

$$\lambda(\mathbf{e^i}) = m + \sum_{k=0}^{i-1} \mu_k \lambda(\mathbf{e^k}) \tag{8.4}$$

where m and μ_k are constant values. The weights μ_k are determined such that the estimator $\lambda(\mathbf{e^i})$ is unbiased and leads to an estimation error of minimal standard deviation.

The first step of the proposed method consists in deriving the function indicating the evolution of the correlation between the measured values of the random field $\lambda(.)$ in samples $\mathbf{e^j}$, $j \neq i$, depending on the distance r between the samples. This function is called the semi-variogram $\hat{\gamma}$. The computation of $\hat{\gamma}(r)$ is detailed in Equation 8.5.

$$\hat{\gamma}(r) = \frac{1}{2|N(r)|} \sum_{N(r)} \{\lambda(\mathbf{e^j}) - \lambda(\mathbf{e^k})\}^2 \tag{8.5}$$

where $N(r) = \{(j,k)$ such that $|\mathbf{e^j} - \mathbf{e^k} = r|\}$ and $|N(r)|$ represents the number of distinct couples (j,k) in the set $N(r)$. $\mathbf{e^i}$ is the sample in which the value of λ has to be infered. From the already measured values of $\lambda(.)$ in $\mathbf{e^j}$, $j \neq i$, the semi-variogram can be computed and identified to a particular type of semi-variogram [Wac14]. This identification allows computing the value $\hat{\gamma}(r)$ for any value of r.

Kriging is an optimal linear estimator and with no bias. The interpolated value of the metric in sample $\mathbf{e^i}$ is noted $\hat{\lambda}(\mathbf{e^i})$ and the real value $\lambda(\mathbf{e^i})$. Kriging gives the interpolated value by computing the weighted average of the available samples leading to an estimation error of minimal standard deviation as presented in Equation 8.7. The methodology for computing the unknown value is:

1. Writing the unknown value $\hat{\lambda}(\mathbf{e^i})$ as a linear combination of the known values as in Equation 8.4.

2. Writing the universality constraint, which indicates that kriging is an unbiased estimator as in Equation 8.6.

3. Writing the optimality constraint, that is to say solving the Equation 8.7.

$$E[\hat{\lambda}(\mathbf{e^i}) - \lambda(\mathbf{e^i})] = 0 \tag{8.6}$$

$$\min(Var[\hat{\lambda}(\mathbf{e^i}) - \lambda(\mathbf{e^i})]) \tag{8.7}$$

The conditions for kriging (optimality and no bias) allow computing the interpolated value $\hat{\lambda}(\mathbf{e^i})$.

Let $\hat{\gamma}_{jk}$ be $\hat{\gamma}(|\mathbf{e^j} - \mathbf{e^k}|)$, where $\mathbf{e^j}$ and $\mathbf{e^k}$ are samples in which the value of λ has been measured. Let $\hat{\gamma}_{ik}$ be $\hat{\gamma}(|\mathbf{e^i} - \mathbf{e^k}|)$, where $\mathbf{e^i}$ is the sample in which the value of λ has to be infered. For clarity, let's denote $\lambda_k = \lambda(\mathbf{e^k})$. If we define two $N+1$-length vectors $\boldsymbol{\lambda}$ and $\boldsymbol{\Gamma_i}$ as:

$$\boldsymbol{\lambda} = \quad (\lambda_0, \lambda_1, ..., \lambda_{N-1}, 0) \tag{8.8}$$

$$\boldsymbol{\Gamma_i} = \quad (\hat{\gamma}_{i\,1}, \hat{\gamma}_{i\,2}, ..., \hat{\gamma}_{i\,N-1}, 1) \tag{8.9}$$

Let the $(N+1) \times (N+1)$ symmetric matrix $\boldsymbol{\Gamma}$ be:

$$\mathbf{\Gamma} = \begin{pmatrix} \hat{\gamma}_{0\,0} & \hat{\gamma}_{0\,1} & \cdots & \hat{\gamma}_{0\,N-1} & 1 \\ \hat{\gamma}_{1\,0} & \hat{\gamma}_{1\,1} & \cdots & \hat{\gamma}_{1\,N-1} & 1 \\ & & \cdots & & \\ \hat{\gamma}_{N-1\,0} & \hat{\gamma}_{N-1\,1} & \cdots & \hat{\gamma}_{N-1\,N-1} & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{pmatrix} \tag{8.10}$$

Then, the interpolated value $\hat{\lambda}(\mathbf{e^i})$ is computed as in Equation 8.11.

$$\hat{\lambda}(\mathbf{e^i}) = \mathbf{\Gamma_i} \cdot \mathbf{\Gamma}^{-1} \cdot \boldsymbol{\lambda} \tag{8.11}$$

The points $\mathbf{e^0}, ..., \mathbf{e^{i-1}}$ in which values of the random field λ are known used for the interpolation can be discrete or continuous. Nevertheless, the limit of the proposed method lies in the fact that the interpolated surface has to be continuous.

Exploitation for word-lengths optimization with *min+1 bit* algorithm To evaluate the number of points where kriging could be applied, the proposed method has been integrated in the optimization algorithm *min+1 bit*. The challenge is to determine whether the sequence of evaluated configurations allows predicting a large number of configurations using kriging. The optimization algorithm has been launched on the exhaustive input data set $\mathcal{I}$ to get the real accuracy metric values for each tested configuration. For each tested configuration, the word-lengths $\mathbf{w^i}$ of all the variables in the application are recorded as well as the real accuracy metric value. The different vectors $\mathbf{w^i}$ are corresponding to the different configurations $\mathbf{e^i}$. Consequently, for each vector of size N_v $\mathbf{w^i} = (w_0^i, w_1^i, ..., w_{N_v}^i)$, the real noise power value λ_i is measured. The points have been recorded in the order in which they have to be measured, for comparison with the results obtained by kriging. When the noise power value is obtained with simulations, the application is simulated with the considered word-lengths vector $\mathbf{w^i}$ on the exhaustive input data set $\mathcal{I}$ and the error e_x between the output of the application in floating-point and in fixed-point is measured and saved in the set E. These steps are summarized as the following function:

$$E \leftarrow \text{simulation}(\mathcal{I}, \mathbf{w^i}) \tag{8.12}$$

From the set E of error values, the chosen accuracy metric is computed. For instance, if the chosen accuracy metric is the noise power, the accuracy metric is computed as:

$$\lambda = E[e_x^2], x \in \mathcal{I} \tag{8.13}$$

In the rest of the chapter, the simulation of the word-lengths configuration $\mathbf{w^i}$ and the computation of the accuracy metric are summarized as:

$$\lambda = \text{evaluateAccuracy}(\mathcal{I}, \mathbf{w^i}) \tag{8.14}$$

The goal of the proposed method, is to replace the simulations for evaluating the accuracy metric values by kriging.

Proposed algorithm

The proposed method to estimate the accuracy metric λ is implemented in the `min + 1` bit optimization algorithm. For each tested word-lengths configuration $\mathbf{w}$, the proposed kriging-based technique has been applied to infer the accuracy metric value in point $\mathbf{w}$ from the surrounding accuracy metric values on the hypercube and for a given distance d.

Algorithm 8 Determination of $\mathbf{w}^{\min}$

1: **procedure** MINKWL($\lambda_m, \mathcal{I}, N_v, N_{max}, d$)
2: $W_{sim} = ()_{0,0}, \lambda_{sim} = ()_{0,1}, N_{sim} = 0$
3: **for** $i \in [1; N_v]$ **do** ▷ Min part
4: $\mathbf{w} \leftarrow (N_{max}, ..., N_{max})$
5: **repeat** ▷ Iterate on the variables
6: $j = 0, W_{tmp} = ()_{0,0}, \lambda_{tmp} = ()_{0,1}, N_n = 0$
7: **repeat** ▷ Iterate on the simulated points
8: $\mathbf{w}^j_{sim} \leftarrow W_{sim}(j, :)$
9: $d_{Cur} = ||\mathbf{w} - \mathbf{w}^j_{sim}||_1$
10: **if** $d_{Cur} \leq d$ **then**
11: $W_{tmp} \leftarrow W_{tmp} \cup \mathbf{w}^j_{sim}$
12: $\lambda_{tmp} \leftarrow \lambda_{tmp} \cup \lambda(j)$
13: $N_n \leftarrow N_n + 1$
14: **end if**
15: $j \leftarrow j + 1$
16: **until** $j < N_{sim}$
17: **if** $N_n > 1$ **then** ▷ Process Kriging
18: $\lambda = \text{kriging}(W_{tmp}, \lambda_{tmp}, \mathbf{w})$
19: **else** ▷ Simulation
20: $\lambda = \text{evaluateAccuracy}(\mathcal{I}, \mathbf{w})$
21: $W_{sim} \leftarrow W_{sim} \cup \mathbf{w}$
22: $\lambda_{sim} \leftarrow \lambda_{sim} \cup \lambda$
23: $N_{sim} \leftarrow N_{sim} + 1$
24: **end if**
25: $\mathbf{w}_i \leftarrow \mathbf{w}_i - 1$
26: **until** $\lambda \leq \lambda_m \vee \mathbf{w}_i \leq 1$
27: $\mathbf{w}^{min}_i \leftarrow \mathbf{w}_i + 1$
28: **end for**
29: **return** $\mathbf{w}^{\min}$
30: **end procedure**

The proposed methodology is described for Algorithm 8 for the determination of $\mathbf{w}^{\min}$ but is similar in Algorithm 9 for the determination of $\mathbf{w}^{\text{res}}$. Both algorithms takes as input the following parameters: the accuracy constraint λ_m, the number of variables to optimize N_v, the maximum word-lengths for the variables to optimize N_{max} and the distance to search for the neighbours of the interpolated point d. The impact of parameter d is studied in the experimental study. The matrix W_{sim} storing the already simulated word-lengths vectors, the vector storing the corresponding accuracy metric values as well as the number of simulated points are initialized to the null elements and to zero (line 2), since no point has been simulated for the moment. Then, for each tested word-length vector $\mathbf{w}$, the already simulated points are analyzed (lines 7-15) to determine if they can be used for kriging. For each point $\mathbf{w}^j_{sim}$ in matrix W_{sim}, its distance to the point $\mathbf{w}$ in which the accuracy metric value is searched is computed. The distance is obtained by computing the $\mathcal{L}_1$ norm between both vectors. If the obtained distance is lower or equal to the distance d, the point $\mathbf{w}^j_{sim}$ is kept as a neighbouring point to obtain the value of the accuracy metric with kriging. This value is stored in W_{tmp} as well as the corresponding accuracy metric value in λ_{tmp} (lines 11-12). If enough surrounding points have already been simulated, that is to say if N_n is higher than 1 (line 16), kriging is applied, else the point is simulated. When kriging is applied, from the already measured points, the matrix Γ in Equation 8.10 is computed and the accuracy metric value is estimated with Equation 8.11.

If the point is interpolated, it is not used for kriging other points. The higher d, the more points can be interpolated. The obtained optimization algorithm is described in Algorithm 8 for the determination of minimum word-lengths vector, and Algorithm 9 for the determination of the optimized word-lengths vector.

Algorithm 9 Determination of $\mathbf{w}^{res}$

procedure OPTIMKWL($\lambda_m, \mathcal{I}, N_v, \mathbf{w}^{\min}, W_{sim}, \lambda_{sim}, N_{sim}, d$)
  $\mathbf{w}^{res} \leftarrow \mathbf{w}^{\min}$
  repeat
    for $i \in [1; N_v]$ **do** ▷ Competition between variables
      $\mathbf{w}_i \leftarrow \mathbf{w}_i + 1$
      $j = 0, W_{tmp} = ()_{0,0}, \lambda_{tmp} = ()_{0,1}, N_n = 0$
      repeat ▷ Iterate on the simulated points
        $\mathbf{w}^j_{sim} \leftarrow W_{sim}(j, :)$
        $d_{Cur} = ||\mathbf{w} - \mathbf{w}^j_{sim}||_1$
        if $d_{Cur} \leq d$ **then**
          $W_{tmp} \leftarrow W_{tmp} \cup \mathbf{w}^j_{sim}$
          $\lambda_{tmp} \leftarrow \lambda_{tmp} \cup \lambda(j)$
          $N_n \leftarrow N_n + 1$
        end if
        $j \leftarrow j + 1$
      until $j < N_{sim}$
      if $N_n > 1$ **then** ▷ Process Kriging
        $\lambda_i = \text{kriging}(W_{tmp}, \lambda_{tmp}, \mathbf{w})$
      else ▷ Simulation
        $\lambda_i = \text{evaluateAccuracy}(\mathcal{I}, \mathbf{w})$
        $W_{sim} \leftarrow W_{sim} \cup \mathbf{w}$
        $\lambda_{sim} \leftarrow \lambda_{sim} \cup \lambda_i$
        $N_{sim} \leftarrow N_{sim} + 1$
      end if
      $\mathbf{w} \leftarrow \mathbf{w}^{res}$
    end for
    $j_c \leftarrow \underset{i}{\text{argmin}}\{\lambda_i\}$
    $\mathbf{w}^{res}_{j_c} \leftarrow \mathbf{w}^{res}_{j_c} + 1$
    $\lambda \leftarrow \lambda_{j_c}$
  until $\lambda \geq \lambda_m$
  return $\mathbf{w}^{res}$
end procedure

8.3 Experimental study

The experimental study aims at showing that:

1. The proposed method can replace simulation for an important number of samples.

2. The quality of the obtained estimation depends on the number of samples taken for inference, controlled by the parameter d.

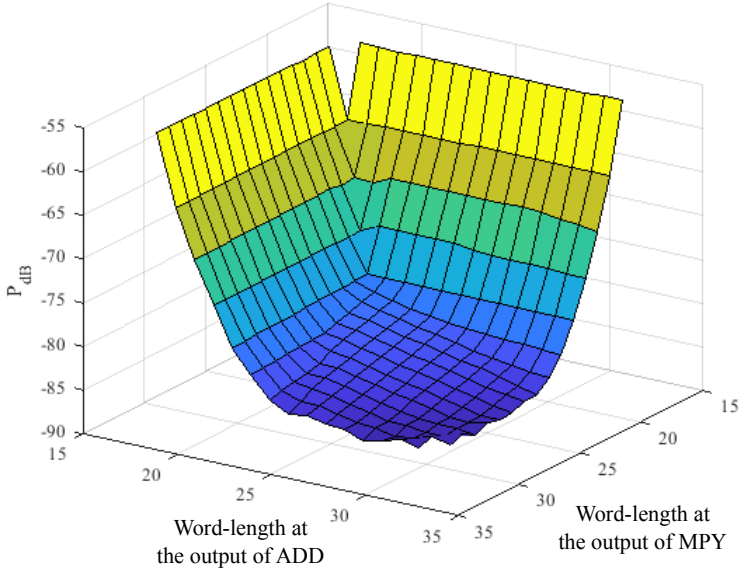


Figure 8.1 – *Evolution of the accuracy metric (noise power in dB) depending on the word-lengths of a FIR filter.*

3. The proposed method can be applied to the estimation of the noise power or to the estimation of an application QoS metric.

The kriging methodology has been implemented with the equations described in [FPTV92]. The proposed method has been applied on the fixed-point refinement algorithm *min+1 bit*. During this optimization process, numerous word-lengths configurations are tested and their impact on the accuracy metric is measured. For instance, for a FIR filter with two variables converted into fixed-point coding, the word-length at the output of the adder and the word-length at the output of the multiplier, the different measurements of the accuracy metric, in this case the noise power, lead to the creation of the surface presented in Figure 8.1. The goal of the proposed method is to estimate with a sufficient quality a non-negligible number of points of the surface without simulations.

In Table 8.1, the obtained results have been reported for the fixed-point refinement of several benchmarks. The distance d, which is the $\mathcal{L}_1$-norm between the point to interpolate and the already simulated points is indicated and varies between 2 and 5. For each considered distance, the percentage of points that can be interpolated instead of being simulated $p(\%)$ is indicated, as well as the average number of already simulated points i that were used for each interpolation. Finally, quality metrics are provided. Let ϵ be the difference between the interpolated and the real value. This difference is expressed as an equivalent number of bits when the accuracy metric is the noise power. In this case, the equivalent number of bits n^i is computed from the noise power value $\hat{P}(\mathbf{w^i})$ in configuration $\mathbf{w^i}$ as:

$$\hat{P}(\mathbf{w^i}) = \frac{2^{-n^i}}{12} \tag{8.15}$$

which can be developed to compute n^i as:

$$n^i = -\log_2(12 \cdot \hat{P}(\mathbf{w^i})) \tag{8.16}$$

When the accuracy metric is the noise power, for each interpolated point, the equivalent number of bits is computed as well as the equivalent number of bits for the real noise power value. The difference between the number of interpolated and accurate bits is computed and its maximum, average values and standard deviation over all the interpolations are indicated, max_ϵ, μ_ϵ and σ_ϵ respectively.

When another accuracy metric is considered, the difference between the interpolated and the real value of the accuracy metric is expressed as a relative difference. If $\hat{\lambda}(\mathbf{w^i})$ is the interpolated value of the accuracy metric in configuration $\mathbf{w^i}$ and $\lambda(\mathbf{w^i})$ is the accurate value of the accuracy metric, the relative difference is computed as:

$$\epsilon = \frac{|\hat{\lambda}(\mathbf{w^i}) - \lambda(\mathbf{w^i})|}{\lambda(\mathbf{w^i})} \tag{8.17}$$

The maximum max_ϵ, average μ_ϵ, and standard deviation σ_ϵ of ϵ are indicated.

Among the considered benchmarks, three benchmarks belong to classical signal processing kernels, a 64-th order FIR filter ($N_v = 2$), an 8-th order IIR filter ($N_v = 5$) and a Fast Fourier Transform (FFT) applied on 64 points [Ca65] ($N_v = 10$). For these benchmarks, the chosen accuracy metric is the noise power at the output of the filter. The proposed method allows faithfully interpolating a large number of the noise power values from close neighbours. Indeed, when using a distance contraint of $d = 3$ for the FIR filter, 52.78% of the points can be interpolated while inducing a low error of interpolation, on average 0.43 bit. For the IIR filter, for $d = 2$, 47.52% of the points can be interpolated inducing a similar error of interpolation. With a measured interpolation time of 10^{-6}s compared to a simulation time of 2.4s for the evaluation of a noise power by simulations, the total time for fixed-point refinement is on average halved with our method. It is to be noted that the low number of variables in these two benchmarks leads to a small fraction of the configuration space that can be infered. According to the FFT, the number of variables is larger than for the two filters. As a matter of consequence, a significantly larger number of points can be infered from $d = 2$, since 78.14% of the points can be infered without using simulations. The error of interpolation also stays really low since it is on average lower than 1 bit for $d \in [\![2; 5]\!]$. In the case of the FFT, if 80% of the noise power evaluations can be done without simulations, the time for fixed-point refinement is divided by 5.

The following considered benchmark is the 2-D motion compensation module of a High Efficiency Video Coding (HEVC) codec [Sa12]. This module has been used as a benchmark in Chapter 7 and processes blocks of 8×8 pixels to interpolate the block in the case of non-integer motion vector. For this module, 23 variables are considered in the word-length optimization process. The considered accuracy metric is the noise power. As for the FFT, since the number of variables in the optimization process is important, a large number of noise power values can be infered, since for $d = 2$, 87.35% of the points can be infered. Nevertheless, from $d = 4$, expanding the search space for interpolation is not useful since the percentage of infered points is only evolving by 0.35%. The average error of estimation is really low since lower or equal to 0.52 bit and the maximum difference is lower than 2.72 bit. In this case, the inference of the noise power values is really interesting since with a constraint on the noise power of -50dB, 2473 evaluations of the noise power are required. Each evaluation of the motion compensation module by simulation takes 1.37s. Consequently, if 90% of the evaluations can be replaced by the FAKEER method, the time for fixed-point refinement is divided by 10.

The last considered benchmark is a deep learning benchmark with $N_v = 10$ variables. This benchmark is an image classification benchmark, the SqueezeNet deep convolutional

neural network [IHM+16]. Contrary to the other tested benchmarks, the configurations $\mathbf{e^i}$ are no more the word-lengths of the data but the noise power allocated to the different kernels of the application in a noise-budgeting problem. The accuracy metric for this benchmark is the QoS metric of the application, that is to say the percentage of good classification on the whole input data set composed of 300 images. This benchmark leads to very similar results than the FFT in terms of percentage of points that can be interpolated without simulations since they have the same number of variables for fixed-point refinement. The difference between the interpolated and the real value is expressed as a relative difference as presented in Equation 8.17. The maximum relative difference ranges between 15.72% and 33.58% but is on average lower than 12.16%. For a distance $d = 3$, almost 90% of the configurations can be estimated with FAKEER instead of simulations, while inducing an average relative error of 6.51%. Finally, when converting the SqueezeNet benchmark with simulation-based fixed-point refinement, 290 configurations are tested for a refinement time of 30 hours. If the FAKEER method is implemented with $d = 3$, 89.31% of the configurations can be estimated with kriging. In this case, the time for fixed-point refinement is divided by a factor 10.

	$\boldsymbol{\lambda}$	N_v	**d**	**p**(%)	**i**	$\mathbf{max_\epsilon}$	$\boldsymbol{\mu_\epsilon}$	$\boldsymbol{\sigma_\epsilon}$
FIR	Noise Power	2	2	33.33	3.78	0.98	0.28	0.35
			3	52.78	5.44	1.66	0.43	0.53
			4	58.33	7.00	2.29	0.46	0.68
			5	66.67	8.61	2.42	0.51	0.67
IIR	Noise Power	5	2	47.52	2.72	1.29	0.44	0.31
			3	64.54	2.09	2.58	0.72	0.58
			4	70.92	2.00	3.24	1.02	0.79
			5	77.30	2.00	3.93	1.24	1.04
FFT	Noise Power	10	2	78.14	3.48	0.82	0.18	0.16
			3	89.07	2.01	1.21	0.34	0.26
			4	91.90	2.04	2.07	0.54	0.38
			5	95.55	2.05	2.88	0.68	0.55
HEVC	Noise Power	23	2	87.35	3.60	1.86	0.07	0.15
			3	93.33	2.38	1.86	0.15	0.20
			4	95.63	2.11	2.24	0.30	0.27
			5	95.96	2.01	2.72	0.52	0.36
SqueezeNet	Classification rate	10	2	78.28	3.33	15.72%	3.50%	3.28%
			3	89.31	2.18	25.75%	6.51%	5.68%
			4	91.38	2.12	31.57%	9.11%	9.09%
			5	93.10	2.09	33.58%	12.16%	10.59%

Table 8.1 – *Experimental Results for* FAKEER *Method.*

8.4 Conclusion

The method presented in this Chapter is the FAKEER method, that has not been published yet. The FAKEER method allows estimating the accuracy metric at the output of an application depending on the word-lengths configuration of the internal variables in the application. The estimation is done with a geostatistical method, kriging. The number of estimated configurations during fixed-point refinement depends on the distance d. The distance d corresponds to the distance between the estimated configuration and its neighbours taken for estimation. We have verified that with a tight proximity in the neighbouring points, kriging enables halving the number of accuracy metric evaluation with simulations, while keeping an estimation error lower than 0.5 bit for small signal processing benchmarks composed of a few variables. When the number of variables in the considered benchmark increases, the search space for the configurations used for kriging is larger and the number of points that can be estimated increases up to 90% on average. However, when comparing the savings brought by the FAKEER method compared to the savings obtained with the adaptive number of samples taken for noise power estimation presented in Chapter 7, the results are less significant. The adaptive number of samples taken for noise power estimation allows reducing the time for fixed-point refinement up to 3 orders of magnitude, while the FAKEER method reduces the time for fixed-point refinement up to 1 order of magnitude. The major advantage of the FAKEER method is that it is not dependent on a particular accuracy metric contrary to the method proposed in Chapter 7. This advantage is particularly interesting when the chosen accuracy metric is hard to evaluate.

CHAPTER 9

Conclusion

"I used the wrong method with the wrong technique"

Depeche Mode

9.1 Summary

Approximate Computing (AC) is an energy-aware computing technique and has been considered in this thesis. AC relies on the exploitation of the tolerance to imprecision of an application. Energy-aware computing techniques are an active field of research to face the end of Moore's law while being able to answer the growing demand in computing capacity. Numerous applications in image, signal or video processing are particularly tolerant to imprecisions. To benefit from this error-resiliency, numerous approximation techniques have been proposed at different abstraction levels, from the circuit to system level. An overview of these techniques has been presented in Chapter 2. These techniques have generally been characterized in terms of energy consumption, latency or area. Nevertheless, it is of major concern to be able to characterize the approximations in terms of error induced and their impact on the output application quality. In Chapter 3, the different error and accuracy metrics as well as the methods proposed in the literature to evaluate them have been presented.

In this thesis, methods and tools have been proposed to quickly evaluate the impact of different AC techniques on the quality of service (QoS) of an application. AC techniques have been considered at two different levels of approximation, the hardware level with the study of inexact arithmetic operators and the data level with the study of fixed-point arithmetic. Both approximation techniques allow studying two different categories of errors. Inexact arithmetic operators generate errors of high amplitude with a low error rate, they are called "fail rare" errors. Fixed-point arithmetic always generates errors with a low amplitude, they are called "fail small" errors. When evaluating the error induced by an AC technique on an application, two stages are required. First, the errors induced by the approximations are characterized and modeled with different types of error metrics, and they are then linked with the application QoS. The majority of the proposed techniques for

modeling the errors induced by the approximations are analytical techniques. Analytical techniques are specific to the considered approximation and hard to automate. To link the induced errors with the application QoS, simulations are commonly used. Exhaustive or Monte-Carlo simulations have been proposed, but exhaustive simulations are too long to be used for sophisticated applications, while the existing techniques based on Monte-Carlo simulations do not provide any confidence information on the obtained metrics. The study of the impact of approximations on the application quality metric is critical to use AC techniques in real applications.

In this document, five main contributions have been presented.

- An approximation technique at the computation level is presented in Chapter 4.
- Cassis, a characterization framework for the errors induced by inexact operators is described in Chapter 5.
- A fast functional simulator of inexact operators, the Fast and Fuzzy (FnF) simulator is proposed in Chapter 6.
- A framework for fast finite precision refinement is presented in Chapter 7.
- Fakeer, an interpolation-based technique to estimate a quality or accuracy metric is detailed in Chapter 8.

In Chapter 4, another level of AC techniques has been explored. An approximation at the computation level has been proposed for a stereovision algorithm. This case study has demonstrated the strong link between the obtained quality at the output of an application implementing approximations and the chosen accuracy metric. This work has motivated the rest of our contributions to find generic methods and tools to measure the quality at the output of an application.

The study of inexact arithmetic operators has been presented in Chapters 5 and 6. Chapter 5 presents a statistical characterization method for the error induced by inexact arithmetic operators. The proposed framework is based on adaptive simulations and characterizes the error in terms of Mean Error Distance (mean ED), Error Rate (ER) and Maximum Error Distance (maximum ED) according to user-defined confidence requirements. Contrary to analytical techniques, this method is operator agnostic. This framework has been presented in two contributions [BCDM18, BCDM19]. In Chapter 6, the errors induced by inexact arithmetic operators are linked with the application QoS through a fast simulation process. The FnF simulator has been proposed for the Approximation Design Space Exploration (ADSE) process to select the best approximation for the considered application. The proposed simulator can replace the simulation of the operator at the logic level and is up to 63 times faster than Bit-Accurate Logic-Level (BALL) simulation of inexact arithmetic operators. The FnF simulator has been presented in two contributions [BDM18c, BDM18a]. The framework for the characterization of the errors induced by inexact arithmetic operators can be used to build the FnF simulator.

The study on fixed-point arithmetic has been presented in Chapters 7 and 8. Chapter 7 proposes a model for a particular intermediate accuracy metric massively used for fixed-point arithmetic, the noise power. This error modeling allows estimating the value of the noise power at the output of an application with a reduced number of simulations. The noise power model has been implemented in a fixed-point refinement algorithm to determine the optimized word-lengths of the internal variables in an application under quality constraint. The proposed refinement algorithm was faster than its original version by two

to three orders of magnitude, saving time during this tedious process. These contributions have been presented in [BDM19a, BDM19b] and exploited in a French patent [BMD19]. The last contribution of this thesis has not been published yet. The FAKEER method has been presented in Chapter 8. It is used during fixed-point refinement of an application but contrary to the contribution presented in Chapter 7, it is not specific to an accuracy metric. The FAKEER method aims at estimating the accuracy metric at the output of an application during fixed-point refinement, without simulations. This method is based upon research on geostatistical methods as kriging. If the application to convert to fixed-point arithmetic has numerous variables, this method is particularly efficient and allows estimating 90% of the points without simulations for a low error of estimation.

The work constituting this thesis comes with the following conclusions:

- The errors induced by AC techniques such as inexact arithmetic operators are unsmooth which jeopardize the implementation of such approximation techniques in industrial applications. This is the reason why we have decided to consider other approximation techniques as finite precision arithmetic during the thesis.
- To my mind, it is important to link the error metrics with statistics that can be computed from a few samples and with a known confidence. Instead of using simulations to compute a single value of the considered error metric, intervals appear as being more relevant.
- To use AC in industry, a lot of work still has to be done to link the quality loss with the implemented approximation. For instance, according to fixed-point arithmetic, which is an old approximation technique, no efficient automatic method is available to quickly study the approximation error and process the fixed-point refinement. Analytical methods have mainly been proposed and the methods based on the perturbation theory are well defined. However, deriving equations from an application description given as a C code is still a challenge. Simulation-based methods are generic but were not scaling with large applications up to the work proposed in this thesis. The results presented in Chapters 7 and 8 could solve a part of this problem. There is still a long way to go for the other approximation techniques.

9.2 Future works

The work presented in this thesis opens many opportunities for future reasearch in the error modeling field for AC. To my mind, interesting research has still to be done on finite precision arithmetic.

Framework for refinement of finite precision systems First, the work proposed in Chapter 7 is being integrated to a source to source compiler, the Generic Compiler Suite (GeCoS) compiler developed in INRIA. The goal of this integration would be to provide the embedded systems engineers with an automated tool to convert their applications from double floating-point precision to finite precision, whether it be fixed-point or custom floating-point arithmetic. The input of this automated tool would be floating-point C or C++ code describing the application to convert to finite precision, and the output would be a C or C++ code with optimized data-types in finite precision. The output of the tool could be used as an input of High-Level Synthesis (HLS) tools. The optimization of the word-lengths of the data in the application would be done with the evaluation tools proposed in Chapters 7 and 8 so as to reduce the time for the design space exploration.

Inferential statistics for custom floating-point Custom floating-point is an interesting finite precision arithmetic has shown by Barrois et al [BS17]. For the refinement of the word-lengths in the case of custom floating-point arithmetic, the size of the mantissa and the exponent have to be explored. To use inferential statistics on custom floating-point arithmetic, we expect that the framework proposed in Chapter 7 would be less efficient because of the error distribution in floating-point arithmetic. For fixed-point arithmetic, the error distribution is uniform while for floating-point arithmetic it is a uniform distribution to which is applied a scale factor. Preliminary results have been obtained using the GeCoS compiler. As expected, the proposed framework is less efficient since the noise induced by floating-point arithmetic is not distributed as for fixed-point arithmetic.

Sustainability of the `Fakeer` method Further works on the FAKEER method include testing the proposed method on different quality metric as for instance metrics to evaluate the quality at the output of the stereo matching algorithm. The `Fakeer` method also has to be tested on applications of various sizes to check its robustness. Finally, standard deviation maps could be used to predict and control the interpolation error depending on a user-defined confidence information. In this case, confidence intervals on the estimated accuracy or quality metric can be computed.

Confidence intervals for various metrics The work based on adaptive simulations to derive confidence intervals on error metrics could be extended to other classical metrics that are expressed as a linear combination of statistical parameters as the Structural Similarity Index Measure (SSIM) for instance. Another interesting error metric that is being considered is the classification rate that could be dealt as the estimation of the ER. The goal would be to provide a library of common metrics to the user.

APPENDIX A

French Summary

A.1 Introduction

La compétition pour concevoir des systèmes électroniques plus rapides, moins chers et plus économes en énergie a des raisons économiques mais cherche également à répondre à la nécessité d'économiser les ressources énergétiques disponibles. En effet, selon la "Semiconductor Industry Association et la Semiconductor Research Corporation", l'énergie totale requise par les systèmes informatiques dépassera la production énergétique mondiale estimée d'ici 2040, si aucune amélioration significative n'est obtenue en termes de réduction de la consommation énergétique [AC15].

Une explosion en capacité de calcul

De 1965 à 2012, la capacité de calcul offerte par les systèmes informatiques, ainsi que leur précision, ont augmenté de façon exponentielle. Selon la loi de Moore [Moo06], énoncée moins d'une décennie après l'invention du circuit intégré à base de Silicium en 1958 par Jack Kilby, le nombre de transistors sur une puce électronique devait doubler tous les 2 ans. En réalité, le nombre de transistors a doublé tous les 18 mois. Cette cadence a permis d'augmenter la vitesse des calculs tout en réduisant le prix des puces électroniques. Gordon Moore a en effet fait une prédiction tout à fait remarquable puisque le nombre de transistors sur un circuit intégré a suivi de près cette tendance entre 1970 et 2016. La réduction des coûts de fabrication et d'achat de puces électroniques a été rendue possible grâce à de nouveaux procédés chimiques permettant de fabriquer les circuits intégrés tout en optimisant les coûts. Au milieu des années 1960, l'utilisation de la lithographie optique au lieu de la peinture à la main a rendu l'emballage des transistors plus coûteux que la fabrication du transistor elle-même. Analysée par Hutcheson [Hut05], la version économique de la loi de Moore stipule que le coût par transistor est divisé par deux tous les 18 mois. De 30 transistors sur les premiers circuits intégrés à base de silicium à 10 milliards de transistors aujourd'hui, la production mondiale de puces électroniques a explosé, grâce à des facteurs techniques tels que la réduction de la taille des caractéristiques des transistors, un rendement accru et une densité de transistors sur puce accrus.

La facilité d'accès à des puces électroniques de plus en plus puissantes a fait également exploser la demande. Néanmoins, depuis 2012, la loi de Moore a atteint ses limites. Les

tendances de la technologie CMOS empêchent de respecter les rendements et les performances prévus. En effet, comme Markov [Mar14] l'a démontré, l'efficacité du calcul est intrinsèquement limitée dans sa nature fondamentale, matérielle, périphérique, circuit et système/logiciel. Par exemple, l'interconnexion sur puce limite considérablement les performances de cette dernière. Les transistors sont eux-mêmes limités par la largeur de la grille diélectrique, qui a atteint la taille de quelques atomes. L'augmentation de la vitesse des calculs a également induit une augmentation de l'énergie nécessaire à ces calculs.

Le matériel du futur propice aux erreurs

Un paradoxe est donc posé : les concepteurs de systèmes d'intégration à très grande échelle doivent réduire le coûts de fabrication des puces électroniques pour répondre à la demande croissante des utilisateurs ce qui implique de modifier leurs marges de conception mais également de produire toujours plus de technologies sujettes aux erreurs. En effet, selon la loi de Moore, si la densité de transistors sur une puce augmente, le nombre de défauts dans une puce augmente elle avec la densité des transistors. Les technologies du futur sont donc toujours plus propices aux erreurs.

En raison de la vulnérabilité aux erreurs des puces actuellement fabriquées, si l'on souhaite assurer une précision stricte sur un résultat, une redondance dans les modules de calcul est nécessaire. Ce processus augmente d'autant plus l'énergie nécessaire pour alimenter la puce. Le futur matériel n'étant pas fiable, le processus de mise au rebut d'une puce en raison d'une imperfection semble être une autre partie du paradoxe actuel.

Une quantité massive de données à traiter

Le volume de données à traiter a également explosé au cours des dernières années : "En 2010, pendant deux jours, le même volume d'information est produit qu'il l'a été en deux milliards d'années jusqu'en 2003" (Eric Schmidt [Sie10]). Les centres de données devraient traiter 175 zettaoctets de données d'ici 2025 [Ins19]. Ce volume croissant de données à traiter s'accompagne d'une croissance exponentielle de la demande de stockage et de capacité de calcul.

De nouvelles méthodes de calcul sont nécessaires pour traiter ce flux massif d'informations à traiter. Par exemple, le plus grand radiotélescope du monde, le Square Kilometer Array (SKA) [DHSL09], dont la construction est prévue pour 2020, sera composé de centaines de milliers d'antennes réceptrices. Dans la première phase du projet SKA, 160 téraoctets de données brutes par seconde seront générés et devront être analysés. Un autre exemple de ce volume exponentiellement croissant de données produites est l'Internet des Objets qui relie ensemble des milliards de dispositifs informatiques. Il est indispensable de travailler sur de nouvelles méthodes de calcul économes en énergie, à tous les niveaux des systèmes informatiques, que ce soit au niveau des capteurs ou au niveau des processeurs.

Ces nouvelles méthodes de calcul sont également recherchées afin de répondre aux contraintes de temps réel et d'énergie lors de la conception des systèmes embarqués, et d'économiser des ressources lors de la phase d'implémentation. Dans ce contexte, les calculs approximés sont l'une des principales approches proposées et sont à l'étude dans cette thèse. Les calculs approximés utilisent la précision numérique d'une application comme nouveau paramètre sur lequel jouer pour concevoir des systèmes plus efficaces en termes de surface, d'énergie ou de rapidité de traitements. La résilience aux erreurs de nombreuses applications est exploitée afin d'économiser de l'énergie ou d'accélérer le traitement. Cette résilience aux erreurs intrinsèques est particulièrement présente dans les

algorithmes de traitement du signal, de l'image, de la vidéo, de l'intelligence artificielle ou du data mining.

A.2 Méthodes de calcul efficaces en énergie

Plusieurs types de méthodes de calcul efficaces en énergie ont été proposés depuis les années 1960. La consommation énergétique étant l'un des principaux facteurs limitant pour augmenter la performance des applications, ces nouvelles méthodes de calcul ont été proposées pour optimiser la consommation énergétique.

A.2.1 Le calcul stochastique

Le calcul stochastique a été proposé dans les années 1960 [Gai69, AH13]. Le principe du calcul stochastique est de traiter les données comme des probabilités représentées par des flux binaires. La probabilité est calculée en comptant le nombre de 1 dans le flux binaire sans que la position des 1 n'aient d'impact sur la valeur obtenue. Un flux binaire est alors appelé un nombre stochastique. Par exemple, pour représenter la valeur $x = 0,75$, de nombreux flux binaires peuvent être utilisés si l'on fait varier la longueur des flux utilisés, puisque x représente la probabilité d'avoir un 1 dans le flux :

$$x = 0.75 = \begin{cases} (1,1,1,0) \\ (1,0,1,1) \\ (1,1,1,1,0,0,1,1) \end{cases}$$

Le fait de traiter les informations sous forme de flux binaires permet d'utiliser des unités de traitement moins coûteuses, moins complexes et consommant moins d'énergie. En effet, une multiplication peut ainsi être implémentée à l'aide d'une porte **ET**.

Par exemple, au lieu de calculer directement la multiplication de $x = \frac{4}{8}$ et $y = \frac{6}{8}$, ils peuvent être représentés comme suit:

$$x = \frac{4}{8} = \begin{cases} (0,1,1,0,1,0,1,0) \\ (0,1,0,1,1,1,0,0) \end{cases}$$

$$y = \frac{6}{8} = \begin{cases} (1,0,1,1,1,0,1,1) \\ (1,1,1,0,1,0,1,1) \end{cases}$$

La multiplication de x et y sera ensuite traitée avec une porte **ET**, ce qui donne comme résultat de l'opération sur les premières représentations de x et y le résultat $(0,0,1,0,1,0,1,0) = \frac{3}{8}$, tandis que pour les secondes représentations de x et y on obtient le résultat $(0,1,0,0,1,0,0,0) = \frac{2}{8}$.

Néanmoins, comme le montre l'exemple, en fonction du nombre stochastique utilisé pour représenter les valeurs (x, y), la précision en sortie de l'opération est modifiée. Pour générer un nombre stochastique représentant une donnée x, des générateurs de nombres pseudo-aléatoires sont utilisés. La difficulté, en générant des nombres stochastiques pour représenter les probabilités, est que chaque bit dans le flux binaire possède le même poids, donc plusieurs nombres stochastiques peuvent représenter la même probabilité. Pour calculer avec des nombres stochastiques non biaisés, ils doivent être non corrélés et suffisamment aléatoires. La corrélation entre les nombres stochastiques peut générer des erreurs pendant le calcul. Pour obtenir des statistiques sur les erreurs commises lors de calcul stochastique, on utilise les techniques d'inférence statistique, d'estimation et de détection. La précision d'un nombre stochastique est représentée par le nombre de bits utilisés pour

coder les données. L'un des principaux problèmes du calcul stochastique est que la longueur des flux binaires utilisés pour traiter les données augmente de façon exponentielle avec la précision requise. Par conséquent, le calcul stochastique est généralement lent pour obtenir une précision relativement faible. Malgré les inconvénients des traitements stochastiques, la production actuelle de circuits matériels propices aux erreurs et le non-respect du rendement ont poussé les chercheurs à reconsidérer ce types de calcul.

A.2.2 Le calcul quantique

Le calcul quantique est présenté ici [Gru99] comme ouverture sur les différents types de calculs efficaces en termes de consommation énergétique. Le calcul quantique vise à reproduire la capacité des particules subatomiques, par exemple les hadrons, qui peuvent exister dans plus d'un état à la fois. Les ordinateurs quantiques peuvent alors être plus efficaces que les ordinateurs classiques pour résoudre des problèmes complexes, comme par exemple lors de la factorisation d'entiers ou la résolution de logarithmes discrets [Sho99]. En effet, contrairement aux traitements informatiques classiques où l'information est stockée sur des bits qui ne peuvent exister que dans deux états distincts, 0 ou 1, en informatique quantique, l'information est stockée sur des qubits, la généralisation quantique d'un bit. Deux lois fondamentales régissent l'informatique quantique.

1. La superposition : les qubits peuvent exister sur un état $|0>$, un état $|1>$ ou bien sur une combinaison linéaire des deux exprimée comme $|\phi>= a \cdot |0> + b \cdot |1>$ où (a, b) sont des nombres complexes.

2. L'enchevêtrement : il existe une corrélation entre les comportements aléatoires de deux qubits pris individuellement.

Les qubits sont une superposition des différentes valeurs et peuvent exister dans plus de 2 états à la fois. Un qubit peut être positionné n'importe où sur une sphère de rayon 1. La probabilité que le qubit soit dans l'état $|0>$ est égale à $|a|^2$, et dans l'état $|1>$ à $|b|^2$. Par conséquent, les calculs avec des qubits permet de stocker plus d'informations avec moins d'énergie. Cependant, la stabilité des qubits reste un problème, de même que les méthodes de refroidissement des ordinateurs quantiques. En 2016, IBM a sorti un ordinateur sur 5 qubits disponible en ligne et programmable en Python pour les chercheurs, et a sorti plus récemment un ordinateur quantique sur 20 qubits.

Néanmoins, l'informatique quantique nécessite de nouveaux systèmes logiciels pour être viable.

A.2.3 Le calcul probabiliste

Contrairement au calcul stochastique ou quantique, **le calcul probabiliste** présenté par Palem et al. [Pal03] est une technique pour estimer les économies d'énergie obtenues lorsqu'on fait un compromis sur la précision des calculs. Le calcul probabiliste relie l'énergie consommée par un calcul à sa probabilité d'être exact. Il est logique de relier les calculs à la notion de probabilité puisque le comportement des erreurs est mieux décrit par un comportement probabiliste. Palem considère que les économies d'énergie peuvent être calculées avec des lois thermodynamiques. Un bit d'information égal à 0 ou 1 est alors appelé une valeur et est considéré comme un micro-état thermodynamique qui peut être mesuré. L'équation de Boltzmann nous donne l'énergie qui peut être économisée lors d'un calcul ayant une probabilité p d'être exacte, comme étant $k \cdot T\ln(\frac{1}{p})$ Joules. Le calcul probabiliste a été fortement influencé par la théorie des automates [VN56] de

Von Neumann dans laquelle il affirme que “l’erreur devrait être traitée par des méthodes thermodynamiques”.

A.2.4 Le calcul approximé

Le calcul approximé est la technique de calcul efficace en termes de consommation énergétique considérée dans cette thèse. Le calcul approximé s’inspire de la conception de l’erreur de Von Neumann, dans laquelle la tolérance à l’imprécision d’une application est exploitée. En effet, en 1956, John Von Neumann [VN56] a étudié comment limiter une erreur induite lors de la construction d’un automate utilisant des composants propices aux erreurs. Au lieu de traiter les erreurs avec la volonté de les éliminer complètement, il a montré que dans certaines conditions, la précision d’une opération réalisée par un automate peut être imparfaite, mais contrôlée et encadrée. De plus, tant que cette erreur était limitée, l’automate était capable de donner des résultats acceptables malgré la présence de valeurs erronées. Selon John Von Neumann, “l’erreur est considérée comme (...) un élément essentiel du processus à l’étude”. L’erreur est alors traitée comme une information nécessaire à prendre en compte tout au long du processus de calcul. Les processeurs implémentant des techniques d’approximations essaient d’imiter les calculs faits par le cerveau. Par exemple, lorsqu’on demande à quelqu’un d’évaluer rapidement un pourcentage en calcul mental, il réfléchit avec des approximations pour avoir une idée précise de la réponse. Le cerveau possède environ 10^{10} neurones qui possèdent eux-mêmes environ 10^4 connexions. Le cerveau a besoin de 25 Watts de puissance continue pour pouvoir calculer 25 péta-opérations par seconde. Sûrement le meilleur processeur jamais créé ! Au cours des deux dernières décennies, la recherche sur le paradigme des calculs approximés a été très active. De nombreuses techniques d’approximations ont été proposées à différents niveaux d’abstraction allant du circuit au système.

A.3 Le calcul approximé

A.3.1 La propriété de résilience aux erreurs

Appliquer les techniques de calcul approximé à une application nécessite une bonne connaissance de son comportement en cas d’erreur. La propriété de résilience aux erreurs d’une application est le point clé de l’application de techniques de calcul approximé : utilisé dans des domaines tels que le traitement vidéo ou d’images, le data mining ou les applications de reconnaissance, le calcul approximé exploite le fait que la précision offerte par une application est généralement supérieure à celle requise. De nombreuses applications peuvent produire des résultats acceptables tout en différant du résultat exact attendu, comme expliqué dans [KIK+98] pour les algorithmes de codage vidéo. La propriété de résilience aux erreurs, pour une application, est la capacité de l’application à produire des résultats acceptables malgré le fait que certains calculs sont ignorés ou approximés et donc inexacts. Cette propriété peut être induite par plusieurs facteurs :

- L’**utilisateur final** : le besoin de l’utilisateur final ou sa perception peut limiter le besoin de précision sur la sortie d’une application. Par exemple, si l’utilisateur final de la sortie de l’application est un réseau neuronal, il peut compenser les erreurs induites par l’utilisation de l’approximation. Un autre exemple est dans le traitement vidéo où l’utilisateur final est l’utilisateur humain et où la perception visuelle humaine limite considérablement la précision nécessaire. Enfin, pour les applications comme le data mining, il n’existe pas de résultat parfait.

- L'**entrée**: les données en entrée d'une application peuvent être redondantes ou bruitées, induisant de façon inhérente du bruit dans les calculs. Par exemple, les données expérimentales peuvent généralement être traitées à l'aide d'approximations.
- Les **calculs**: les calculs au sein d'une application peuvent être statistiques, probabilistes ou récursifs. De plus, les calculs itératifs peuvent compenser les erreurs commises par l'utilisation de modules de calculs approximés.

Il est à noter cependant que certaines applications ne tolèrent pas d'erreurs. Par exemple, dans les applications basées sur le contrôle, une petite erreur dans les calculs peut conduire à une mauvaise décision et, en fin de compte, à une erreur non acceptable à la sortie de l'application. En outre, dans les systèmes critiques, l'application doit être prévisible, c'est-à-dire que, pour une entrée donnée, le résultat en sortie doit être connu à l'avance.

En ce qui concerne les applications résilientes aux erreurs, l'erreur entre la sortie de l'application de référence et la sortie de l'application approximée peut être utilisée pour trouver un compromis en termes de performance, c'est-à-dire temps de calcul ou consommation d'énergie, et de coût de mise en œuvre, c'est-à-dire mémoire requise ou complexité. Le calcul approximé offre un nouveau degré de liberté pour concevoir une application comme illustré dans la Figure A.1.

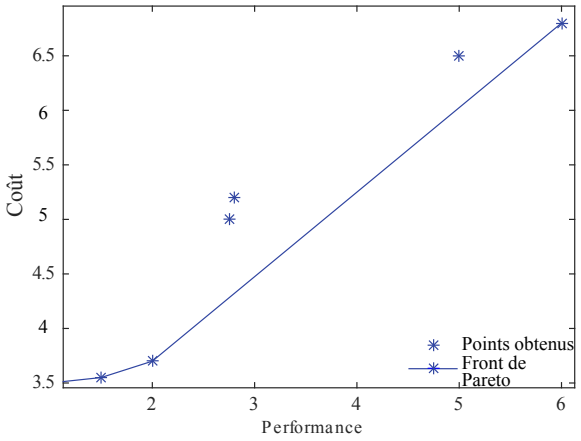
Néanmoins, la propriété de résilience aux erreurs d'une application n'est pas suffisante pour appliquer les techniques de calcul approximé. Les programmeurs doivent clairement identifier les parties de l'application qui tolèrent les approximations ainsi que les parties où une précision stricte est requise. Par exemple, une application avec des opérations non linéaires peut ne pas tolérer d'approximations à proximité de ces opérations. Les approximations ayant un impact sur le flot de contrôle ou les accès en mémoire peuvent conduire à des comportements désastreux comme Yetim et al. [YMM13] le présentent avec une approximation conduisant à une erreur de segmentation mémoire. Une analyse de sensibilité peut être conduite pour annoter l'algorithme et indiquer les parties les moins sensibles, comme présenté dans [CCRR13]. Chippa et al. ont proposé un framework de caractérisation de la résilience des applications pour identifier les parties tolérantes aux erreurs, et les caractériser en fonction de diverses techniques d'approximations mises en oeuvre.

De plus, pour choisir une stratégie d'approximation, une exploration de l'ensemble des possibilités d'approximations est généralement effectuée afin de trouver la meilleure configuration par rapport aux contraintes de la plateforme sur laquelle l'algorithme est intégré. Une fois qu'une stratégie d'approximation a été choisie, l'impact de l'approximation sur la qualité en sortie de l'application doit être quantifié. La mesure de la qualité en sortie de l'application dépend de l'application. Par exemple, pour une application de traitement du signal, la métrique de qualité de l'application peut être le rapport signal sur bruit. Pour une application de traitement d'image, la mesure de qualité de l'application peut être l'indice de similitude structurelle [WBSS04].

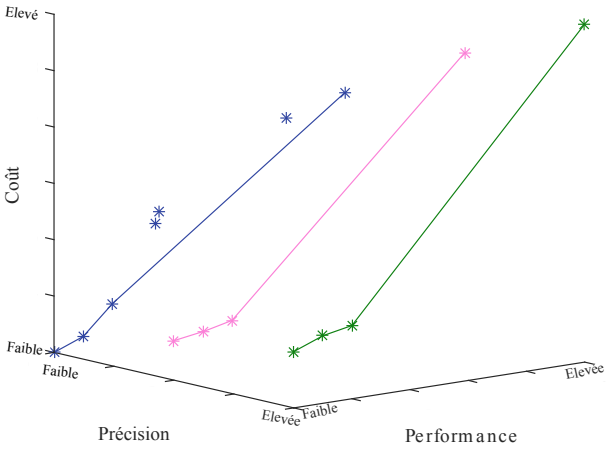
A.3.2 Les différents niveaux de techniques d'approximations

Le calcul approximé peut être appliqué à différents niveaux, comme présenté sur la Figure A.2.

Un premier axe possible d'approximations est au niveau des **données**. Le nombre de données à traiter peut être réduit menant à des données à traiter non à jour ou bien à un plus petit volume de données à traiter. Les données peuvent également être stockées avec



(a) Conception d'une application sans calcul approximé.



(b) Conception d'une application avec calcul approximé.

Figure A.1 – *Un nouveau degré de liberté pour la conception d'application avec des calculs approximés.*

une précision réduire, permettant d'économiser des bits et donc de l'énergie. Les données peuvent aussi être compressées. Dans les programmes parallélisés, la synchronisation entre différents threads peut être relachée afin d'utiliser des données moins à jour.

Un autre axe possible d'approximations est au niveau des **calculs**. Dans ce cas, l'algorithme lui-même est modifié et ajusté de façon à trouver un compromis entre la précision et la performance. On peut par exemple ne pas exécuter certains calculs ou certains accès mémoires. L'approximation de fonctions peut également être utilisée, en remplaçant les fonctions mathématiques sophistiquées par des polynômes, ou en stockant certaines valeurs des fonctions dans des tables par exemple.

Enfin, des approximations peuvent être faites au niveau **matériel**. Dans ce cas, on recherche une implémentation matérielle d'un circuit moins énergivore, comme par exemple les nombreuses implémentations alternatives d'opérateurs arithmétiques exacts. Un circuit peut également être approximé en termes de paramètres de fonctionnement : la

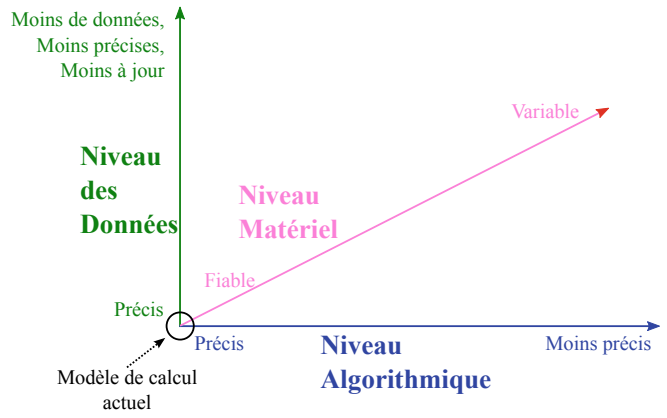


Figure A.2 – *Différents niveaux d'approximation.*

modification de la tension ou de la fréquence de fonctionnement d'un circuit a un impact sur la qualité du résultat et sur la consommation énergétique. Pour finir, un système de stockage moins fiable peut être utilisé pour stocker des données résilientes aux erreurs.

Au-delà des trois axes distincts de calcul approximé, Hedge et al. [HS01] ont proposé une technique pour identifier les parties d'une application pouvant être approximées à différents niveaux d'abstraction. Les applications de recherche et synthèse de données sont ciblées avec cette méthode s'appliquant aux différents niveaux d'abstraction, car elles possèdent plusieurs propriétés intéressantes pour l'utilisation de calcul approximé : les algorithmes statistiques et agrégatifs peuvent moyenner ou bien éliminer les erreurs, et les algorithmes itératifs ont la propriété de compenser les erreurs.

Du point de vue du programmeur, il reste un long chemin à parcourir entre la mise en œuvre d'une version approximée utilisable d'une application à partir d'un modèle de référence exact de celle-ci. C'est la raison pour laquelle des méthodologies entières ont été créées pour appliquer des techniques d'approximation à une application entière : de la subdivision de l'application en blocs, au calcul des économies d'énergie apportées par les approximations, de véritables recettes ont été proposées pour aider les programmeurs à utiliser efficacement ces techniques.

Identifier le niveau d'application des approximations est un véritable défi actuel. En effet, des approximations peuvent être appliquées du niveau du transistor jusqu'au processeur lui-même.

A.4 Étendue de la thèse et contributions

Pour implémenter des calculs approximés dans une application, la fonctionnalité de l'application doit être garantie et les erreurs induites par l'approximation doivent être encadrées et quantifiées. Des outils sont nécessaires pour explorer rapidement les différentes perspectives d'approximation et leur impact sur la qualité de sortie des applications. La difficulté d'étudier l'impact des approximations sur un algorithme vient du fait que les versions approximées des algorithmes sont généralement plus complexes en termes d'implémentation matérielle ou d'accès mémoire que leur implémentation d'origine. La simulation des algorithmes appproximés est plus compliquée et l'effort d'adaptation pour les utiliser nécessite généralement toute une équipe d'ingénieurs à la fois au niveau système et matériel. Le défi majeur lorsqu'on introduit des approximations dans une application est d'évaluer l'impact

de l'approximation sur la qualité de service à la sortie de l'application. Les approximations potentielles doivent être analysées et affinées pour choisir les meilleures approximations en fonction des différentes contraintes d'implémentation de l'application. L'espace d'exploration possible est grand et nécessite de pouvoir simuler rapidement l'application pour évaluer l'impact de l'approximation sur la qualité de service. Le choix de l'ensemble des configurations à simuler est également un point critique.

Dans cette thèse, l'objectif est de proposer de nouvelles techniques pour résoudre le problème de la caractérisation et de la propagation des erreurs dans les systèmes approximés. Les méthodes proposées pour caractériser les erreurs induites par les approximations sont présentées, ainsi que les méthodes permettant de relier les erreurs induites à la qualité de sortie de l'application. Les principales contributions de cette thèse sont :

1. Un framework générique basé sur des simulations permettant de caractériser de façon statistique l'erreur induite par les opérateurs inexacts. Cette méthode est basée sur les statistiques inférentielles et la théorie des valeurs extrêmes. Ces méthodes permettent de trouver un sous-ensemble à simuler pour avoir une estimation des statistiques recherchées en fonction d'une confiance demandée par l'utilisateur de la méthode. Cette contribution a été publiée dans [BCDM18] et [BCDM19].

2. Un simulateur rapide d'opérateurs arithmétiques inexacts a été développé. Ce simulateur permet de mesurer rapidement l'impact d'une approximation sur la qualité de service de l'application. Cette contribution a été publiée dans [BDM18c] et [BDM18a].

3. Une nouvelle méthode permettant d'évaluer la puissance du bruit de calcul lors d'une conversion d'une application en virgule fixe a été proposée. L'erreur induite par l'arithmétique virgule fixe est caractérisée de façon statistique ce qui permet de calculer une estimation de la puissance du bruit à l'aide d'un nombre réduit et adaptatif de simulations. Cette contribution a été publiée dans [BDM19a].

4. Une nouvelle méthode basée sur les statistiques inférentielles a été développée pour raffiner les largeurs des variables d'une application convertie en virgule fixe. Le nombre de simulations est adapté en fonction de la précision d'estimation de la puissance du bruit voulue. Cette contribution a été présentée sous forme de poster à la conférence DAC 2019 et a mené au dépôt d'un brevet Français FR1903747 [BMD19].

5. Une méthode pour inférer une métrique de précision en sortie d'une application utilisant le krigeage, une méthode géostatistique, a été développée. Celle-ci est illustrée sur l'algorithme de raffinage des largeurs d'une application convertie en arithmétique virgule fixe. Cette méthode permet de réduire le nombre d'évaluations de la métrique de qualité par simulations. Cette contribution n'a pas encore été publiée.

Pour finir, durant cette thèse, une autre contribution a été une forte implication dans les événements de vulgarisation scientifique. Le calcul approximé a été présenté dans plusieurs conférences pour le grand public pendant des événements tels que le "Festival des Sciences", "En Direct du Labo" ou encore "A la découverte de la recherche". Trois articles pour le grand public ont été écrits et sont cités ci-dessous:

1. Une présentation générale de la problématique d'économie des ressources énergétiques dans le domaine de l'Internet des Objets ainsi que les solutions apportées par le calcul approximé sont présentées dans [Bon18a].

2. Une courte présentation de la problématique globale de cette thèse est décrite dans l'article [Bon18b].

3. Une vue d'ensemble de l'utilisation de l'Internet des Objets en médecine et particulièrement en rythmologie est présentée dans l'article [Bon].

A.5 Plan de la thèse

Cette thèse est organisée en deux parties : la Partie I introduit le concept de calcul approximé et présente un panel de techniques d'approximations dans le **Chapitre 2**. Ensuite, les différentes métriques proposées pour caractériser les erreurs induites par les approximations sont présentées dans le **Chapitre 3**, ainsi que les modèles pour relier les erreurs à la qualité de service de l'application. La Partie II présente et évalue formellement les contributions de cette thèse. **Chapitre 4** présente une technique d'approximation au niveau algorithmique et l'évalue à l'aide de deux métriques d'erreur différentes. Ce chapitre expose les motivations de la thèse. Ensuite, les contributions principales de cette thèse sont présentées dans la Figure A.3. **Chapitre 5** présente la méthode proposée pour caractériser le profil des erreurs induites par l'utilisation d'opérateurs inexacts. **Chapitre 6** décrit les simulateurs conçus pour pouvoir simuler rapidement un opérateur inexact et mesurer son impact sur la qualité de service en sortie de l'application. Dans le **Chapitre 7**, un modèle basé sur des simulations pour l'évaluation de la précision des systèmes convertis en virgule fixe est présenté. Les statistiques inférentielles sont également utilisées pour ce modèle. Celui-ci a été implémenté dans un algorithme d'optimisation des largeurs des données codées en arithmétique virgule fixe. La méthode proposée est basée sur le modèle d'erreur statistique de la puissance de bruit proposé précédemment et réduit considérablement le nombre de simulations nécessaires à l'évaluation de la puissance de bruit. Cherchant à aller plus loin dans l'amélioration du temps d'évaluation de l'impact de la précision finie sur la qualité d'une application, **Chapitre 8** explique comment les simulations peuvent être évitées en estimant la métrique de précision à la sortie d'une application à l'aide de méthodes géostatistiques comme le krigeage. Enfin, **Chapitre 9** conclut cette thèse et propose des orientations pour de futurs travaux de recherche.

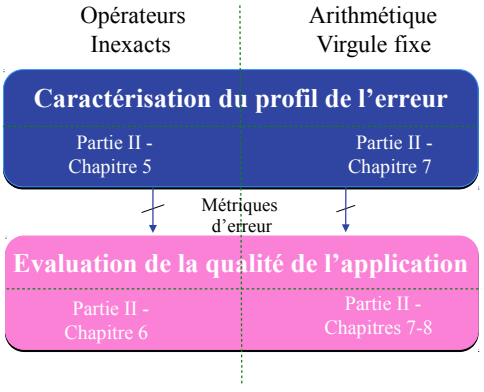


Figure A.3 – *Contributions de la thèse*

List of Figures

List of Tables

Acronyms

AA Affine Arithmetic. 68, 69, 186, 189

AAM Approximate Array Multiplier. 38–40, 49, 54, 55, 127–132, 134, 186

AC Approximate Computing. ii, 7–16, 19, 20, 22, 27, 45, 47–49, 51–54, 56–59, 61–63, 65, 66, 75, 76, 81–83, 85, 91, 94, 95, 101, 111, 133, 134, 160, 171–173

ACA Adder that takes only the k last bits to speculate the carry for each sum bit. 33–36, 49, 54, 55, 58, 64, 104, 106, 108, 116, 118, 127–134, 185

ADAS Advanced Driver-Assistance Systems. 82

ADSE Approximation Design Space Exploration. iii, 13, 77, 118, 119, 127, 133, 134, 172, 187, 190

AFA Approximate Full Adder. 32

AFAs Approximate Full Adders. 30, 31

ALU Arithmetic Logic Unit. 139

ASICs Application-Specific Integrated Circuits. 138

BALL Bit-Accurate Logic-Level. 54, 55, 63, 96, 105, 108, 110, 111, 116, 117, 120, 127–132, 134, 172, 186, 187, 189, 190

BER Bit Error Rate. 43, 50, 66, 74

BWER Bitwise Error Rate. 50

CAD Computer-Aided Design. 138

CCBA Carry-Cut Back Adder. 36, 108, 116, 118

CDF Cumulative Distribution Function. 101, 102

CLA Carry-Look-Ahead Adder. 27–29, 33, 35, 61

CORDIC COordinate Rotation DIgital Computer. 24, 25, 185

CPU Central Processing Unit. 118

CPUs Central Processing Units. 6

CSAs Carry-Save Adders. 37

DAG Directed Acyclic Graph. 29, 70–72, 186

DCT Discrete Cosine Transform. 19

DRAM Dynamic Random Access Memory. 43, 44, 56, 186

DSE Design Space Exploration. 12, 47, 130, 133, 134

DSP Digital Signal Processors. 41, 82

DSPs Digital Signal Processors. 19

DVFS Dynamic Voltage and Frequency Scaling. 43, 56, 62, 63, 73

ED Error Distance. 49, 59, 97, 105, 119, 125

ER Error Rate. ii, 34, 36, 42, 49, 58, 59, 64, 95–100, 102–106, 108, 109, 115, 118–121, 125, 126, 159, 172, 174, 187

ESA Equally Segmented Adder. 58

ETA Error-Tolerant Adder. 32–34, 185, 189

ETAII Error-Tolerant Adder Type II. 33–35, 58, 133, 185

ETAIV Error-Tolerant Adder Type IV. 33, 34, 185

ETAs Error-Tolerant Adders. 32

EVT Extreme Value Theory. 97, 101

FA Full Adder. 37, 41, 60, 61, 186, 189

FFT Fast Fourier Transform. 155, 168, 169

FIR Finite Impulse Response. 37, 143–146, 155, 157, 162, 167, 168, 187, 188

FnF Fast and Fuzzy. iii, 77, 117, 119–134, 137, 172, 187, 190

FNS Fixed-Number of Samples. 105, 107–109, 117, 189

FPGA Field-Programmable Gate Array. 42, 186

FPGAs Field Programmable Gate Arrays. 18, 138

GeCoS Generic Compiler Suite. 173, 174

GPP General Purpose Processor. 19

GPUs Graphics Processing Units. 82

HA Half Adder. 37

HEVC High Efficiency Video Coding. 22, 85, 155, 157, 168

HLS High-Level Synthesis. 138, 173

IA Interval Arithmetic. 50, 51, 67–69, 144, 186, 189

IIR Infinite Impulse Response. 67, 70, 141, 143, 144, 146, 155–157, 162, 168, 187

IMPACT IMPrecise Adder for low-power Approximate CompuTing. 31

IoT Internet of Things. 7, 14

ISA Inexact Speculative Adder. 35, 36, 104, 116, 185

LOA Lower-part-OR adder. 31, 32

LSB Least Significant Bits. 19, 30, 32, 43, 59

LSBs Least Significant Bits. 24, 29–33, 38, 39, 43, 58–60, 64, 70, 71, 122, 124

LTI Linear-Time Invariant. 74

LUT Look-Up Table. 12, 23, 25, 26, 73

LUTs Look-Up Tables. 24, 64

MA Mirror Adder. 30, 31

MAA Modified Affine Arithmetic. 64, 69, 70, 118

MAC Multiply-accumulate. 16, 118

maximum ED Maximum Error Distance. iii, 50, 62, 64, 95–97, 101–104, 109–115, 118, 120, 125, 172, 187, 189

mean ED Mean Error Distance. ii, 58, 59, 61–65, 95–99, 102–106, 108, 109, 115, 118, 120, 125, 159, 172, 186, 187

MIA Modified Interval Arithmetic. 51, 64, 69, 70, 73, 118

MSB Most Significant Bit. 19, 26, 30, 32, 38, 43, 121, 124, 126

MSBs Most Significant Bits. 24, 29–33, 38, 43, 59, 121, 122, 124, 125

MSE Mean-Squared Error. 91, 92

MSED Mean Squared Error Distance. 65

NRMSE Normalized Rooted Mean Squared Error. 130, 131, 187

OFDM Orthogonal Frequency-Division Multiplexing. 19

PDF Function that gives the probability for a given variable to fall between specific values, in the continuous case. 49

PMF Function that gives the probability for a given variable to be equal to a specific value, in the discrete case. 48–51, 59–61, 63, 64, 69–72, 118, 186

PRV Pseudo-Random Variable. iii, 117, 119–124, 132

PRVs Pseudo-Random Variables. 124

PSNR Peak Signal to Noise Ration. 65, 91

QoS Quality guaranteed by the programmer to the user, in terms of error amplitude or probability. 13, 14, 22, 52, 54, 67, 81, 83, 95, 96, 117–119, 127, 134, 147, 159, 160, 167, 169, 171, 172, 187

RCA Ripple-Carry-Adder. 27–29, 31–34, 37, 41, 42, 61, 186

RMS recognition, mining and synthesis. 13

RTL Register Transfer Level. 138

SCSA Speculative Carry Selection Adder. 58

SFG Signal Flow Graph. 74

SIMD Single Instruction Multiple Data. 82, 83

SKA Square Kilometer Array. 7, 176

SNR signal-to-noise ratio. 12, 16, 52

SoC System on Chip. 41, 43, 138

SQNR Signal-to-Quantization-Noise Ratio. 48, 69, 74

SRAM Static Random Access Memory. 43

SSIM Structural Similarity Index Measure. ii, 52, 53, 83, 91–94, 134, 174

STD Standard Deviation. 103

UDM Underdesigned Multiplier. 37, 38, 189

VLSA Variable Latency Speculative Adder. 33, 35

VLSI Very Large Scale Integration. 6, 37, 38

W-CDMA Wideband-Code Division Multiple Access. 16, 19

Personal Publications

Scientific publications

[BCDM18] Justine Bonnot, Vincent Camus, Karol Desnos, and Daniel Menard. Cassis: Characterization with adaptive sample-size inferential statistics applied to inexact circuits. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 677–681. IEEE, 2018. 13, 95, 116, 172, 183

[BDM18a] Justine Bonnot, Karol Desnos, and Daniel Menard. Stochastic modeling to accelerate approximate operators simulation. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2018. 13, 117, 134, 172, 183

[BDM18b] Justine Bonnot, Karol Desnos, and Daniel Ménard. Algorithm-level approximation for fast (or not) embedded stereovision algorithm. *Proceedings of International Conference on Embedded Computer Systems: Architectures, MOdeling and Simulation (SAMOS'18)*, 2018. 17, 81

[BDPM18] Justine Bonnot, Karol Desnos, Maxime Pelcat, and Daniel Menard. A fast and fuzzy functional simulator of inexact arithmetic operators for approximate computing systems. In *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, pages 195–200. ACM, 2018. 117

[BMD19] Justine Bonnot, Daniel Ménard, and Karol Desnos. Procédé et dispositif d'optimisation de longueurs de représentation de variables, April 2019. French Patent, FR1903747. 14, 138, 173, 183

[BCDM19] Justine Bonnot, Vincent Camus, Karol Desnos, and Daniel Menard. Adaptive simulation-based framework for error characterization of inexact circuits. *Microelectronics Reliability*, 96:60–70, 2019. 13, 95, 116, 172, 183

[BDM19a] Justine Bonnot, Karol Desnos, and Daniel Menard. Accuracy evaluation based on simulation for finite precision systems using inferential statistics. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1508–1512. IEEE, 2019. 14, 138, 160, 173, 183

[BDM19b] Justine Bonnot, Karol Desnos, and Daniel Menard. Fast simulation-based fixed-point refinement with inferential statistics (work in progress). In *DAC 2019-2019 Design Automation Conference (DAC)*. ACM, 2019. 138, 173

[MBP+17] Alexandre Mercat, Justine Bonnot, Maxime Pelcat, Karol Desnos, Wassim Hamidouche, and Daniel Menard. Smart search space reduction for approximate computing: A low energy hevc encoder case study. In *Journal of Systems Architecture*, 80:56–67, 2017. 85

[MBPHM17b] Alexandre Mercat, Justine Bonnot, Maxime Pelcat, Wassim Hamidouche, and Daniel Menard. Exploiting computation skip to reduce energy consumption by approximate computing, an hevc encoder case study. In *Proceedings of the Conference on Design, Automation & Test in Europe*, pages 494–499. European Design and Automation Association, 2017.

Popular science publications

[Bon] Justine Bonnot. Réduire la consommation énergétique de l'iot en cardiologie grâce au modèle mathématique des erreurs. 14, 184

[Bon18a] Justine Bonnot. Comment rendre les objets connectés moins énergivores ? https://theconversation.com/comment-rendre-les-objets-connectes-moins-energivores-102455, September 2018. 14, 183

[Bon18b] Justine Bonnot. Je diminue la consommation des objets connectés. https://www.espace-sciences.org/sciences-ouest/367/ce-que-je-cherche/justine-bonnot, December 2018. 14, 184

Bibliography

[ABBB13] Oliver Jakob Arndt, Daniel Becker, Christian Banz, and Holger Blume. Parallel implementation of real-time semi-global matching on embedded multi-core architectures. In *Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIII), 2013 International Conference on*, pages 56–63. IEEE, 2013. 82, 155

[AC15] Semiconductor Industries Association and Semiconductor Research Corporation. Rebooting the it revolution, a call for action. https://www.src.org/newsroom/rebooting-the-it-revolution.pdf, 2015. 5, 175

[ACN14] Roberto Airoldi, Fabio Campi, and Jari Nurmi. Approximate computing for complexity reduction in timing synchronization. *EURASIP Journal on Advances in Signal Processing*, 2014(1):155, 2014. 16

[ACV05] Carlos Alvarez, Jesus Corbal, and Mateo Valero. Fuzzy memoization for floating-point multimedia applications. *IEEE Transactions on Computers*, 54(7):922–927, 2005. 24

[AEJ+02] Alan Allan, Don Edenfeld, William H Joyner, Andrew B Kahng, Mike Rodgers, and Yervant Zorian. 2001 technology roadmap for semiconductors. *Computer*, 35(1):42–53, 2002. 43

[AH13] Armin Alaghi and John P Hayes. Survey of stochastic computing. *ACM Transactions on Embedded computing systems (TECS)*, 12(2s):92, 2013. 8, 177

[AKK15] Ismail Akturk, Karen Khatamifard, and Ulya R Karpuzcu. On quantification of accuracy loss in approximate computing. In *Workshop on Duplicating, Deconstructing and Debunking (WDDD)*, volume 15, 2015. 48, 49

[Ard02] Wolfgang M Arden. The international technology roadmap for semiconductors—perspectives and challenges for the next 15 years. *Current Opinion in Solid State and Materials Science*, 6(5):371–377, 2002. 6

[AT03] Behzad Akbarpour and Sofiene Tahar. Modeling systemc fixed-point arithmetic in hol. In *International Conference on Formal Engineering Methods*, pages 206–225. Springer, 2003. 55

[Bar17] Benjamin Barrois. *Methods to evaluate accuracy-energy trade-off in operator-level approximate computing*. PhD thesis, Rennes 1, 2017. 18, 39, 189

[BBW14] Harshita Bhurat, Tom Bryan, and Julia Wall. Best Practices for Converting MATLAB Code to Fixed Point. Technical report, MathWorks, 2014. 55

[BCDM18] Justine Bonnot, Vincent Camus, Karol Desnos, and Daniel Menard. Cassis: Characterization with adaptive sample-size inferential statistics applied to inexact circuits. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 677–681. IEEE, 2018. 13, 95, 116, 172, 183

[BCDM19] Justine Bonnot, Vincent Camus, Karol Desnos, and Daniel Menard. Adaptive simulation-based framework for error characterization of inexact circuits. *Microelectronics Reliability*, 96:60–70, 2019. 13, 95, 116, 172, 183

[BDM18a] Justine Bonnot, Karol Desnos, and Daniel Menard. Stochastic modeling to accelerate approximate operators simulation. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2018. 13, 117, 134, 172, 183

[BDM18b] Justine Bonnot, Karol Desnos, and Daniel Ménard. Algorithm-level approximation for fast (or not) embedded stereovision algorithm. *Proceedings of International Conference on Embedded Computer Systems: Architectures, MOdeling and Simulation (SAMOS'18)*, 2018. 17, 81

[BDM18c] Justine Bonnot, Karol Desnos, and Daniel Ménard. A fast and fuzzy functional simulator of inexact arithmetic operators for approximate computing systems. *GLSVLSI'18: 2018 Great Lakes Symposium on VLSI*, 2018. 13, 96, 100, 134, 172, 183

[BDM19a] Justine Bonnot, Karol Desnos, and Daniel Menard. Accuracy evaluation based on simulation for finite precision systems using inferential statistics. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1508–1512. IEEE, 2019. 14, 138, 160, 173, 183

[BDM19b] Justine Bonnot, Karol Desnos, and Daniel Menard. Fast simulation-based fixed-point refinement with inferential statistics (work in progress). In *DAC 2019-2019 Design Automation Conference (DAC)*. ACM, 2019. 138, 173

[BDPM18] Justine Bonnot, Karol Desnos, Maxime Pelcat, and Daniel Menard. A fast and fuzzy functional simulator of inexact arithmetic operators for approximate computing systems. In *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, pages 195–200. ACM, 2018. 117

[BHL07] Paul Busch, Teiko Heinonen, and Pekka Lahti. Heisenberg's uncertainty principle. *Physics Reports*, 452(6):155–176, 2007. 6

[BKM94] J-C Bajard, Sylvanus Kla, and J-M Muller. Bkm: a new hardware algorithm for complex elementary functions. *IEEE Transactions on Computers*, 43(8):955–963, 1994. 24

[BKSL08] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. The parsec benchmark suite: Characterization and architectural implications. In *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, pages 72–81. ACM, 2008. 21

[BMD19] Justine Bonnot, Daniel Ménard, and Karol Desnos. Procédé et dispositif d'optimisation de longueurs de représentation de variables, April 2019. French Patent, FR1903747. 14, 138, 173, 183

[BMP06] Hervé Brönnimann, Guillaume Melquiond, and Sylvain Pion. The design of the boost interval arithmetic library. *Theoretical Computer Science*, 351(1):111–118, 2006. 67

[BMS18] Alberto Bosio, Daniel Menard, and Olivier Sentieys. A Comprehensive Analysis of Approximate Computing Techniques: From Component- to Application-Level. ESWEEK 2018 - Embedded Systems Week, September 2018. 15

[BN04] Friedbert Berens and N Naser. Algorithm to system-on-chip design flow that leverages system studio and systemc 2.0. 1. *Synopsys Inc., March*, 2004. 55

[BNM16] Justine Bonnot, Erwan Nogues, and Daniel Menard. New non-uniform segmentation technique for software function evaluation. In *Application-specific Systems, Architectures and Processors (ASAP), 2016 IEEE 27th International Conference on*, pages 131–138. IEEE, 2016. 26

[Bon] Justine Bonnot. Réduire la consommation énergétique de l'iot en cardiologie grâce au modèle mathématique des erreurs. 14, 184

[Bon18a] Justine Bonnot. Comment rendre les objets connectés moins énergivores ? https://theconversation.com/comment-rendre-les-objets-connectes-moins-energivores-102455, September 2018. 14, 183

[Bon18b] Justine Bonnot. Je diminue la consommation des objets connectés. https://www.espace-sciences.org/sciences-ouest/367/ce-que-je-cherche/justine-bonnot, December 2018. 14, 184

[BS17] Benjamin Barrois and Olivier Sentieys. Customizing fixed-point and floating-point arithmetic-a case study in k-means clustering. In *SiPS 2017-IEEE International Workshop on Signal Processing Systems*, 2017. 140, 174

[BSM17] Benjamin Barrois, Olivier Sentieys, and Daniel Menard. The hidden cost of functional approximation against careful data sizing: a case study. In *Proceedings of the Conference on Design, Automation & Test in Europe*, pages 181–186. European Design and Automation Association, 2017. 17, 127, 138

[Ca65] J. Cooley and al. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965. 155, 168

[Ca01] M-A Cantin and al. An automatic word length determination method. In *Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on*, volume 5, pages 53–56. IEEE, 2001. 148, 150, 155, 162

[Ca02] G. A. Constantinides and al. The complexity of multiple wordlength assignment. *Applied mathematics letters*, 15(2):137–140, 2002. 138

[Ca03] G. A. Constantinides and al. Wordlength optimization for linear digital signal processing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(10):1432–1442, 2003. 148

[Ca09] G. Caffarena and al. Architectural synthesis of fixed-point dsp datapaths using fpgas. *International Journal of Reconfigurable Computing*, 2009:8, 2009. 148

[CCRR13] Vinay K Chippa, Srimat T Chakradhar, Kaushik Roy, and Anand Raghunathan. Analysis and characterization of inherent application resilience for approximate computing. In *Proceedings of the 50th Annual Design Automation Conference*, page 113. ACM, 2013. 12, 49, 52, 53, 66, 180, 186

[CCSE18] Vincent Camus, Mattia Cacciotti, Jeremy Schlachter, and Christian Enz. Design of approximate circuits by fabrication of false timing paths: The carry cut-back adder. In *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS)*, 2018. 54, 63, 64, 96, 108, 111, 118

[CDV12] Alexandre Chapoutot, Laurent-Stéphane Didier, and Fanny Villers. Range estimation of floating-point variables in simulink models. In *Design and Architectures for Signal and Image Processing (DASIP), 2012 Conference on*, pages 1–8. IEEE, 2012. 102

[CGS15] Fernando Cladera, Matthieu Gautier, and Olivier Sentieys. Energy-aware computing via adaptive precision under performance constraints in ofdm wireless receivers. In *VLSI (ISVLSI), 2015 IEEE Computer Society Annual Symposium on*, pages 591–596. IEEE, 2015. 19

[CJL10] Sylvain Chevillard, Mioara Joldeş, and Christoph Lauter. Sollya: An environment for the development of numerical codes. In *International Congress on Mathematical Software*, pages 28–31. Springer, 2010. 25

[CKK+13] Wei-Ting J Chan, Andrew B Kahng, Seokhyeong Kang, Rakesh Kumar, and John Sartori. Statistical analysis and modeling for error composition in approximate computation circuits. In *2013 IEEE 31st International Conference on Computer Design (ICCD)*, pages 47–53. IEEE, 2013. 64, 72

[CLFHC10] Gabriel Caffarena, Juan A López, Angel Fernández-Herrero, and Carlos Carreras. Sqnr estimation of non-linear fixed-point algorithms. In *2010 18th European Signal Processing Conference*, pages 522–526. IEEE, 2010. 69

[Con03] George A Constantinides. Perturbation analysis for word-length optimization. In *11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 2003. FCCM 2003.*, pages 81–90. IEEE, 2003. 139

[CSE15] Vincent Camus, Jeremy Schlachter, and Christian Enz. Energy-efficient inexact speculative adder with high performance and accuracy control. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 45–48. IEEE, 2015. 35, 50, 104

[CSE16] Vincent Camus, Jeremy Schlachter, and Christian Enz. A low-power carry cut-back approximate adder with fixed-point implementation and floating-point precision. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. Ieee, 2016. 36, 104

[CWK+15] Jeremy Constantin, Lai Wang, Georgios Karakonstantis, Anupam Chattopadhyay, and Andreas Burg. Exploiting dynamic timing margins in microprocessors for frequency-over-scaling with instruction-based clock adjustment. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015*, pages 381–386. IEEE, 2015. 42

[DDT05] Florent De Dinechin and Arnaud Tisserand. Multipartite table methods. *IEEE Transactions on Computers*, 54(3):319–330, 2005. 25

[DFS04] Luiz Henrique De Figueiredo and Jorge Stolfi. Affine arithmetic: concepts and applications. *Numerical Algorithms*, 37(1-4):147–158, 2004. 68

[DHSL09] Peter E Dewdney, Peter J Hall, Richard T Schilizzi, and T Joseph LW Lazio. The square kilometre array. *Proceedings of the IEEE*, 97(8):1482–1496, 2009. 7, 176

[Dum15] Maarten Dumont. *Real-Time View Interpolation for Eye Gaze Corrected Video Conferencing*. PhD thesis, transnationale Universiteit Limburg, 2015. 83

[DVM12] Kai Du, Peter Varman, and Kartik Mohanram. High performance reliable variable latency carry select addition. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012*, pages 1257–1262. IEEE, 2012. 54, 63, 96

[EDL+04] Dan Ernst, Shidhartha Das, Seokwoo Lee, David Blaauw, Todd Austin, Trevor Mudge, Nam Sung Kim, and Krisztián Flautner. Razor: circuit-level correction of timing errors for low-power operation. *IEEE Micro*, 24(6):10–20, 2004. 41, 42

[Fa99] P. D. Fiore and al. Closed-form and real-time wordlength adaptation. In *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, volume 4, pages 1897–1900. IEEE, 1999. 148

[FKB+15] Fabio Frustaci, Mahmood Khayatzadeh, David Blaauw, Dennis Sylvester, and Massimo Alioto. Sram for error-tolerant applications with dynamic energy-quality management in 28 nm cmos. *IEEE Journal of Solid-State Circuits*, 50(5):1310–1323, 2015. 43

[FPTV92] Brian P Flannery, William H Press, Saul A Teukolsky, and William Vetterling. Numerical recipes in c. *Press Syndicate of the University of Cambridge, New York*, 24:78, 1992. 167

[FRC03] Claire F Fang, Rob A Rutenbar, and Tsuhan Chen. Effcient static analysis of fixed-point error in dsp applications via affine arithmetic modeling. In *Proc. ICCAD*, pages 275–282. Citeseer, 2003. 69

[Fre15] Louis E Frenzel. *Handbook of serial communications interfaces: a comprehensive compendium of serial digital input/output (I/O) standards*. Newnes, 2015. 50

[Gai69] Brian R Gaines. Stochastic computing systems. In *Advances in information systems science*, pages 37–172. Springer, 1969. 8, 177

[GCH06] Olivier Gay, David Coeurjolly, and Nathan Hurst. Libaffa-c++ affine arithmetic library for gnu/linux, 2006. 68

[GK17] GA Gillani and Andre BJ Kokkeler. Improving error resilience analysis methodology of iterative workloads for approximate computing. In *Proceedings of the Computing Frontiers Conference*, pages 374–379. ACM, 2017. 53

[GLMS10] Thorsten Grtker, Stan Liao, Grant Martin, and Stuart Swan. System design with systemc. 2010. 55, 143, 154, 160

[GMP+11] 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. 30, 31

[GMRR13] 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. 60, 61, 186, 189

[Gra16] Mentor Graphics. Algorithmic c data-types, 2016. 160

[Gru99] Jozef Gruska. *Quantum computing*, volume 2005. McGraw-Hill London, 1999. 9, 178

[HEKC01] Kyungtae Han, Iksu Eo, Kyungsu Kim, and Hanjin Cho. Numerical word-length optimization for cdma demodulator. In *ISCAS 2001. The 2001 IEEE International Symposium on Circuits and Systems (Cat. No. 01CH37196)*, volume 4, pages 290–293. IEEE, 2001. 161

[HL11] Jiawei Huang and John Lach. Exploring the fidelity-efficiency design space using imprecise arithmetic. In *Proceedings of the 16th Asia and South Pacific Design Automation Conference*, pages 579–584. IEEE Press, 2011. 119

[HLR11] Jiawei Huang, John Lach, and Gabriel Robins. Analytic error modeling for imprecise arithmetic circuits. *Proc. SELSE*, 2011. 51, 64, 69, 72, 118

[HLR12] Jiawei Huang, John Lach, and Gabriel Robins. A methodology for energy-quality tradeoff using imprecise hardware. In *Proceedings of the 49th Annual Design Automation Conference*, pages 504–509. ACM, 2012. 64, 69, 74

[HMS08] Thibault Hilaire, Daniel Menard, and Olivier Sentieys. Bit accurate roundoff noise analysis of fixed-point linear controllers. In *2008 IEEE International Conference on Computer-Aided Control Systems*, pages 607–612. IEEE, 2008. 73

[HS98] Rajamohana Hegde and Naresh R Shanbhag. Energy-efficiency in presence of deep submicron noise. In *1998 IEEE/ACM International Conference on Computer-Aided Design. Digest of Technical Papers (IEEE Cat. No. 98CB36287)*, pages 228–234. IEEE, 1998. 41

[HS01] Rajamohana Hegde and Naresh R Shanbhag. Soft digital signal processing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 9(6):813–823, 2001. 13, 41, 42, 182

[Hut05] GD Hutcheson. The economic implications of moore's law. In *High Dielectric Constant Materials*, pages 1–30. Springer, 2005. 5, 175

[Hwa79] Kai Hwang. Computer arithmetic principles, architecture, and design. 1979. 38

[IHM+16] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016. 169

[Ins19] CB Insights. The future of data centers. https://www.cbinsights.com/research/future-of-data-centers/#storage, 2019. 7, 176

[Int] Intel. Moore's law timeline. http://download.intel.com/pressroom/kits/events/moores_law_40th/MLTimeline.pdf. 6, 185

[Ja10] H. Javaid and al. Fidelity metrics for estimation models. In *Proceedings of the International Conference on Computer-Aided Design*, pages 1–8. IEEE Press, 2010. 149

[JC08] Fabienne Jézéquel and Jean-Marie Chesneaux. Cadna: a library for estimating round-off error propagation. *Computer Physics Communications*, 178(12):933–955, 2008. 75

[JHL15] Honglan Jiang, Jie Han, and Fabrizio Lombardi. A comparative review and evaluation of approximate adders. In *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*, pages 343–348. ACM, 2015. 39

[JLL+17] H. Jiang, C. Liu, L. Liu, F. Lombardi, and J. Han. A review, classification and comparative evaluation of approximate arithmetic circuits. In *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 2017. 54, 63, 96

[KE12] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012. 82

[KEGA96] Sunder S Kidambi, Fayez El-Guibaly, and Andreas Antoniou. Area-efficient multipliers for digital signal processing applications. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 43(2):90–95, 1996. 39

[KGE11] 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. 37

[KHWC98] Holger Keding, Frank Hurtgen, Markus Willems, and Martin Coors. Transformation of floating-point into fixed-point algorithms by interpolation applying a statistical approach. In *9th International Conference on Signal Processing Applications and Technology (ICSPAT 98)*, 1998. 74, 140, 144, 145, 156

[KIK+98] Aggelos K Katsaggelos, Faisal Ishtiaq, Lisimachos P Kondi, M-C Hong, M Banham, and J Brailean. Error resilience and concealment in video coding. In *Signal Processing Conference (EUSIPCO 1998), 9th European*, pages 1–8. IEEE, 1998. 10, 179

[Kin85] Robert R Kinnison. *Applied extreme value statistics*. Battelle, 1985. 101, 102

[KWCM98] Holger Keding, Markus Willems, Martin Coors, and Heinrich Meyr. Fridge: a fixed-point design and simulation environment. In *Proceedings of the conference on Design, automation and test in Europe*, pages 429–435. IEEE Computer Society, 1998. 74, 140

[LCCNT08] Juan A López, Gabriel Caffarena, Carlos Carreras, and Octavio Nieto-Taladriz. Fast and accurate computation of the round-off noise of linear time-invariant systems. *IET Circuits, Devices & Systems*, 2(4):393–408, 2008. 74

[LCLV09] Dong-U Lee, Ray CC Cheung, Wayne Luk, and John D Villasenor. Hierarchical segmentation for hardware function evaluation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 17(1):103–116, 2009. 26

[LEN+11] Avinash Lingamneni, Christian Enz, Jean-Luc Nagel, Krishna Palem, and Christian Piguet. Energy parsimonious circuit design through probabilistic pruning. In *2011 Design, Automation & Test in Europe*, pages 1–6. IEEE, 2011. 29

[LGC+06] D-U Lee, Altaf Abdul Gaffar, Ray CC Cheung, Oskar Mencer, Wayne Luk, and George A Constantinides. Accuracy-guaranteed bit-width optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(10):1990–2000, 2006. 74

[LHG+16] Robert LiKamWa, Yunhui Hou, Julian Gao, Mia Polansky, and Lin Zhong. Redeye: analog convnet image sensor architecture for continuous mobile vision. In *ACM SIGARCH Computer Architecture News*, volume 44, pages 255–266. IEEE Press, 2016. 7

[LHL15] Cong Liu, Jie Han, and Fabrizio Lombardi. An analytical framework for evaluating the error characteristics of approximate adders. *IEEE Transactions on Computers*, 64(5):1268–1281, 2015. 58, 62, 65, 96

[Liu71] Bede Liu. Effect of finite word length on the accuracy of digital filters–a review. *IEEE Transactions on Circuit Theory*, 18(6):670–677, 1971. 73

[LNC96] Jeffrey T Ludwig, S Hamid Nawab, and Anantha P Chandrakasan. Low-power digital filtering using approximate processing. *IEEE Journal of Solid-State Circuits*, 31(3):395–400, 1996. 21

[LNLB16] Liming Lou, Paul Nguyen, Jason Lawrence, and Connelly Barnes. Image perforation: Automatically accelerating image pipelines by intelligently skipping samples. *ACM Transactions on Graphics (TOG)*, 35(5):153, 2016. 85

[Low14] Richard Lowry. Concepts and applications of inferential statistics. 2014. 98, 99, 100, 104, 140, 141, 149

[LPM18] Shikai Li, Sunghyun Park, and Scott Mahlke. Sculptor: Flexible approximation with selective dynamic loop perforation. In *Proceedings of the 2018 International Conference on Supercomputing*, pages 341–351. ACM, 2018. 21

[LYHW10] Shyue-Kung Lu, Chun-Lin Yang, Yuang-Cheng Hsiao, and Cheng-Wen Wu. Efficient bisr techniques for embedded memories considering cluster faults. *IEEE transactions on very large scale integration (VLSI) systems*, 18(2):184–193, 2010. 43

[LZP10] Yang Liu, Tong Zhang, and Keshab K Parhi. Computation error analysis in digital signal processing systems with overscaled supply voltage. *IEEE transactions on very large scale integration (VLSI) systems*, 18(4):517–526, 2010. 61, 63, 186

[Ma06] S-H Min and al. Hybrid genetic algorithms and support vector machines for bankruptcy prediction. *Expert systems with applications*, 31(3):652–660, 2006. 148

[MAFL10] Hamid Reza Mahdiani, Ali Ahmadi, Sied Mehdi Fakhraie, and Caro Lucas. Bio-inspired imprecise computational blocks for efficient vlsi implementation of soft-computing applications. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 57(4):850–862, 2010. 31

[Mal77] JP Malengé. Le point sur quelques méthodes directes de recherche de l'extremum d'une fonction monodimensionnelle unimmodale. *RAIRO-Operations Research-Recherche Opérationnelle*, 11(3):323–335, 1977. 85

[Mar14] Igor L Markov. Limits on fundamental limits to computation. *Nature*, 512(7513):147, 2014. 6, 176

[MBJ14] Joshua San Miguel, Mario Badr, and Natalie Enright Jerger. Load value approximation. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 127–139. IEEE Computer Society, 2014. 20

[MBP+17] Alexandre Mercat, Justine Bonnot, Maxime Pelcat, Karol Desnos, Wassim Hamidouche, and Daniel Menard. Smart search space reduction for approximate computing: A low energy hevc encoder case study. *Journal of Systems Architecture*, 80:56–67, 2017. 85

[MCR09] Jiayuan Meng, Srimat Chakradhar, and Anand Raghunathan. Best-effort parallel execution framework for recognition and mining applications. In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–12. IEEE, 2009. 20

[Men08] Daniel et al. Menard. Accuracy constraint determination in fixed-point system design. *EURASIP Journal on Embedded Systems*, 2008:1, 2008. 160

[MHH+17] Sana Mazahir, Osman Hasan, Rehan Hafiz, Muhammad Shafique, and Jörg Henkel. Probabilistic error modeling for approximate adders. *IEEE Transactions on Computers (TC)*, 66(3):515–530, 2017. 59, 62, 63, 64, 96, 119, 121, 186

[MHHS17] Sana Mazahir, Osman Hasan, Rehan Hafiz, and Muhammad Shafique. Probabilistic error analysis of approximate recursive multipliers. *IEEE Transactions on Computers*, 66(11):1982–1990, 2017. 59

[Mit16] Sparsh Mittal. A survey of architectural techniques for near-threshold computing. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 12(4):46, 2016. 40, 41, 186

[Moo62] Ramon E Moore. Interval arithmetic and automatic error analysis in digital computing. *Ph. D. Dissertation, Department of Mathematics, Stanford University*, 1962. 50

[Moo06] Gordon E Moore. Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp. 114 ff. *IEEE Solid-State Circuits Society Newsletter*, 11(3):33–35, 2006. 5, 175

[MPMN14] Judicaël Menant, Muriel Pressigout, Luce Morin, and Jean-Francois Nezan. Optimized fixed point implementation of a local stereo matching algorithm onto c66x dsp. In *Design and Architectures for Signal and Image Processing (DASIP), 2014 Conference on*, pages 1–6. IEEE, 2014. 82, 133

[MRR07] Wolfgang Müller, Wolfgang Rosenstiel, and Jürgen Ruf. *SystemC: methodologies and applications*. Springer Science & Business, 2007. 55, 154

[MS02] Daniel Menard and Olivier Sentieys. Automatic evaluation of the accuracy of fixed-point algorithms. In *Proceedings 2002 Design, Automation and Test in Europe Conference and Exhibition*, pages 529–535. IEEE, 2002. 74

[MS13] Raj Gaurav Mishra and Amit Kumar Shrivastava. Implementation of custom precision floating point arithmetic on fpgas. *HCTL Open International Journal of Technology Innovations and Research (IJTIR)*, 1:10–26, 2013. 18

[MVJ+09] Pramod K Meher, Javier Valls, Tso-Bing Juang, K Sridharan, and Koushik Maharatna. 50 years of cordic: Algorithms, architectures, and applications. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 56(9):1893–1907, 2009. 21, 24

[Na14] E. Nogues and al. Efficient fixed-point refinement of dsp dataflow systems. In *2014 IEEE Workshop on Signal Processing Systems (SiPS)*, pages 1–6, Oct 2014. 148

[NMP16] Erwan Nogues, Daniel Menard, and Maxime Pelcat. Algorithmic-level approximate computing applied to energy efficient hevc decoding. *IEEE Transactions on Emerging Topics in Computing*, 2016. 20, 22, 23

[NMS09] Hai-Nam Nguyen, Daniel Menard, and Olivier Sentieys. Dynamic precision scaling for low power wcdma receiver. In *2009 IEEE International Symposium on Circuits and Systems*, pages 205–208. IEEE, 2009. 19

[Oa14] A. V. Oppenheim and al. *Discrete-time signal processing*. Pearson Education, 2014. 155

[ÖNG08] Emre Özer, Andy P Nisbet, and David Gregg. A stochastic bitwidth estimation technique for compact and low-power custom processors. *ACM Transactions on Embedded Computing Systems (TECS)*, 7(3):34, 2008. 102

[Pal03] Krishna V Palem. Energy aware algorithm design via probabilistic computing: from algorithms and models to moore's law and novel (semiconductor) devices. In *Proceedings of the 2003 international conference on Compilers, architecture and synthesis for embedded systems*, pages 113–116. ACM, 2003. 9, 178

[Par07] Keshab K Parhi. *VLSI digital signal processing systems: design and implementation*. John Wiley & Sons, 2007. 161

[PCR10] Jongsun Park, Jung Hwan Choi, and Kaushik Roy. Dynamic bit-width adaptation in dct: an approach to trade off image quality and computation energy. *IEEE transactions on very large scale integration (VLSI) systems*, 18(5):787–793, 2010. 19

[PRMS10] Karthick Parashar, Romuald Rocher, Daniel Menard, and Olivier Sentieys. A hierarchical methodology for word-length optimization of signal processing systems. In *2010 23rd International Conference on VLSI Design*, pages 318–323. IEEE, 2010. 66, 160

[PTVF88] William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. Numerical recipes in c. *Cambridge University Press*, 1:3, 1988. 24, 25

[RCHS09] Andreas Reinhardt, Delphine Christin, Matthias Hollick, and Ralf Steinmetz. On the energy efficiency of lossless data compression in wireless sensor networks. In *2009 IEEE 34th Conference on Local Computer Networks*, pages 873–880. IEEE, 2009. 44

[RD15] Nele Reynders and Wim Dehaene. *Ultra-Low-Voltage Design of Energy-Efficient Digital Circuits*. Springer, 2015. 40

[RD18] Avishek Sinha Roy and Anindya Sundar Dhar. A novel approach for fast and accurate mean error distance computation in approximate adders. In *Circuits and Systems (ISCAS), 2018 IEEE International Symposium on*, pages 1–5. IEEE, 2018. 59, 62

[RHHF14] Amir Rahmati, Matthew Hicks, Daniel Holcomb, and Kevin Fu. Refreshing thoughts on dram: Power saving vs. data integrity. In *Workshop on Approximate Computing Across the System Stack (WACAS)*, 2014. 44

[RMSS07] Romuald Rocher, Daniel Menard, Olivier Sentieys, and Pascal Scalart. Analytical accuracy evaluation of fixed-point systems. In *2007 15th European Signal Processing Conference*, pages 999–1003. IEEE, 2007. 74

[RSNP12] Lakshminarayanan Renganarayana, Vijayalakshmi Srinivasan, Ravi Nair, and Daniel Prener. Programming with relaxed synchronization. In *Proceedings of the 2012 ACM workshop on Relaxing synchronization for multicore and manycore scalability*, pages 41–50. ACM, 2012. 19

[RTR07] Rolf-Dieter Reiss, Michael Thomas, and RD Reiss. *Statistical analysis of extreme values*, volume 2. Springer, 2007. 101

[Sa12] G. J. Sullivan and al. Overview of the high efficiency video coding(hevc) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012. 155, 168

[SAHH15] Muhammad Shafique, Waqas Ahmad, Rehan Hafiz, and Jörg Henkel. A low latency generic accuracy configurable adder. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2015. 63

[SB04] Changchun Shi and Robert W Brodersen. A perturbation theory on statistical quantization effects in fixed-point dsp with non-stationary inputs. In *Circuits and Systems, 2004. ISCAS'04. Proceedings of the 2004 International Symposium on*, volume 3, pages III–373. IEEE, 2004. 118

[SBR+15] Adrian Sampson, André Baixo, Benjamin Ransford, Thierry Moreau, Joshua Yip, Luis Ceze, and Mark Oskin. Accept: A programmer-guided compiler framework for practical approximate computing. *University of Washington Technical Report UW-CSE-15-01*, 1(2), 2015. 66

[SDF+11] Adrian Sampson, Werner Dietl, Emily Fortuna, Danushen Gnanapragasam, Luis Ceze, and Dan Grossman. Enerj: Approximate data types for safe and general low-power computation. In *ACM SIGPLAN Notices*, volume 46, pages 164–174. ACM, 2011. 56

[SDMHR11] Stelios Sidiroglou-Douskos, Sasa Misailovic, Henry Hoffmann, and Martin Rinard. Managing performance vs. accuracy trade-offs with loop perforation. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pages 124–134. ACM, 2011. 21

[Sed12] Enrique et al. Sedano. A fast interpolative wordlength optimization method for dsp systems. In *2012 VIII Southern Conference on Programmable Logic*, pages 1–6. IEEE, 2012. 161

[SGGL13] Vadim Smolyakov, Glenn Gulak, Timothy Gallagher, and Curtis Ling. Fault-tolerant embedded-memory strategy for baseband signal processing systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 21(7):1299–1307, 2013. 43

[Sho99] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999. 9, 178

[Sie10] MG Siegler. Eric schmidt: Every 2 days we create as much information as we did up to 2003. https://techcrunch.com/2010/08/04/schmidt-data/?guccounter=1&guce_referrer_us=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce_referrer_cs=HHr6zNdmkmnUod57tXPmPA, 2010. 7, 176

[SK00] Dennis Sylvester and Kurt Keutzer. A global wiring paradigm for deep submicron design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(2):242–252, 2000. 6

[SLC12] Enrique Sedano, Juan A López, and Carlos Carreras. Acceleration of monte-carlo simulation-based quantization of dsp systems. In *2012 19th International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 189–192. IEEE, 2012. 74

[Spr79] Melvin Dali Springer. The algebra of random variables. Technical report, 1979. 72

[SS03] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–195. IEEE, 2003. 86, 89, 133

[SS15] Deepashree Sengupta and Sachin S Sapatnekar. Femto: Fast error analysis in multipliers through topological traversal. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 294–299. IEEE Press, 2015. 72

[SSHS17] Deepashree Sengupta, Farhana Sharmin Snigdha, Jiang Hu, and Sachin S Sapatnekar. Saber: Selection of approximate bits for the design of error tolerant circuits. In *Proceedings of the 54th Annual Design Automation Conference 2017*, page 72. ACM, 2017. 60, 64, 70

[SSHS18] Deepashree Sengupta, Farhana Sharmin Snigdha, Jiang Hu, and Sachin S Sapatnekar. An analytical approach for error pmf characterization in approximate circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018. 60, 71

[SSRS15] Arjun Suresh, Bharath Narasimha Swamy, Erven Rohou, and André Seznec. Intercepting functions for memoization: A case study using transcendental functions. *ACM Transactions on Architecture and Code Optimization (TACO)*, 12(2):18, 2015. 23

[Str68] Anthony Strecok. On the calculation of the inverse of the error function. *Mathematics of Computation*, 22(101):144–158, 1968. 99

[SW14] Stefano Salvini and Stefan J Wijnholds. Fast gain calibration in radio astronomy using alternating direction implicit methods: Analysis and applications. *Astronomy & Astrophysics*, 571:A97, 2014. 53

[SWK+05] Giacinto P Saggese, Nicholas J Wang, Zbigniew T Kalbarczyk, Sanjay J Patel, and Ravishankar K Iyer. An experimental study of soft errors in microprocessors. *IEEE micro*, 25(6):30–39, 2005. 7

[TBJJ11] Yangyang Tang, Emmanuel Boutillon, Christophe Jégo, and Michel Jézéquel. Hardware efficiency versus error probability in unreliable computation. In *2011 IEEE Workshop on Signal Processing Systems (SiPS)*, pages 168–173. IEEE, 2011. 41

[VARR11] Rangharajan Venkatesan, Amit Agarwal, Kaushik Roy, and Anand Raghunathan. Macaco: Modeling and analysis of circuits for approximate computing. In *2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 667–673. IEEE, 2011. 73

[VBI08] Ajay K Verma, Philip Brisk, and Paolo Ienne. Variable latency speculative addition: A new paradigm for arithmetic circuit design. In *Proceedings of the conference on Design, automation and test in Europe*, pages 1250–1255. ACM, 2008. 29, 30, 33, 104

[VDZ08] Jeroen Van Der Zijp. Fast half float conversions. Technical report, Working paper, 2012ftp://www. fox-toolkit. org/pub/fasthalffloatconversion. pdf, 2008. 18

[VN56] John Von Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components. *Automata studies*, 34:43–98, 1956. 9, 178, 179

[VPC+15] Vassilis Vassiliadis, Konstantinos Parasyris, Charalambos Chalios, Christos D Antonopoulos, Spyros Lalis, Nikolaos Bellas, Hans Vandierendonck, and Dimitrios S Nikolopoulos. A programming model and runtime system for significance-aware energy-efficient computing. In *ACM SIGPLAN Notices*, volume 50, pages 275–276. ACM, 2015. 21

[VWF00] Lan-Da Van, Shuenn-Shyang Wang, and Wu-Shiung Feng. Design of the lower error fixed-width multiplier and its application. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 47(10):1112–1118, 2000. 38, 39, 127

[Wac14] Hans Wackernagel. Geostatistics. *Wiley StatsRef: Statistics Reference Online*, 2014. 162, 163

[WBSS04] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 12, 91, 134, 180

[Wid61] B. Widrow. Statistical analysis of amplitude-quantized sampled-data systems. *Transactions of the American Institute of Electrical Engineers, Part II: Applications and Industry*, 79(6):555–568, 1961. 139

[WK08] Bernard Widrow and István Kollár. Quantization noise. *Cambridge University Press*, 2, 2008. 122

[WKL96] Bernard Widrow, Istvan Kollar, and Ming-Chang Liu. Statistical theory of quantization. *IEEE Transactions on instrumentation and measurement*, 45(2):353–361, 1996. 57

[WLGQ17] Yi Wu, You Li, Xiangxuan Ge, and Weikang Qian. An accurate and efficient method to calculate the error statistics of block-based approximate adders. *arXiv preprint arXiv:1703.03522*, 2017. 58, 62, 96

[WP98] Suhrid A Wadekar and Alice C Parker. Accuracy sensitive word-length selection for algorithm optimization. In *Proceedings International Conference on Computer Design. VLSI in Computers and Processors (Cat. No. 98CB36273)*, pages 54–61. IEEE, 1998. 139

[Wys] Mark Wyse. Modeling approximate computing techniques. *Academic paper*. 66

[YMM13] Yavuz Yetim, Margaret Martonosi, and Sharad Malik. Extracting useful computation from error-prone processors for streaming applications. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 202–207. EDA Consortium, 2013. 11, 180

[ZCA+08] Dan Zuras, Mike Cowlishaw, Alex Aiken, Matthew Applegate, David Bailey, Steve Bass, Dileep Bhandarkar, Mahesh Bhat, David Bindel, Sylvie Boldo, et al. Ieee standard for floating-point arithmetic. *IEEE Std 754-2008*, pages 1–70, 2008. 17

[ZGWY10] 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. IEEE, 2010. 33

[ZGY09] Ning Zhu, Wang Ling Goh, and Kiat Seng Yeo. An enhanced low-power high-speed adder for error. 2009. 33, 34, 104, 189

[ZGZ+10] Ning Zhu, Wang Ling Goh, Weija Zhang, Kiat Seng Yeo, and Zhi Hui Kong. Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 18(8):1225–1229, 2010. 32

[ZYYX14] Qian Zhang, Feng Yuan, Rong Ye, and Qiang Xu. Approxit: An approximate computing framework for iterative methods. In *Proceedings of the 51st Annual Design Automation Conference*, pages 1–6. ACM, 2014. 22

INSA de RENNES
Service des Formations

AVIS DU JURY SUR LA REPRODUCTION DE LA THESE SOUTENUE

Titre de la thèse:
Error analysis for approximate computing systems

Nom Prénom de l'auteur : BONNOT JUSTINE

Membres du jury :
- Madame JEZEQUEL Fabienne
- Monsieur SENTIEYS Olivier
- Monsieur HILAIRE Thibault
- Monsieur BOSIO Alberto
- Monsieur JEGO Christophe
- Monsieur MENARD Daniel
- Monsieur DESNOS Karol

Président du jury : O. Sentieys

Date de la soutenance : 09 Octobre 2019

Reproduction de la these soutenue

☒ Thèse pouvant être reproduite en l'état
☐ Thèse pouvant être reproduite après corrections suggérées

Fait à Rennes, le 09 Octobre 2019

Signature du président de jury

Le Directeur,

M'hamed DRISSI

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

INSA RENNES

Titre : Analyse d'erreurs pour les systèmes utilisant des calculs approximés

Mots clés : approximation, erreur, qualité, énergie, statistiques inférentielles, systèmes embarqués

Résumé : Le calcul approximé est une technique de calcul efficace en énergie et reposant sur l'exploitation de la tolérance à l'imprécision d'une application. Développé pour faire face à la fin de la loi de Moore, il répond à une demande croissante en capacité de calcul. Les techniques d'approximation ont été proposées à différents niveaux d'abstraction, du circuit au système.
Cette thèse porte sur le développement de méthodes et outils permettant d'évaluer rapidement l'impact des différentes techniques d'approximation sur la qualité du résultat en sortie d'une application. L'étude des erreurs induites est essentielle pour utiliser ces approximations dans l'industrie. Deux niveaux d'approximation ont été considérés, le niveau matériel avec l'étude des opérateurs arithmétiques inexacts et le niveau des données avec l'étude de l'arithmétique virgule fixe.
Premièrement, des méthodes efficaces de caractérisation basées simulation ont été proposées pour obtenir des statistiques sur les erreurs induites par l'approximation considérée. Les statistiques inférentielles ont été utilisées pour quantifier le nombre d'observations nécessaires pour estimer les statistiques de l'erreur et ainsi réduire le temps d'évaluation. Les méthodes de caractérisation proposées sont basées sur des simulations adaptatives et caractérisent l'erreur d'approximation de façon statistique selon les exigences de confiance définies par l'utilisateur.
Ensuite, les métriques d'erreur obtenues ont été reliées à la métrique de qualité de l'application. Pour les opérateurs inexacts, un simulateur a été conçu pour le processus d'exploration de l'espace d'approximation pour sélectionner la meilleure pour l'application considérée. Pour la virgule fixe, le modèle d'erreur a été intégré à un algorithme de raffinage pour déterminer la largeur optimisée des variables de l'application.
Les résultats de cette thèse proposent des méthodes concrètes pour faciliter la mise en œuvre du calcul approximé dans les applications industrielles, accélérant les méthodes proposées dans l'état de l'art de un à trois ordres de grandeur.

Title: Error Analysis for Approximate Computing Systems

Keywords: approximation, error, quality, energy, inferential statistics, embedded systems

Abstract: Approximate Computing is an energy-aware computing technique that relies on the exploitation of the tolerance to imprecision of an application. Developed to face the end of Moore's law, it answers the growing demand in computing capacity. Approximation techniques have been proposed at different abstraction levels, from circuit to system level.
This thesis focuses on the development of methods and tools to quickly evaluate the impact of different Approximate Computing techniques on the application quality metric. The study of the induced errors is critical to use approximations in the industry. Approximate Computing techniques have been considered at two different levels, the hardware level with the study of inexact arithmetic operators and the data level with the study of fixed-point arithmetic. First, efficient simulation-based characterization methods have been proposed to derive statistics on the errors induced by the considered approximation. Inferential statistics have been proposed to reduce the time for error characterization. The proposed characterization methods are based on adaptive simulations and statistically characterizes the approximation error according to user-defined confidence requirements.
Then, the obtained error metrics are linked with the application quality metric. For inexact operators, a simulator has been proposed for the approximation design space exploration process to select the best approximation for the considered application. For fixed-point arithmetic, the proposed error model has been implemented in a fixed-point refinement algorithm to determine the optimized word-lengths of the internal variables in an application. The results of this thesis are proposing concrete methods to ease the implementation of Approximate Computing in industrial applications, speeding up state-of-the-art methods from one to three orders of magnitude.