Doctoral Dissertation
Doctoral Program in Computer Science and Systems Engineering (30^{th}cycle)

Design Techniques for Energy-Quality Scalable Digital Systems

By

Daniele Jahier Pagliari

******

Supervisor(s):
Prof. Enrico Macii
Prof. Massimo Poncino

Politecnico di Torino
2018

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Daniele Jahier Pagliari
2018

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

To the memory of my beloved grandfather Guido Jahier.

Acknowledgements

Despite all the difficulties, I will always remember the years of my Ph.D. as some of the best in my life. For this, I would like to express my deepest gratitude to the following people.

To Professor Enrico Macii for giving me the opportunity of undertaking this Ph.D. To Professor Massimo Poncino, for his invaluable advice and support throughout the years. For helping me grow as a researcher and as an engineer, sharing his immense expertise and knowledge with me, and for being both a mentor and a friend. Lastly, for transmitting me his unshakable and contagious optimism.

To the entire EDA Group, for welcoming me in their big family. In particular, to the guys and girls of "Lab 4" for being supportive and often helpful in my research, and even more for just being great friends. I will always treasure the moments spent together, be it in the lab or out for dinner.

To Edith Beigné, for giving me the opportunity of working at CEA, an experience that has enriched me greatly, both technically and personally. For being an example to aspire to, both in terms of technical proficiency, and as a human being. To Yves Durand, for his scientific advice and for his sense of humor. To the entire LISAN lab and in particular to David, for their endless support and patience. To my friends in Grenoble, for making my time there unforgettable. I wish I can see them all again very soon.

To my parents, for their unconditional and endless love. To my entire family and in particular to Sergio, whose voice and smile I miss greatly.

To my lifelong friends, for being on my side when I started, and still being there now. To Veronica, to whom I have already said everything, just for sharing another piece of the road with me.

Abstract

Energy efficiency is one of the key design goals in modern computing. Increasingly complex tasks are being executed in mobile devices and Internet of Things end-nodes, which are expected to operate for long time intervals, in the orders of months or years, with the limited energy budgets provided by small form-factor batteries.

Fortunately, many of such tasks are error resilient, meaning that they can tolerate some relaxation in the accuracy, precision or reliability of internal operations, without a significant impact on the overall output quality. The error resilience of an application may derive from a number of factors. The processing of analog sensor inputs measuring quantities from the physical world may not always require maximum precision, as the amount of information that can be extracted is limited by the presence of external noise. Outputs destined for human consumption may also contain small or occasional errors, thanks to the limited capabilities of our vision and hearing systems. Finally, some computational patterns commonly found in domains such as statistics, machine learning and operational research, naturally tend to reduce or eliminate errors.

Energy-Quality (EQ) scalable digital systems systematically trade off the quality of computations with energy efficiency, by relaxing the precision, the accuracy, or the reliability of internal software and hardware components in exchange for energy reductions. This design paradigm is believed to offer one of the most promising solutions to the impelling need for low-energy computing.

Despite these high expectations, the current state-of-the-art in EQ scalable design suffers from important shortcomings. First, the great majority of techniques proposed in literature focus only on processing hardware and software components. Nonetheless, for many real devices, processing contributes only to a small portion of the total energy consumption, which is dominated by other components (e.g. I/O, memory or data transfers). Second, in order to fulfill its promises and become

diffused in commercial devices, EQ scalable design needs to achieve industrial level maturity. This involves moving from purely academic research based on high-level models and theoretical assumptions to engineered flows compatible with existing industry standards. Third, the time-varying nature of error tolerance, both among different applications and within a single task, should become more central in the proposed design methods. This involves designing "dynamic" systems in which the precision or reliability of operations (and consequently their energy consumption) can be dynamically tuned at runtime, rather than "static" solutions, in which the output quality is fixed at design-time.

This thesis introduces several new EQ scalable design techniques for digital systems that take the previous observations into account. Besides processing, the proposed methods apply the principles of EQ scalable design also to interconnects and peripherals, which are often relevant contributors to the total energy in sensor nodes and mobile systems respectively. Regardless of the target component, the presented techniques pay special attention to the accurate evaluation of benefits and overheads deriving from EQ scalability, using industrial-level models, and on the integration with existing standard tools and protocols. Moreover, all the works presented in this thesis allow the dynamic reconfiguration of output quality and energy consumption.

More specifically, the contribution of this thesis is divided in three parts. In a first body of work, the design of EQ scalable modules for processing hardware data paths is considered. Three design flows are presented, targeting different technologies and exploiting different ways to achieve EQ scalability, i.e. timing-induced errors and precision reduction. These works are inspired by previous approaches from the literature, namely *Reduced-Precision Redundancy* and *Dynamic Accuracy Scaling*, which are re-thought to make them compatible with standard *Electronic Design Automation* (EDA) tools and flows, providing solutions to overcome their main limitations.

The second part of the thesis investigates the application of EQ scalable design to serial interconnects, which are the de facto standard for data exchanges between processing hardware and sensors. In this context, two novel bus encodings are proposed, called *Approximate Differential Encoding* and *Serial-T0*, that exploit the statistical characteristics of data produced by sensors to reduce the energy consumption on the bus at the cost of controlled data approximations. The two techniques achieve different results for data of different origins, but share the common features of allowing runtime reconfiguration of the allowed error and being compatible with standard serial bus protocols.

Finally, the last part of the manuscript is devoted to the application of EQ scalable design principles to displays, which are often among the most energy-hungry components in mobile systems. The two proposals in this context leverage the emissive nature of *Organic Light-Emitting Diode* (OLED) displays to save energy by altering the displayed image, thus inducing an output quality reduction that depends on the amount of such alteration. The first technique implements an image-adaptive form of brightness scaling, whose outputs are optimized in terms of balance between power consumption and similarity with the input. The second approach achieves concurrent power reduction and image enhancement, by means of an adaptive polynomial transformation. Both solutions focus on minimizing the overheads associated with a real-time implementation of the transformations in software or hardware, so that these do not offset the savings in the display.

For each of these three topics, results show that the aforementioned goal of building EQ scalable systems compatible with existing best practices and mature for being integrated in commercial devices can be effectively achieved. Moreover, they also show that very simple and similar principles can be applied to design EQ scalable versions of different system components (processing, peripherals and I/O), and to equip these components with knobs for the runtime reconfiguration of the energy versus quality tradeoff.

Contents

List of Figures

List of Tables

Chapter 1

The Energy-Quality Tradeoff

Energy efficiency has become one of the main objectives in the design of modern digital systems, due to a combination of economic and technological drivers.

On the economic side, the rise of the ubiquitous computing paradigm and of the *Internet of Things* (IoT) spurred the diffusion of battery-powered devices, which are supposed to operate autonomously for long periods of time, with small form factors and very limited cost budgets [2].

At the same time, at the end of Dennard's scaling era [3, 4] classical technological knobs to achieve energy efficiency are providing diminishing returns. *Metal-Oxide-Semiconductor Field Effect Transistors* (MOSFET) feature scaling has been historically responsible for most of the energy reduction, approximately 100x per decade, in digital *Integrated Circuits* (ICs) [5]. However, with integration approaching its physical limits, scaling is expected to continue only for few generations [6]. Moreover, the manufacturing cost per transistor has been increasing after 28nm, hindering the adoption of the newest technology nodes in highly cost-constrained IoT end-nodes. Supply voltage (V_{DD}) scaling, the other main knob for energy efficiency in digital ICs, has practically stopped as V_{DD} is approaching the transistors threshold voltage (V_{th}), which is in turn not being reduced due to its exponential influence on leakage power.

In classical digital designs, a significant portion of the total energy is spent in ensuring that the system produces reliable, precise and accurate outputs, that are as close as possible to the *correct* result for the problem at hand [5, 7, 8]. One example is the use of large bit-width floating point numeric formats, whose superior flexibility and precision (on average) compared to fixed point is paid by a more complex and power-hungry hardware implementation. Similarly, the use of space and time redundancy at different levels of granularity is also representative of this trend, from *Error-Correcting Codes* (ECC) in communication channels and memories, to duplicated hardware modules, software functions, or entire devices.

If the need for energy efficiency had not been so pressing, this trend would have probably continued. However, the aforementioned market and technology directions require to find new knobs to minimize consumption. To this end, analyses of prominent applications in modern computing show that significant savings can come exactly from a relaxation of reliability, accuracy and precision constraints [5, 7–9].

Indeed, in the early stages of computing history, most applications were in the military, scientific and financial domains, which naturally called for reliability [7]. Conversely, modern computers are increasingly involved in consumer applications, such as audio/video processing, gaming, web browsing, etc., in which the strict correctness of results is less critical. Moreover, a large fraction of these applications, especially in the IoT domain, interact with the physical world and/or with humans (e.g. sensing and actuation, signal processing, data display, etc.). In the former case, producing highly accurate results is often *impossible*, due to the inherent noise of physical signals, while in the latter case it can be *unnecessary*, as human sense organs might not be able to notice the difference.

In these and other applications, the *quality* of outputs, defined as their usability from the perspective of the system's intent, is not dramatically affected by small or occasional *errors*, i.e. relaxations in the reliability, accuracy or precision of calculations. For this reason, we broadly refer to them as *error tolerant* or *error resilient* [5, 7–9]. A more detailed analysis of the features that cause error tolerance is provided in Section 1.1.

For an error tolerant application, the classical design measures for producing highly reliable, precise and accurate results are often unnecessary. This property can be leveraged to obtain significant energy efficiency improvements, by reducing or eliminating the aforementioned overheads involved with design techniques for "correctness" [7].

Spurring from these observations, a new paradigm for designing digital system, generally described as *Energy-Quality* (EQ) scalable, is gaining significant traction in recent years. In this paradigm, the quality of a system's outputs is systematically traded off for energy efficiency. To some extent, some authors consider EQ scalable design techniques as the most promising ways to maintain the promises of the IoT, despite the approaching end of classing scaling [5].

It should be noted that some of the basic concepts behind this paradigm are as old as the history of computing. In fact, the idea of tailoring the complexity and cost of a system to be "just enough" for its outputs to be "useful" is a basic principle of digital design, and engineering in general. Examples of applications of this concept are found throughout domains, from best-effort transmission in networking (e.g. UDP) [10], to lossy data compression (e.g. JPEG) [11].

The novelty in EQ scalable design lies in the *systematic* application of this principle. Output quality is considered as *a new optimization dimension*, along with the traditional metrics of performance, energy consumption and cost [5, 12]. Ideally, the objective of this paradigm is to build platforms that allow a *holistic tuning* of the quality and energy efficiency of a system, not restricted to application design, but permeating all layers of the computing stack (device, circuit, architecture, system software, etc.) and all system components (processing, memory, I/O, etc.) [13]. Jointly, an industrial application of this approach calls for *Electronic Design Automation* (EDA) methods able automate the implementation of such platforms. Currently, as discussed in Section 1.6, this goal has only been partially achieved. Nonetheless, the EQ scalable paradigm is only in its infancy, and several efforts from both research and industry, including in some small way this manuscript, are directed towards this goal.

In different research communities, EQ scalable designs have been realized with different approaches, and have been referred to with several names. Some of the most popular embodiments of this paradigm include *approximate computing* [8], *dynamic effort scaling* [12], *algorithmic noise tolerance* [14], *best-effort computing* [15] and, to some extent, *stochastic computing* [16].

In the rest of this chapter, we will first analyze more in depth the enabling factors for EQ scalable design, and then provide an overview of the main practical embodiments of this paradigm. The content presented in the following is an extended and revised version of our previous publications, mainly the work of [17].

1.1 Error Tolerance in Modern Computing

Error tolerance, or resilience, is a property of a computing application [9, 18, 7]. It can be broadly defined as the ability of the application to tolerate some errors or losses of precision in internal computations, without a significant impact on the quality of outputs[1]. In this definition, output quality is considered from the perspective of the user of an application, which can be either the final human user, or another application performing some further processing on its outputs [18].

For an error tolerant application, the quality of outputs can be measured as a *continuous function* of the quality of computations, rather than as a *boolean* one. Consequently, outputs are deemed *acceptable* or not, depending on their associated quality level, on the application requirements, and on the current environmental conditions, rather than simply *correct* or *wrong* [9].

Several factors affect the error resilience of an application, and each author provides his/her own categorization [7–9, 18, 13, 5]. The simplest and most general subdivision distinguishes the factors that cause error tolerance in those related to *inputs*, *outputs*, and *application internals*, as detailed in Sections 1.1.1-1.1.3. A diagram reporting an overview of this categorization is shown in Figure 1.1.

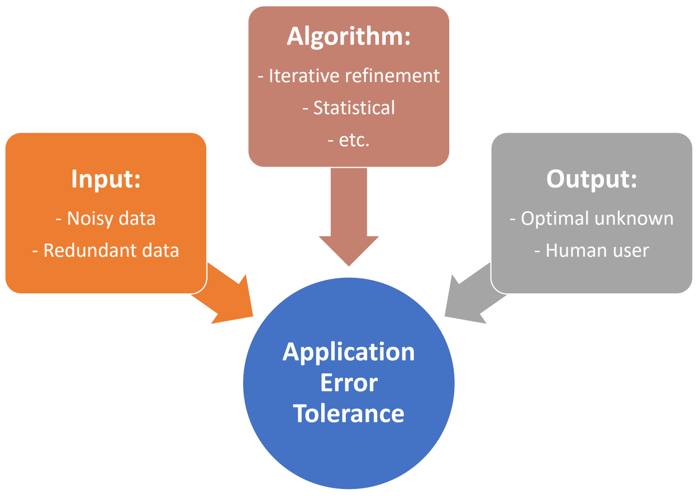


Fig. 1.1 Overview of main factors that affect the error resilience of an application.

[1]Note that here the term "quality" is not used as the opposite of "quantity" but as a synonym for "goodness".

1.1.1 Input features

Error tolerance can arise from the fact that an application processes inputs that are *noisy* and/or *redundant*. Examples of this very common situation have been anticipated in the introduction to this chapter. The typical case is that of applications that process data coming from sensors that sample physical quantities from the environment, e.g. in sensor nodes for the IoT, control systems, human-machine interfaces, radio transceivers, etc.

Indeed, all sensor are inherently affected by environmental noise and interference. This noise naturally propagates to the internals of an application, perturbing its state. This is exactly the same effect produced by small errors or precision reductions introduced intentionally within computations. Therefore, in presence of noisy inputs, errors in computation can be tolerated, as long as their effects on the final results are smaller or comparable with respect to the effects of noise. Effectively, the latter constitute an *upper bound* on the achievable quality of outputs.

Redundant inputs, instead, are common in statistical and machine learning applications. In this case, some computations can be approximated or even skipped completely, because they do not add information (quality) to the results. The latter aspect is partially related also to the nature of the algorithms involved in these applications, as explained in Section 1.1.3.

1.1.2 Output features

The influence of application outputs on the error resilience of an application is twofold. Error tolerance can arise either when the definition of optimal, "golden" results is informal or fuzzy, or when outputs are destined to be used by a human.

The first situation may occur because multiple outcomes are equivalently valuable for the purposes of the system mission. This is the case for many data mining tasks; as an example, two similar outputs produced by a web browser search engine might be equally good for the end-user, and it could very difficult to rank them.

Alternatively, the optimality of the results produced by a computation can be inherently unknown, because of the presence of random, semi-random or heuristic operations. In those situations, slightly perturbing an internal operation might not worsen, or even improve the quality of the final results. Applications in the domains of optimization and operation research (e.g. stochastic gradient descent, simulated annealing, genetic algorithms, etc.) often fall in this category. Again, there is not a net distinction among these features, which are conceptually related to outputs, and the characteristics of the algorithms that produce them, analyzed in Section 1.1.3. In literature, the broad family of *Recognition, Mining and Synthesis* (RMS) applications, is often reported as a significant example of error resilience [9].

Resilience can also come from the fact that the outputs of a computing application are meant to be used by people. Human sense organs used to interact with computers, mainly sight and hearing, have a limited resolution both in time and space. In other words, we are unable to distinguish very small or sparse differences, or very fast variations in videos or sounds.

An example is shown in Figure 1.2. The rightmost picture has been generated starting from the original shown on the left, by introducing a random “error” in every pixel. Specifically, the 3 *Least Significant Bits* (LSBs) of each 8-bit color component of a pixel (the image is in 24-bit RGB format) have been replaced with uniformly drawn random noise. Nonetheless, at normal viewing distance, the two images look identical, and only by zooming in on the background it is possible to notice the distortion introduced by “errors” on LSBs.

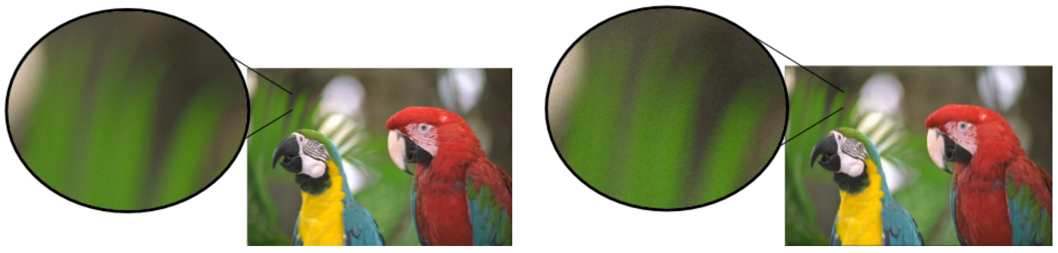

(a) Original image. (b) Random 3 LSBs.

Fig. 1.2 Example of the effect of small errors on perceived quality for an image.

In general, errors are unnoticed by our sense organs if they are either rare or small. This concept is further analyzed in Section 1.2.3. Considering the visual system, two full-HD images that differ only in few pixels are hardly distinguishable to the naked eye (rare error in the spatial dimension). Similarly, a single altered frame in a video stream is also practically unnoticeable (rare error in the temporal dimension). Conversely, two images or videos are still perceived identical, even if differences are much more frequent in space and time, as long as the pixels colors are only slightly altered (small error). This is the case of Figure 1.2, where *all* pixels have been perturbed, but the relative value difference has been kept small by modifying only LSBs.

Most applications that are error resilient due to the limited capabilities of human sense organs belong to the domains of multimedia and telecommunication e.g. audio/video compression, imaging tasks, etc.

Notice that the two output-related error resilience factors mentioned in this section are strongly related to each other. For instance, the fact that certain differences in an image are not noticeable by humans can be interpreted as a quality equivalence among multiple outputs.

1.1.3 Algorithm features

The error resilience of an application can also stem from the inner characteristics of the involved algorithms. Specifically, there are computational patterns that naturally favor the mitigation or the rejection of errors.

A typical example of such patterns is *iterative refinement*, i.e. the generic paradigm of finding a solution to a problem by iteratively improving an initial inexact (e.g. random) solution. This pattern is used in many applications, from recognition and mining to optimization and to the solution of systems of linear equations [19]. The progressive improvement of the considered solution can be seen as a "movement" on the space of all possible feasible solutions. In presence of computational errors or precision reductions, as long as these do not "deviate" too much the direction of this movement (e.g. leading to worsen the current solution) it can be formally proven that the algorithm will still converge to the same final result, although possibly in more iterations [20].

Algorithms that involve *statistical aggregations*, i.e. those that extract compact knowledge from large amounts of data, are also very good in rejecting errors. Indeed, occasional deviations on single data tend to have little relevance on the final results, as their impact is "averaged" over the entire set of data. Moreover, unbiased errors (i.e. whose mean effect is zero), tend to cancel with each other in presence of aggregations [5].

Once again, it is important to remark that there is clearly a partial overlap between input, output and algorithm-dependent features that impact error resilience. For example, pseudo-randomness and heuristic decisions, mentioned in Section 1.1.2 can be also thought of as algorithmic features.

1.2 Measuring Quality

In order to leverage the inherent error tolerance of an application for the design of a digital system, it is fundamental to determine what kind of errors are *acceptable*. In Section 1.1 we have explained that, for an error resilient task, the quality of results can be measured on a continuous scale. Therefore, the problem reduces to measuring this quality, and imposing constraints on it (e.g. a minimum threshold). Computational errors are deemed acceptable as long as they do not let output quality decrease below the threshold, and vice versa. In other words, since EQ scalable designs optimize the tradeoff between output quality and energy, both dimensions of this tradeoff must be quantifiable [5, 19].

Well-established models are available in literature for measuring power and energy consumption in digital circuits [21]. In this section, we briefly analyze the main metrics for assessing quality, and the basic principles related to this assessment.

In an EQ scalable design, the final output quality is a function of the errors or precision reductions purposely introduced by the designer in the system. Therefore, in this context, *quality metrics* and *error metrics* are often used interchangeably. In general, a quality metric can be obtained by taking the inverse of the corresponding error metric.

As we will detail in Section 1.2.4, there is no universal metric for quality and/or error. Nonetheless, most metrics depend on the combination of two basic factors: *Error Rate* and *Error Significance*. We describe these two factors in Section 1.2.1. We then go on to list some of the most used error/quality metrics in Section 1.2.2. Finally, Sections 1.2.3-1.2.4 analyze some of the basic principles related to quality measurement.

1.2.1 Error Rate and Error Significance

The overall quality of outputs of a computing system affected by errors is almost invariably influenced by two concurring factors, commonly referred to as *Error Rate* (*ER*) and *Error Significance* or magnitude (*ES*) [19].

The *ER* is the *rate* or *frequency* of occurrence of errors. It is most commonly measured as the number of erroneous outputs over a given time interval, divided by the total number of outputs in that interval, both erroneous (N_e) and correct (N_c):

$$ER = \frac{N_e}{(N_e + N_c)} \tag{1.1}$$

The *ER* can be considered informally as the *average probability* of occurrence of an error over the considered interval [9].

The *ES* defines the *severity* of a single error. While the definition of error rate is common to most system, the same is not true for *ES*, since the notion of severity is strictly related to the objective of an application. Therefore, there are a number of different expressions for measuring *ES* in literature.

In general, *ES* is measured as the *deviation* of an erroneous value (V_e) from the corresponding correct value (V_c) at a given time instant. The simplest of such measures is the *numerical difference D* (in absolute value), defined as [22, 23]:

$$||D|| = ||V_c - V_e|| \tag{1.2}$$

Alternatively, the squared difference D^2 is often used in place of the absolute value. Both $||D||$ and D^2 can also be converted to *relative differences*, by normalizing them to the correct value. For example:

$$RD = \left| \frac{V_c - V_e}{V_c} \right| \tag{1.3}$$

The previous three metrics are used when the considered datum has numerical interpretation. The *Hamming Distance* (*HD*), instead, is a commonly found alternative when all bits of the datum datum have equal importance. It is defined as the number of bits that differ between the binary representations of V_c and V_e [23]:

$$HD = \sum_{bit(i)=1} (V_c \oplus V_e) \tag{1.4}$$

where $\oplus$ is the bitwise XOR operator.

1.2.2 Composite Metrics

For most realistic applications, quality is evaluated by some form of composite metric, that encompasses the information of both *ER* and *ES* [24–26].

One such metric is the *Rate Significance* (*RS*), defined as the product between the rate and the *maximum* significance over a set of vectors [26]:

$$RS = ER \cdot \max(ES) \tag{1.5}$$

where *ER* and *ES* are defined according to one of (1.1) and (1.2-1.4). There are also variants of the *RS* expression in which *ER* and *ES* are assigned different importance by means of weighting factors [26].

Another very common combined metric is *Mean Error Distance* (*MED*), defined as the mean of $||D||$, computed as in (1.2), over a set of N output vectors [25]:

$$MED = \frac{1}{N}\sum_{i=1}^{N} ||V_{c,i} - V_{e,i}|| = \mathbb{E}_T\left[||V_c - V_e||\right] \tag{1.6}$$

Here, the significance is accounted by $||D||$, whereas the error rate is implicitly considered by taking the average, since in general, the less errors occur, the smaller the *MED*. The above definition corresponds to the sample average of the error distance over time ($\mathbb{E}_T$), where $\mathbb{E}$ stands for expected value. If errors occurring at a given time instant only depend on the value of the correct output in the same instant, *MED* can alternatively be defined as:

$$MED = \sum_{j=1}^{M} ||V_{c,j} - V_{e,j}|| \cdot \mathbb{P}(V_{c,j}) = \mathbb{E}_S\left[||V_c - V_e||\right] \tag{1.7}$$

where S is the set of possible values assumed by the correct output, composed of M elements (e.g. $[0 : M-1]$), and $\mathbb{P}(V_{c,j})$ is the probability of occurrence of value j. In most cases, however, errors do not depend only on the magnitude of the correct value, but are also influenced either by previous data history, or by external conditions. Therefore, the definition of (1.6) is by far the most used.

The *Normalized Error Distance* (*NED*) is simply the *MED* normalized to the maximum possible value on a given output [25]:

$$NED = \frac{MED}{\max(\|D\|)} \tag{1.8}$$

This metric is commonly used in place of the *MED* whenever it is necessary to compare errors on data that have different ranges of variations (e.g. binary variables with different number of bits), or different data representations.

A common alternative to the *MED* is the *Mean Squared Error* (*MSE*), defined as [24]:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(V_{c,i} - V_{e,i})^2 = \mathbb{E}_T\left[(V_c - V_e)^2\right] \tag{1.9}$$

As before, under the assumption that errors only depend on current output values:

$$MSE = \mathbb{E}_S\left[(V_c - V_e)^2\right] \tag{1.10}$$

In signal processing and communication applications, a commonly found metric is the *Signal-to-Noise Ratio* (*SNR*), defined as [27, 24, 14, 28]:

$$SNR = \frac{\sigma_s^2}{\sigma_w^2} \tag{1.11}$$

where σ_s^2 is the power of the *signal* component, i.e. the set of error-free values carrying useful information and σ_w^2 is the power of the *noise* component, i.e. the combination of all disturbances affecting the measured values.

When using this metric, errors in computations are modeled as an additional *noise source* (σ_{err}^2). Therefore, one advantage of the *SNR* is being able to combine in a single quality metric both purposely introduced errors and other sources of noise (e.g. not ideality of analog components, external environmental noise, etc.). In particular, assuming that computational errors and other sources of noise are *uncorrelated*, the definition of *SNR* becomes [28]:

$$SNR = \frac{\sigma_s^2}{\sigma_{noise}^2 + \sigma_{err}^2} \tag{1.12}$$

This quantity is is often expressed in decibel (dB). Notice that *SNR* is strictly related to MSE, as the "noise" due to errors (N_{appr}) is typically defined as the numerical deviation between the correct and erroneous value:

$$N_{appr} = D \tag{1.13}$$

and hence its average power σ_{appr}^2 is exactly expressed by (1.9).

A further variant of the *SNR* is the *Peak Signal to Noise Ratio* (*PSNR*), which is computed as the ratio between the *maximum possible value* assumed by the correct data and the noise variance:

$$PSNR = \frac{\max(\|V_c\|)}{\sigma_{noise}^2 + \sigma_{err}^2} \tag{1.14}$$

All above metrics are somewhat general purpose and used across application domains. Even the *SNR* and its variants, which are traditionally specific to the signal processing domain, have been adopted for example by designers of EQ scalable arithmetic circuits [28]. However, for most complex applications, the final output quality cannot be captured by any of the simple metrics mentioned above. In contrast, a high-level metric is typically used, that captures the notion of quality from the perspective of a user of the system.

High-level metrics are invariably specific to a single domain and may even differ among applications belonging to the same domain [9, 29, 30]. For example, the quality of a machine learning algorithm can be measured by classification accuracy or F-score; for an image processing application, a proxy of the perceived similarity among the input and output images is measured by the *Mean Structural Similarity Index* (MSSIM) [31], etc. An exhaustive enumeration of all possible domain specific metrics is out of the scope of this work.

1.2.3 The Fail Small or Fail Rare Concept

Regardless of the considered domain, the great majority of high level quality/error metrics still measure a combination of error rate (ER) and significance (ES). Clearly, both these components should be reduced in order to increase quality. However, it is often sufficient to minimize one of the two in order to obtain an acceptable output.

For example, considering the RS metric of (1.5), it is evident that as long as the ER is very low, the value of the error will be low as well, regardless of the maximum ES. Vice versa, a higher error rate could still produce a small final error, given that the maximum significance is kept contained.

We used the RS in this example for simplicity of explanation, but the same concept tends to apply also to most high-level metrics. A visual example of the same effect has been provided in Section 1.1.2 and Figure 1.2.

This relation between computational errors and output quality is perfectly summarized by the *"fail small or fail rare"* concept, formulated and verified experimentally in [9]. In this work, the authors provide a rule of thumb for designing EQ scalable systems, that can be formulated as follows: *the quality of outputs produced by an EQ scalable system acting over a set of inputs for a given period of time is acceptable if the computational errors introduced in the system are either small in significance or rare in time.* The fail small or fail rare rule is an essential guideline to the design of EQ scalable systems.

1.2.4 The Time Dependence of Quality

Perhaps the most relevant feature of an application's output quality is its *dynamicity* over time. In particular, the notion of *acceptable quality* can be shown to vary in response to several factors [5, 9]. This has a strong impact on the type of errors that can be introduced in a system at any given time, and consequently also influences the way in which EQ scalable systems should be designed.

Task Dependence. In Section 1.2.2, we have mentioned that each application virtually has its own metric to evaluate the quality of outputs [9]. Therefore, a first dependency of quality is on the considered *task*. Since a complex system normally executes more than one application at a time, this translates into a dependency over time. In other words, the amount of errors that can be introduced in a system changes dynamically according to the application being executed.

Context Dependence. Within a single application, acceptable quality is further influenced by the *usage context*. As an example, consider a wearable device providing video output through its display subsystem. The acceptable video quality will clearly change depending on the external illumination conditions, as well as on the level of user attention. To provide two representative use cases, acceptable quality constraints will relax when the user is running and using the device as a fitness tracker, and tighten when the same device is used for entertainment (video streaming).

Data Dependence. Input data also influences acceptable quality. For instance, lower computational precision can be accepted when dealing with very noisy data, since the amount of information that can be extracted from those data is limited by noise. Similarly, when performing a machine learning task such as image recognition, more precision might be needed when the subject to be recognized is positioned in a small corner of the image, compared to when it occupies the full frame. Intuitively, "errors" on single pixels have a greater impact in the former case, due to the smaller resolution of the subject. This concept was partially anticipated in Section 1.1.1, but here we analyze it from a slightly different point of view. The goal of Section 1.1.1 was to show that the *nature* of the inputs of an application (e.g. physical quantities sampled by sensors) influences its error tolerance. Here, we are stressing that the amount of information conveyed by inputs may change *over time*, and that the acceptable precision or amount of errors in computation should vary accordingly.

The time dependence of quality is well formalized in [5] by the context of *quality slack*. The author compares the gap between the actual output quality produced by a system and the minimum acceptable quality to the *timing slack* of a digital circuit, i.e. the gap between its operating period and its maximum propagation delay. As for the latter, the quality slack changes over time, due to changes in the operating conditions. Similarly to how the timing slack is affected (among others) by voltage, temperature, and input patterns, the quality slack is affected by task, context and data [2].

[2]The author of [5] also mentions *hardware resiliency* as another factor that influences the quality slack, but since this is strictly related to hardware, while our discussion is kept as general as possible, we do not consider it in this chapter.

1.3 The Benefits of Quality Scaling

In the previous part of this chapter, we have shown that error tolerance is a key feature of many computing applications. We have then discussed how output quality can be measured, and how the time-varying nature of quality affects the amount of "errors" that can be tolerated. The last step, addressed in this section, is to demonstrate how introducing "errors" can help in improving other design metrics. Indeed, it is possible to show that quality scaling can *concurrently improve* all other metrics of a digital system, i.e. increase performance, reduce cost and energy consumption [5]. We will mainly focus on energy consumption, as it is the main target of this thesis.

There are many different kinds of *deviations* from the nominal behavior of a system that can be broadly considered as "errors". The (not exhaustive) list that follows provides some examples of such deviations, explaining how they can bring benefits in other metrics. In general, all these approaches attempt to *reduce the quality slack*, introduced in Section 1.2.4, in order to let the actual output quality produced by the system adhere as much as possible to the minimum requirements.

Reducing data-path precision. Perhaps the most straight-forward way to improve performance and energy while relaxing quality consists in reducing the precision, i.e. the number of bits, of datapath circuits. Both in fixed and floating point formats, a smaller bit-width corresponds to a reduction in the switching activity and active area of the circuit, which translate into energy benefits [32–34]. Moreover, in arithmetic circuits, the timing paths with the longest propagation delays normally connect the least significant input bits with the most significant output bits. Therefore, in principle, reducing the number of bits also allows for faster computations. Some of the design techniques proposed in this thesis use precision reduction as the underlying way to perform EQ scaling. The details will be provided in Chapter 3 and Chapter 4.

Reducing design margins. Classic digital design techniques are based on so-called design margins. These are sort of safety "guardbands" that are imposed to the main constraints of a design (e.g. timing or thermal constraints), in order to account for unpredictable variability in the design process itself or in the operating conditions [35]. Imposing these margins invariably causes overheads in performance, area and energy. Considering timing constraints, for instance, a system with margins will be designed to safely operate without timing violations at a slightly faster frequency than what is actually needed in the great majority of cases [7]. This implies using bigger and more consuming logic gates, higher supply voltages, etc. Alternatively, if the designer is willing to accept some occasional timing violations, the system could be designed without margins (or even with negative margins) [5]. This allows significant power reductions and concurrent performance improvements. This is the principle used by another one of the techniques presented in Chapter 3.

Eliminating redundancy. Another source of substantial overheads in standard digital designs is the extensive use of redundancy [7]. As anticipated at the beginning of the chapter, space and time redundancy is leveraged at many levels of the computing stack, from error detection and correction in memories and data encodings, to full systems replication for fault tolerance. For an error tolerant application, some of these measures can be totally or partially avoided (e.g. correcting errors only in the most significant bits of each memory word [36]), to obtain energy reductions.

Relaxing synchronization. In complex applications that are composed of multiple concurrent tasks, synchronization often represents a performance bottleneck. Although this is mainly affecting performance, it also secondarily causes energy overheads, as parts of the system are kept idling (and consuming) while waiting for synchronization. In an error tolerant application, synchronization primitives can sometimes be eliminated or relaxed, at the cost of some dependencies violations [15].

Operating on probabilities rather than values. There are many applications which naturally call for a probabilistic treatment of data, as opposed to a deterministic one [16]. A typical example is maximum likelihood decoding in communication systems, but the same concept also applies to some digital signal processing tasks [37]. For these applications, the entire computing stack can be rebuilt in order to naturally deal with data interpreted as random variables, in a way that is much more energy-efficient than in traditional designs. This is the basic principle of *stochastic computing*, described more in depth in Section 1.5.1.

Evaluating approximated functions. Another simple yet effective way to exploit error tolerance in an application is to replace complex processing with simpler, approximate evaluations. Indeed, not all kinds of computations are equivalently efficient to implement in digital hardware (which in some cases can only performs additions and multiplications natively [32]). Nonetheless, very complex functions are often well approximated by simpler hardware-friendly computations. This kind of approximation can be applied to a large number of domains, from the computation of gradients in an iterative algorithm [20], to that of an image transformation for OLED displays [38]. The latter, specifically, is the solution adopted in our techniques proposed in Chapter 5.

The above list is far from exhaustive, and only serves to provide some motivating examples for the EQ scalable design paradigm. A more comprehensive list of techniques will be provided in Chapter 2.

1.4 A General EQ-Scalable System Architecture

Figure 1.3 shows a general architecture common to most EQ scalable systems, that embodies all the concepts analyzed in the previous part of this chapter. The architecture is composed of two main parts.

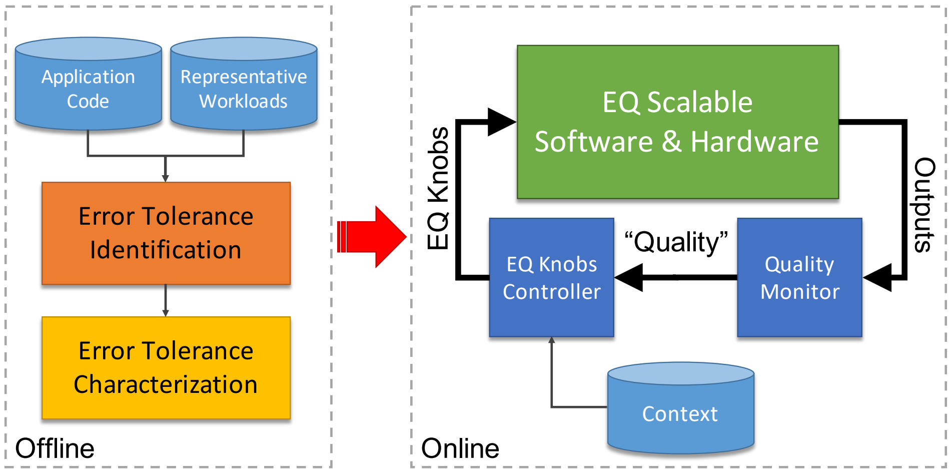


Fig. 1.3 General architecture of an Energy-Quality scalable system.

The *Offline Phase* (on the left) is devoted to the characterization of the application(s) that will be executed by the system. In particular, the error tolerance of each application must be assessed, based on the concepts introduced in Section 1.1. Furthermore, different sections of the same application may have different degrees of tolerance to quality degradations; for example, errors in flow control sections may cause dramatic consequences on the application behavior, whereas errors on data processing sections may be much more tolerable. Therefore, error resilient sections of an application are first isolated by the *Error Tolerance Identification* phase. Then, for each of these sections, the amount of tolerance to different types of "errors" (such as the examples listed in Section 1.3) is assessed in the *Error Tolerance Characterization* phase. This is normally done using software models of the different erroneous behaviors. Given the already stressed dependence of error tolerance on input data, the entire offline phase bases its characterization on a set of representative input workloads to the application, provided by the designer. Different techniques for assessing error tolerance have been provided in literature [9, 12, 13]. Some more details will be provided in Chapter 2.

The right part of Figure 1.3, labeled *Online Phase*, includes the components of the actual system at runtime. The green block, labeled *EQ Scalable Software & Hardware*, represents the part of the system that supports energy-quality scalability (in some cases, it may be the entire system). The two blue blocks, instead, are needed to dynamically adapt the quality at runtime, which is fundamental according to the analysis of Section 1.2.4.

In particular, the *Quality Monitor* block is in charge of using application outputs to estimate the current quality level. This can be done in various ways, (e.g. using an estimation model, or periodically recomputing a few outputs with maximum precision, and comparing the precise results with those generated by the system with errors) [39]. The Quality Monitor block outputs the current quality level of the application outputs and, in particular, the information on whether this level is superior to the minimum acceptable quality, characterized in the previous offline phase.

Based on this information, the *EQ Knobs Controller* blocks modifies the amount of error introduced in the system by tuning a set of knobs. In general, EQ knobs can affect both software (e.g. relaxed synchronization, computation skipping, functional approximation, etc.) and hardware (e.g. bit-width reduction, design margin and redundancy elimination, etc.). Different systems may expose a different set of knobs. In its operation, the controller also considers the current application context, as discussed in Section 1.2.4.

Altogether, the blue and green blocks form a sort of feedback control loop, aiming at minimizing energy while keeping the quality close to the minimum acceptable level.

Schemes similar to Figure 1.3 have been proposed by several authors in literature [8, 5, 12, 39], and are sometimes referred to with the name of *dynamic effort* *scaling* [12, 39]. Clearly, this architecture is highly general, and its detail may different substantially from system to system. For instance, some systems may not need a preliminary offline characterization; online quality monitoring may sometimes be unnecessary or impossible, etc. Furthermore, the implementation of each square block in the figure may change significantly from one system to the other.

Regardless of these differences, however, the main factor that enables the application of the scheme in Figure 1.3 is the availability of hardware and software that can expose energy/quality scaling knobs (i.e. the green block). This thesis mainly focuses on this part of the scheme, proposing new software and hardware techniques for achieving EQ scalability.

1.5 Paradigms for EQ Scalable Design

As anticipated at the beginning of this Chapter, several different paradigms proposed in literature can be included under the general label of Energy-Quality scalable design. All these approaches exploit application error tolerance as a knob improve energy efficiency, but they do so in considerably different ways. In this section, we provide a distinction among the three main perspectives, i.e. *approximate computing*, *stochastic computing* and *probabilistic computing*.

The reader should be warned that the nomenclature in this field reflects its relatively young stage of development, and as such it is not yet consistent among authors and communities. Therefore, while we consider some of the other terms used in the previous part of this chapter (e.g. *best-effort computing* [15] and *algorithmic noise tolerance* [14]) as elements of one of the three main families, other authors may disagree on this classification.

Finally, labels such as *dynamic effort scaling* [39] more broadly refer to the entire set of EQ scaling techniques, and can be therefore considered as synonyms of EQ scalable design.

1.5.1 Stochastic Computing

Stochastic Computing (SC) is probably the oldest form of error tolerant design, being first proposed in the 60s by two independent researchers in Europe and in the United

States [40, 41]. In SC, the semantic of binary data is completely rethought with respect to standard computing. Rather than being interpreted as a positional number, a sequence of bits in SC evaluates numerically to the *probability* of occurrence of logic value '1'. For example, a 10-bit sequence of bits (or *stream*) composed of four logic 1s and six logic 0s is interpreted as the rational number $4/10$. More formally, an n-bit string with n_1 bits assuming value 1 corresponds to the number:

$$p = \frac{n_1}{n} \tag{1.15}$$

Evidently, all numbers in this system belong to the interval [0, 1]. Moreover, there is not a unique representation of a number. For instance:

$$(0,1,0,1,0,1)_{SC} = (1,1,1,0,0,0)_{SC} = (0,0,1,1,0,1)_{SC} = 0.5_{10} \tag{1.16}$$

One of the main advantages of Stochastic Computing is the fact that some arithmetic operations between streams represented in the format of (1.15) can be implemented very efficiently in hardware [16]. For example, the multiplication among two streams is simply realized by a bit-wise AND gate, as shown in Figure 1.4.

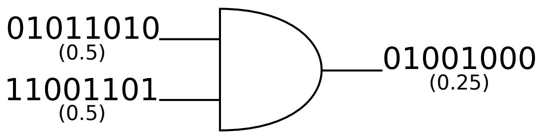


Fig. 1.4 Multiplication in stochastic computing.

However, the results of these operations are only exact under the assumption of infinitely long, uncorrelated streams. For example, it is easy to see that if the two inputs to the AND gate in Figure 1.4 are identical (e.g. $A = B = 01011010 = 0.5|_{10}$), the result of the multiplication is $0.5x0.5 = 0.5$, which is clearly wrong. In general, the longer the streams, the lower the error introduced on average by an operation; hence, a clear tradeoff exists between the performance (and energy consumption) of a SC system and its output quality.

Stochastic streams are also more tolerant to faults compared to standard binary words, in particular against soft (or transient) faults, e.g. sudden bit-flips due to radiations or other disturbances. This greater tolerance comes from the fact that

SC streams are *not positional*. Therefore, in the occurrence of a bit-flip, regardless of the position of the affected bit, the value deviation produced on the numerical interpretation of the stream is always $1/n$. In contrast, the value impact of a bit-flip in a standard positional binary representation doubles for every bit, from the Least Significant Bit to the Most Significant Bit.

The aforementioned advantages of stochastic computing come at the cost of some important drawbacks. First of all, increasing the precision of a calculation requires an *exponential* increase in the length of bit streams. This affects both the speed of calculations and the bandwidth required for memory access and communication.

Moreover, as mentioned above, SC operations produce a maximally precise result (for the considered stream length) only when their inputs are *uncorrelated*. Therefore, stochastic input streams must be generated using independent random or pseudo-random architectures (e.g. *linear feedback shift registers*). This clearly increases the hardware cost for implementing SC circuits, as every input virtually needs a separate generator (although optimization in this sense have been proposed [16]). Moreover, the correlation problem is only partially solved by using independent generators. In fact, some circuit topologies (feedback loops and reconvergent fan-outs) naturally enforce a strong correlation among streams in the internal layers of a logic network.

Correlation and performance issues have limited the application of SC in general purpose computing. Nonetheless, this paradigm has found successful applications in several domains, such as the implementation of *Artificial Neural Networks* (ANNs), control systems and image processing [16]. Recently, SC has been adopted in the decoding of *Low-Density Parity Check* (LDPC) codes. Many of these algorithms are probabilistic in nature, and as such can be naturally mapped to a SC representation. Moreover, they are also inherently error resilient, hence they can tolerate occasional errors (e.g. due to correlations), as a “price to pay” for greatly reduced implementation cost and energy consumption [42].

1.5.2 Probabilistic Computing

While stochastic computing completely revolutionizes the semantic of binary data, giving them a probabilistic interpretation, it still uses hardware in a deterministic way. For instance, the AND gate in Figure 1.4 operates exactly as in a standard logic network. *Probabilistic Computing* (PC) goes even further, and proposes to exploit hardware unreliability to explore the energy versus quality tradeoff. This paradigm originates from the increasing reliability issues in digital integrated circuits, mainly caused by feature size scaling [3, 43].

Indeed, some physical phenomena such as thermal noise and aging have an increasingly relevant impact in devices with features of few nanometers. Therefore, *Complementary Metal Oxide Semiconductor* (CMOS) transistor devices are becoming increasingly more unreliable with each new technological generation. Hardware designers and manufacturers must put great effort in building reliable circuits from unreliable physical switches. This involves the use of redundancy and design margins to make circuits tolerant to faults, thermal variations, etc.

Probabilistic computing takes the opposite road, and accepts to work with unreliable circuits. To this end, the outputs of switching devices such as transistors are associated with a probability of being correct ($p < 1$), and computer architectures are built on top of this probabilistic model [43].

The whole idea of PC makes sense whenever a small drop in p below 1 can provide a large reduction in energy consumption. Therefore, the results of a probabilistic computation can be correct with a high probability, despite being produced much more efficiently than those from a standard reliable architecture.

Transistors exhibiting this type of favorable energy versus quality tradeoff have been manufactured, and are referred to as *Probabilistic CMOS* (PCMOS). Complex circuits can be built using *Probabilistic Boolean Logic* (PBL) as an abstract model to design networks of PCMOS devices [44]. This has allowed to design probabilistic architectures for various error tolerant applications, including pattern recognition and decision systems, with promising results in terms of power efficiency [43].

1.5.3 Approximate Computing

Approximate Computing (AC) is currently the most successful design paradigm for exploring the energy versus quality tradeoff, at least in terms of number of applications in which it has been employed [8, 12, 13]. In some papers, this paradigm is also referred to as *inexact computing*.

With respect to both probabilistic and stochastic computing, AC employs a much less drastic approach to exploit error tolerance, and this is probably the main reason for its success. Differently from probabilistic computing, AC maintains the traditional deterministic logic model for transistors, and does not explicitly model the underlying unreliability of physical devices. Differently from stochastic computing, standard semantics are still (mostly) used for binary information (e.g, binary, two's complement, etc.).

The main goal of approximate computing is to build imprecise or inexact circuits using *deterministic* architectures (e.g. based on standard CMOS transistors). Clearly, statistical considerations are still extensively used, e.g. for extracting aggregate information on the errors committed, but this is done on top of a reliable hardware model. Therefore, errors in AC are *deterministic deviations* from the standard functionality of a software or hardware system, introduced either statically by design, or dynamically (i.e. in a runtime reconfigurable way). These deviations can range from functional simplifications to precision reductions, and are aimed at improving some other metric of the system (mainly energy efficiency, but often also performance). Approximate computing techniques have been proposed at all levels of abstraction, from transistor/gate-level up to software. In most of these, the voluntary introduction of errors "by design" operated by the AC paradigm has been shown to produce *disproportionate benefits* [13], i.e. statistically negligible quality reductions for significant energy savings.

As mentioned, part of the success of approximate computing is due to the fact that it maintains most of the underlying models used in standard digital systems design, rather than completely changing the perspective as in SC and PC. This allows the reuse of existing techniques and tools, modified appropriately to target the exploration of the energy versus quality tradeoff. Consequently, AC is much more accessible to software and hardware designers accustomed to standard design flows, from both research and industry [45, 22, 29].

Finally, there is a further category of hardware design approaches for error tolerant applications that, rather than modifying the *functionality* of a circuit, act on its *operating conditions*. Specifically, these solutions reduce or eliminate the standard hardware design margins, as anticipated in Section 1.3. This group of approaches mainly leverages *Voltage Over-Scaling* (VOS), i.e. the scaling of a circuit's supply voltage below the minimum value that ensures timing correctness, and is sometimes referred to as *Algorithmic Noise Tolerance* (ANT) [14]. Strictly speaking, ANT techniques do not belong to the Approximate Computing family, as they do not enforce a fully deterministic hardware behavior (in contrast, they allow unpredictable timing errors). However, at the design stage, ANT methods still use standard CMOS models and architectures, and differences with respect to a canonical design occur only at runtime. Therefore, ANT methods can still be considered somewhat similar to approximate computing approaches (surely more so than stochastic or probabilistic).

1.6 Motivation: Automating and Expanding Energy-Quality Scaling

Design techniques that explore the energy versus quality tradeoff have the potential of creating many new opportunities in today's computing. Error tolerant applications are flourishing in the IoT era, where massive quantities of data gathered from (noisy) analog sensors are processed by (statistical) algorithms for knowledge extraction, often to produce outputs that are consumed by humans. Approximate computing and other EQ scalable design paradigms have shown the potential for drastically reducing energy consumption in these applications, i.e. for overcoming the main obstacle that, in many cases, prevents their further improvement and refinement.

However, at the current state of the art, the widespread adoption of the EQ scaling paradigm is hampered by two main limitations: the scarcity of *automation* and *generality* in the proposed techniques, and the lack of methods that target other components of a system *besides processing*.

The availability of *automatic* or semi-automatic design tools has always been a key element for the industrial adoption of any computing paradigm. In contrast, most of the techniques proposed for EQ scalability are *design-specific*, and tend to lack generality. For example, as detailed in Chapter 2, tens of different works in the approximate computing community have addressed the design of functionally-approximate arithmetic operators (mainly adders and multipliers). However, only few of these works provide general methods for implementing these operators, within a standard EDA flow. Vice versa, the great majority of techniques are based on *manual design*, and the optimizations performed are specific to one particular adder or multiplier architecture.

Another aspect where lack of generality is evident is the abundance of *static* methods, where the amount of errors introduced in a system's component is *fixed at design time* [13]. These solutions ignore completely the aforementioned dependency of the acceptable errors on many time-dependent factors (application, context and input data). By doing so, their usage is confined to a *single* application, under a fixed quality constraint and under strong assumptions on the nature of input data and on the usage context. Vice versa, any general EQ scaling technique needs to allow *dynamic reconfiguration* of the amount of error committed (and corresponding energy consumption) at runtime.

Although there are some research efforts that apply EQ scaling concepts to other components of a digital system (e.g. memories [36]), the great majority of proposals in this field only target *processing elements*, ranging from software [46] to processor cores [47], to specialized hardware accelerators [48]. Considering the typical architecture of an *embedded* digital system shown in Figure 1.5, this corresponds to focusing only on the light-blue block. However, in many of such

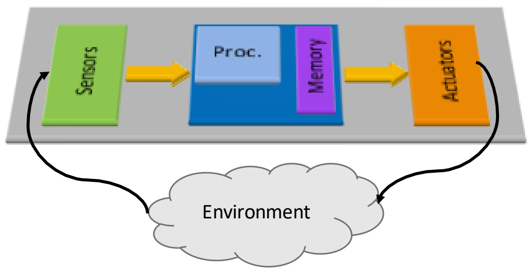
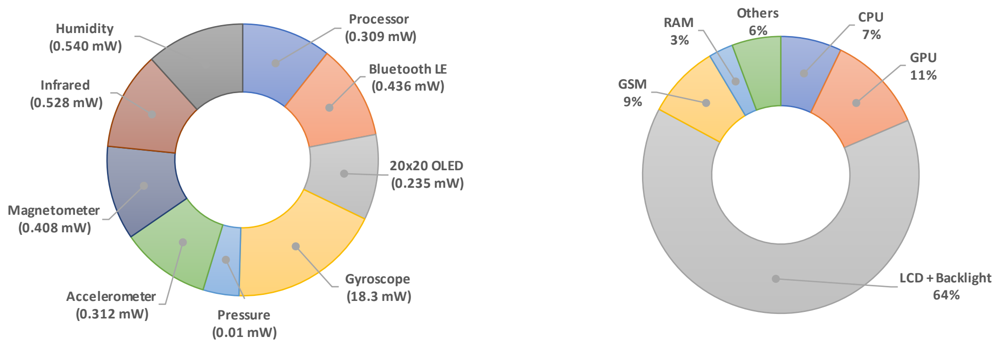


Fig. 1.5 Typical architecture of an embedded digital system.

system systems, especially those for which energy efficiency is most critical (e.g. IoT end-nodes), processing is not the main contributors to the total consumption. In a sensor node, energy may be mostly consumed by sensors and data transmission, whereas in a mobile device, the display subsystem might account for most of it. Two examples of these scenarios are reported in Figure 1.6. Therefore, many power savings opportunities are lost if EQ scaling principles are applied only to the "brain" of a system.

This thesis proposes several new techniques for EQ scalable design of digital systems, that take the two issues above into consideration. Besides processing, we propose some applications of the ideas analyzed in this chapter to *communication*

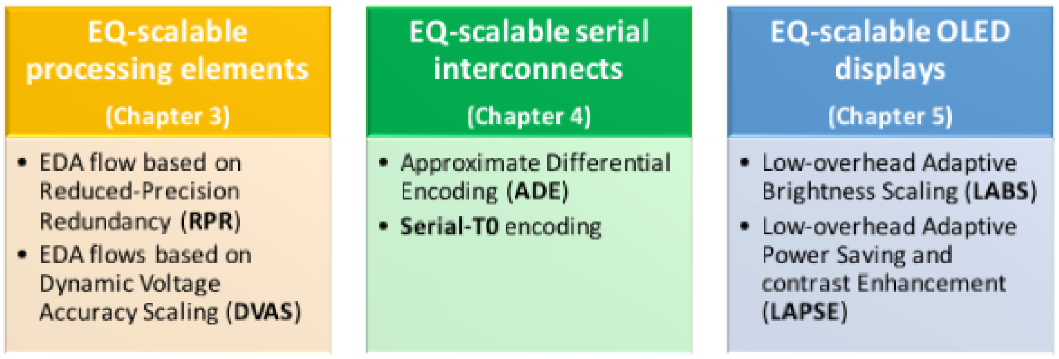


(a) Power breakdown in an IoT sensor node (logarithmically scaled chart) from [7].

(b) Power breakdown in a smartphone during video playback from [49].

Fig. 1.6 A typical power consumption breakdown in two different digital systems.

buses and to *peripherals* (namely OLED displays). In both these works, as well as in those that target computing elements, we devise our approaches with an *automation-driven* approach. To this end, we make our methods general, and applicable to the broadest possible set of targets. Moreover, we focus on ensuring compatibility with existing standards and design flows (e.g. industrial EDA flows for digital logic). Finally, most of our proposed methods allow to easily reconfigure the amount of introduced errors at runtime, by means of simple knobs. This allows to easily move the target component to a different point on the energy versus quality plane, e.g. in response to a change in external conditions or input data.



EQ-scalable processing elements (Chapter 3)

- EDA flow based on Reduced-Precision Redundancy (**RPR**)
- EDA flows based on Dynamic Voltage Accuracy Scaling (**DVAS**)

EQ-scalable serial interconnects (Chapter 4)

- Approximate Differential Encoding (**ADE**)
- **Serial-T0** encoding

EQ-scalable OLED displays (Chapter 5)

- Low-overhead Adaptive Brightness Scaling (**LABS**)
- Low-overhead Adaptive Power Saving and contrast Enhancement (**LAPSE**)

Fig. 1.7 Summary of the innovations described in this thesis.

A summary of the innovations described in this thesis is reported in Figure 1.7. Details on each of these techniques will be provided in Chapters 3-5.

Chapter 2

Related Work

The objective of this chapter is to provide an overview of the main techniques proposed in literature for EQ scalable design. The focus of this analysis will be mostly directed towards works that are related to the new techniques proposed in this thesis, in order to provide the necessary background for presenting our new proposals. Consequently, the following should not be intended as a complete survey on the topic of EQ scalable design. More comprehensive surveys taken from the literature on each sub-family of techniques will be referenced when available. The content of this chapter is an extended, updated and revised version of our previous publications found in [17, 33, 38, 50–58]

Due to the reasons mentioned in Chapter 1, we will concentrate our focus on approaches that are constructed, at least partially, upon existing design methods, such as those belonging to the approximate computing and ANT families. Moreover, we will also consider some techniques that are not explicitly labeled as approximate, but still leverage the tradeoff among error and energy, such as those for OLED displays. Conversely, we will skip a detailed analysis of stochastic computing and probabilistic computing literature, as those two paradigms are significantly different from all the proposals presented in this manuscript.

In an attempt to classify EQ scalable design techniques, one possible way consists in dividing them by abstraction level. A diagram of such classification is shown in Figure 2.1. For each level of abstraction, the figure reports some of the most popular approaches. Clearly, as any other rigid classification, this subdivision by abstraction

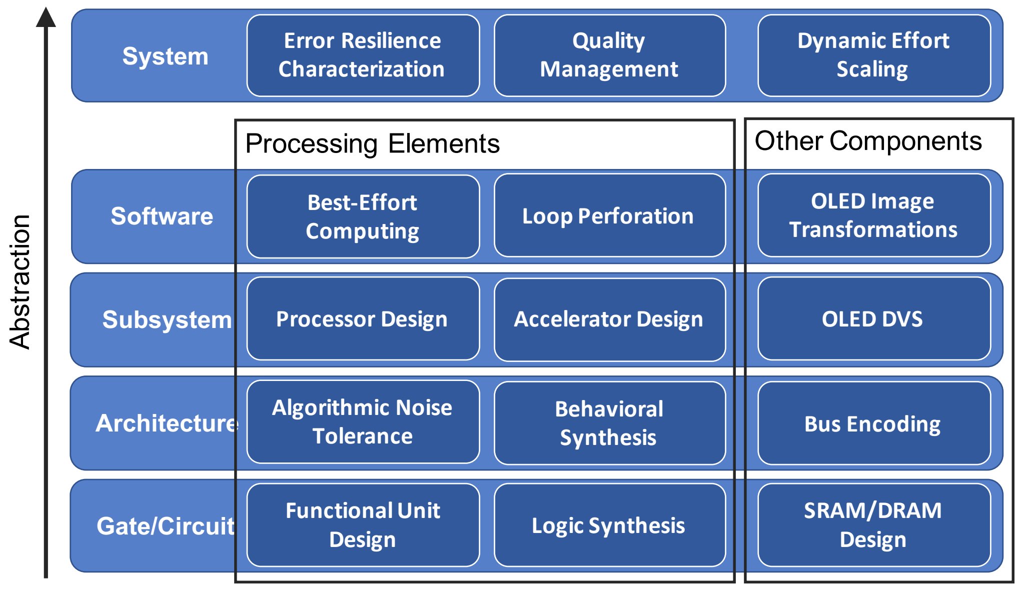


Fig. 2.1 Overview of some of the main techniques for EQ scalable design, organized by abstraction level.

is far from perfect. For example, some works for RAM memories combine both circuit and architecture-level measures, algorithmic noise tolerance can also be employed in entire processors, etc.

Figure 2.1 clearly shows that the majority of proposed approaches focuses on processing components, as anticipated in Section 1.6. In the rest of this chapter, we will first review these methods, starting bottom-up from the lowest abstraction levels. This choice is motivated by the fact that lower level techniques often constitute the elementary building blocks for higher level ones. At the end of the chapter, we will then illustrate the (few) proposals that apply the EQ scaling paradigm beyond processing.

2.1 Gate/Circuit Level Techniques

We label as "gate/circuit level techniques" those approaches that operate on the transistor or gate-level netlist of a circuit. Two main trends can be identified in literature for the implementation of EQ scalable designs at this abstraction level. A very large body of research has been devoted to the development of *Functional Units* (FUs) for digital data-paths. These works propose inexact or quality-configurable versions of the most common arithmetic data-path modules, such as adders and multipliers. Other researchers have taken a different approach, and developed tools for the automatic *logic synthesis* of inexact circuits.

2.1.1 Approximate and Quality-Configurable Functional Units

The design of approximate or quality-configurable FUs is one of the most florid sectors of the literature in this field. As mentioned, efforts are mainly directed toward the two most common datapath elements, i.e. *adders* and *multipliers*. A good survey of the earliest results in this field can be found in [19].

The term *approximate* or *inexact* FUs refers to logic netlists that implement *modified versions* of the logic function for the corresponding arithmetic operation. In these blocks, errors are inserted statically by design, and as such, these blocks do not allow a dynamic reconfiguration of the output quality. In contrast, *quality-configurable* FUs are designed to support multiple operating modes, each corresponding to a different amount of error committed, and corresponding power consumption.

Approximate Functional Units

Two main types of approximate adders can be identified in the literature. The first is based on modified single-bit *Full-Adders* (FAs), simplified at the gate or transistor level in order to produce a slightly different output logic function in exchange for power reduction. Multiple-bit adders are then constructed as a combination of accurate and approximate FAs. The second type of solution works at a coarser level, splitting the adder in sub-blocks and working on the logic function and interconnection of each block to trade-off error for power.

In the first family of approaches, one of the earliest designs is the *Lower-Part-OR Adder* (LOA) [59]. In this solution, some of the LSBs of the two input operands are combined using a simple OR gate instead of a complete FA; the remaining MSBs are summed accurately. The block diagram of a LOA is reported in Figure 2.2. A similar LSB/MSB subdivision is adopted in the five *Approximate Mirror Adders* (AMAs) proposed in [60]. However, in this case, the inexact 1-bit cells for LSBs are built using transistor-level simplifications of a mirror adder, i.e. a low-cost FA implementation.

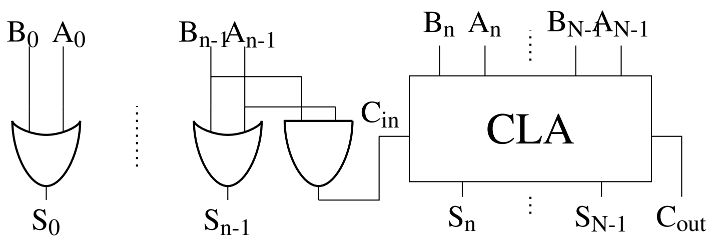


Fig. 2.2 Lower-Part-OR Adder block diagram.

The same approach is also used in [61] to generate three *Approximate XOR/XNOR-based Adders* (AXAs), but starting from a FA implementation that uses XOR and XNOR gates. Transistor-level simplifications in AMAs and AXAs directly reduce area and power in the LSBs, by reducing internal capacitances and switching activity. Moreover, they also decrease the delay of the FA, thus allowing lower supply voltage operation, for further power savings. Both goals are pursued while minimizing the truth table differences between the approximate netlists and the original FA.

The second family of adders is mostly constituted by designs that *break the carry propagation chain*. Indeed, the carry chain constitutes the critical timing path of an adder, hence breaking it enables significant delay reductions, at the expense of errors in the sum output, for some input combinations. As before, delay reduction can then be exploited to scale the supply voltage and reduce power. Moreover, the carry chain is also responsible for most switching activity in an adder, due to its frequent glitches. Therefore, reaching signal stability earlier also has a direct impact on power. One group of researchers has proposed four adders based on this principle, named *Error Tolerant Adders* (ETAs) [62–65]. After the first ETA was proposed, subsequent versions, named ETAII, III and IV, tried to improve it by reducing the impact of errors on the MSBs, breaking the chain in *non-uniform* segments or borrowing concept from carry save architectures. In [66], very similar ideas are applied to a tree adder with *Kogge-Stone* architecture, to create an approximate circuit called *Almost Correct Adder* (ACA). In [67], an improvement for ACA and ETAII is proposed. In both designs, after an interruption in the carry chain, the carry-in of the next bit is assumed to be 0 for simplicity. However, this choice biases the error towards smaller values of the sum. Instead, the authors of [67] propose to use one input bit from the previous block as estimate of the carry-in. Under the assumption of randomly distributed inputs, this estimate has a 50% probability of being correct, with a sort of *dithering* effect that improves the error distribution.

For both types of approximate adder design approaches, we have reported the first seminal works proposed in literature. Although many variants and improvements over these first implementations have been proposed more recently, they still follow the same basic principles. A recent survey on the subject can be found in [68].

Approximate multipliers are sometimes constructed using inexact adders as building blocks. In [67], both ACA and ETAII are used for the partial product summation of a *Wallace tree*-based multiplier. In [66], a similar idea is proposed, but applied also to partial product generation, in a *Booth multiplier* with Wallace tree. The authors of [69] approximate a *carry-save* array multiplier removing some of the least significant FAs involved in partial products accumulation, and replacing them with constant 0s. They name their architecture *Broken-Array Multiplier* (BAM). The approach proposed in [70] also aims at reducing the complexity of an array multiplier. In this case the multiplication operation is split in two parts, similarly to a LOA. The most significant part of the operands is processed with an accurate carry-save multiplier, while the least significant part is processed with a simpler circuit, composed of a cascade of 3-input OR gates. In [71], a multi-bit unsigned multiplier is constructed starting from an approximate elementary cell that performs 2x2-bit multiplication. It is shown that, by changing a single entry in the Karnaugh map of the 2x2 multiplier, hardware complexity can be reduced of almost 50%.

Finally, several approximate *divider* units are found in [72]. These designs are based on the non-restoring division algorithm, and approximations are introduced in the implementation of the subtractor (i.e. the basic building block of the divider). The authors propose 1-bit subtractor cell replacement, akin to those for full adders described above. They then propose several approaches for implementing the multi-bit subtractors included in divider operators using the proposed cells.

Quality-Configurable Functional Units

Quality-configurable functional units must allow different *modes of operation* in terms of energy and errors [1]. They do so by dynamically changing the complexity of

[1]Note that we refer to these modes as *quality-modes* rather than *precision-modes*, because for the majority of these methods, low-energy operation is not realized simply reducing the precision (i.e. the number of bits), but rather by implementing a (partially) *different logic function*. The quality of

the *active part* of the operator, i.e. the section of the circuit that is directly connected with inputs in a given mode. In practice, a portion of the operator is either disabled or replaced with simpler circuitry through multiplexing, when the unit is working in a low-energy, high-error mode. The power benefits of this approach are similar to the previous case of approximate FUs. First, activating simpler logic directly affects power by reducing the switching activity of the circuit. Second, it also impacts the total delay of the operator, and therefore allows further power reduction by means of supply voltage scaling. As for approximate FUs, most literature on quality-configurable operators focuses on adders and multipliers, due to their ubiquitous presence in digital circuits.

Some quality-configurable adders are based on enhancing an approximate unit with *error recovery* circuitry, dynamically activated only when zero (or low) error results are needed. This is the approach taken in [73, 23]. In particular, the authors of [73] extend the ACA architecture to generate a *variable latency speculative adder*, in which errors due to the broken carry chain are detected and a compensation factor is added to eliminate them. The solution of [23] allows to dynamically set the error rate on different output bits at runtime. Larger error configurations allow a more aggressive voltage scaling, and consequently lower power consumption. In both these designs, error compensation or recovery happens in the *next clock cycle* with respect to the first (approximate) addition. Therefore, higher quality modes incur an additional latency penalty.

Other solutions for adders are based on *reconfigurable carry-chain* segmentation [74]. This work is similar to the idea of [64] and [66] in the context of approximate units. Here, however, the connection among sub-adders is made dynamically reconfigurable by means of multiplexing. Specifically, the carry-in signal to each sub-adder can be selectively routed from the carry-out of the previous block, or from the output of a simpler *carry prediction* circuit.

Concerning multipliers, the work of [75] proposes a design in which some columns of the partial product matrix can be selectively disabled by means of signal gating. An alternative method is described in [76]. In this case, all partial products are computed, but their accumulation is implemented by means of modified adders. These blocks are able to produce an approximate sum output and a secondary

the outputs of this function can be measured in any of the ways listed in Section 1.2; precision is just one of these possible measures.

correction output. The latter can be selectively used, depending on the operating mode, to reduce the accumulation error, at the cost of additional energy consumption. Differently from the ones described previously, this last design does not support a maximum precision mode (i.e. a mode that produces the exact same output function of a standard operator of the same bit-width).

Dynamic Accuracy Scaling and its Variants

Dynamic Accuracy Scaling (DAS) and its variants are among the simplest yet most effective ways to implement quality-configurable functional units [34, 48]. Since these techniques are at the basis of some of our new proposals discussed in Chapter 3, we analyze them in a separate section, although they conceptually belong to the previous category.

The initial observation that spurs the development of DAS techniques is that using functional modifications to implement quality-configurable hardware operators has several drawbacks. First, most of the methods described in the previous section lack generality, being applicable only to a single functional unit architecture. For example, the works of [75] and [76] can only be applied to array multipliers. Second, the additional circuitry required to implement multiple quality-modes often entails significant overheads. Indeed, while those functional units consume less power than their "standard" counterparts when working at low-quality, they typically incur large power overheads in the highest quality mode. In some cases, e.g. [73] and [23], also their latency increases significantly. Last, most architectural solutions for quality-configurable operators design only support few operating modes. For instance, several works only support one inexact and one correct mode [73].

DAS and its further developments solve these issues by implementing quality-configurable functional units that include minimal architecture modifications. Instead, they propose to realize multiple quality-modes simply by reducing the input bit-width of the operator, i.e. its *precision* [2]. In practice, this is achieved *gating* some of the input LSBs to zero.

[2]Indeed, in the case of DAS techniques, quality modes do correspond to precision modes. The term "accuracy" used in the acronym for these methods is slightly improper, as accuracy normally refers to *models*, rather than computations, and indicates a systematic deviation from the correct output.

The standard DAS approach obtains power savings when precision is reduced just thanks to the deactivation of a part of the circuit, which affects dynamic power consumption [34]. *Dynamic Voltage and Accuracy Scaling* (DVAS), a first improvement of DAS, increases these savings by also performing supply voltage scaling. This is made possible by the fact that (in theory) the longest timing paths in arithmetic units are those connecting input LSBs to output MSBs. When some LSBs are gated, these critical paths become *false* or *deactivated*, i.e. a signal transition will never propagate through them due to the fixed zero at the input. Therefore, the worst case delay of the FU decreases, allowing to scale the supply voltage below the minimum value that ensured timing compliance at maximum precision, while maintaining the same operating frequency. This provides much more relevant dynamic and leakage power reductions compared to DAS.

Some of the advantages of these methods compared to the ones mentioned above are immediately clear. First, bit-width gating can be realized with a 1-bit granularity, hence allowing for a large number of different quality modes. For example, a 32-bit unit could be used in all bit-width modes down to a minimum of 4-bit, for a total of 28 different modes. Second, this solution has practically zero overheads when working at maximum precision, compared to a standard design of the same unit. Third, input gating can be applied, in principle, to any arithmetic operator, and its benefits are maintained as long as the mentioned relation between timing paths and input/output bits holds. This makes DAS and DVAS much more general than previous architecture-based solutions.

Furthermore, an interesting study performed in [77] has demonstrated that, despite its simplicity, bit-width reduction is almost always superior to any other approximate or quality-configurable FU implementation in terms of energy versus quality tradeoff. Conceptually, this happens because other approaches, although introducing simplifications in the netlist used at low-quality (with respect to a standard design), still invest a significant amount of power to compute erroneous information, whose usefulness is limited. Conversely, DAS and DVAS directly *avoid* computing the outputs related to some LSBs, thus saving substantially more energy. In other words, the additional energy effort required for computing approximate results, as opposed to skipping part of the computation, is disproportionate.

Dynamic Voltage Accuracy and Frequency Scaling (DVAFS) [48] is a further improvement of DVAS. It comes from the observation that input gating in DAS and DVAS leaves a part of the operator unused. While this part does not switch and hence only contributes to leakage consumption, even further benefits could be obtained by reusing it for other computations. In practice, this is achieved by means of *sub-word parallel* operations. For example, when a 32-bit multiplier is working at a reduced precision of 16-bit, the LSB-part of the circuit can be used to perform *another* 16-bit multiplication in parallel, by means of some additional gating logic to decouple the two halves. Doing so, the switching activity of the operator goes back to being approximately equal to the maximum precision mode. However, if some minor architecture modifications are performed in order to allow separation among the two operations, the critical delay of the functional unit is still reduced (at least in principle), thus allowing to scale the supply voltage as in DVAS. Moreover, thanks to sub-word parallelism, the clock frequency can be reduced of a factor of 2 while maintaining the original throughput, for additional power savings. Although very effective, DVAFS is a less flexible approach compared to DVAS. In fact, it does require some manual architectural modifications, and it can only be applied to circuits whose netlist can be modified to support subword-parallel operation. A full chip including DVAFS-based operators has been published in [78].

The advantages of DAS and its variants make them very promising general techniques for realizing quality-configurable FUs. However, these methods have some practical limitations when they are adopted in the context of a standard EDA synthesis flow. Problems are related in particular to the assumption that a circuit delay decreases proportionally to the number of gated input LSBs. This issue is the main motivation for two of the works presented in Chapter 3, and will be further discussed at the beginning of that chapter.

2.1.2 Approximate Logic Synthesis

Approximate Logic Synthesis (ALS) techniques generalize the concepts introduced for inexact functional unit, and apply them to *any* gate-level netlist in an automatic way. These approaches are fundamental to automate approximate design, since larger and more complex circuits, as well as those that are less “structured” compared to an arithmetic FU, cannot be manually turned into inexact versions just as easily as adders and multipliers.

Similarly to FUs, the algorithms described in this section can be divided into those that produce statically inexact circuits [22, 26, 45, 79], i.e. netlists in which the type and amount of errors are fixed at design time, and those that produce quality-configurable circuits [79].

An interesting early work on this subject can be found in [22], where a method for ALS is proposed targeting two-level combinational circuits expressed in *Sum Of Products* (SOP) form. The main aim of this work is area and power reduction, so the proposed algorithm tries to minimize the total number of *literals* in the SOP expression. It does so by complementing the maximum number of minterms from the off-set to the on-set of the expression, without violating a quality constraint provided in terms of maximum error rate.

The same authors extended their work to the much more common case of multi-level circuits in [26]. Rather than reasoning on minterms, in this case simplifications on the netlist are performed with methods inspired by redundancy elimination optimizations. In particular, they consider the effects on output quality produced by stuck-at faults at different locations in the netlist. When these effects are acceptable, the netlist is simplified as if the fault location was redundant.

The authors of [80] propose *Probabilistic Pruning* (PP) and *Probabilistic Logic Minimization* (PLM), two general techniques for the synthesis of statically approximate circuits. PLM is a generalization of the minterm-flipping approach proposed in [22], which allows to also account for the case of non-uniform minterm probabilities. This is by far the most common occurrence in real applications, due both to the distribution of inputs and to the internal topology of the circuit. Probabilistic pruning, instead, is a higher-level approach, aimed at large datapath blocks. It consists of an iterative identification and elimination of netlist components, driven by a greedy heuristic algorithm, and constrained by an error function, which can be expressed in several different forms. Circuit components to be pruned are selected based on a merit function that combines their *activation* (i.e. the probability of a transition at their output), and their *significance* (i.e. their impact on the output error). Both activation and significance can be computed either using mathematical models or by means of simulations.

One of the most effective ALS tools for combinational logic is the one proposed in [45], under the name of *Systematic methodology for Automatic Logic Synthesis of Approximate circuits* (SALSA). This work introduces several novelties with respect

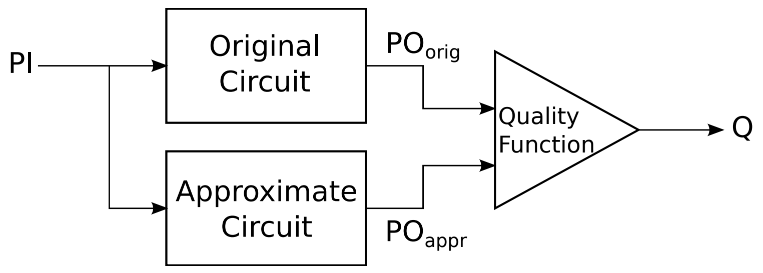


Fig. 2.3 Quality Constraint Circuit (QCC) in SALSA.

to the previous ones. Most importantly, it allows to fully leverage the optimization capabilities of commercial synthesis tools for standard circuits, by transforming ALS into a traditional synthesis problem. Moreover, it also unlinks the synthesis procedure from a specific quality metric, allowing the designer to set their preferred metric as input, thus greatly enhancing the flexibility of the tool.

The algorithm proposed in SALSA works by constructing so called *Quality Constraint Circuit* (QCC), as shown in Figure 2.3. The triangular block shown on the right of the QCC diagram, contains a logic-level implementation of the user-defined quality function. This block receives as inputs the outputs of the original (accurate) circuit and of the approximate circuit being synthesized. It then internally computes some form of error function, and outputs a single-bit signal Q, which is at logic 1 if and only if the quality constraint is respected. For example, Q can be set to 1 whenever the ED (see Section 1.2) between the output of the accurate and approximate circuits is smaller than a threshold. Notice that both the original and approximate netlists are connected to the same primary inputs (PIs), so that this kind of quality evaluation is meaningful. The only constraint on the allowed quality functions is that they must be expressible by synthesizable logic.

Given the structure of Figure 2.3, SALSA then considers all output bits of the approximate circuit $PO_{appr,i}$, and computes the primary input values for which Q is independent on each of these bits, i.e. the *Observability Don't Cares* (ODCs) of Q with respect to $PO_{appr,i}$. Those are the input combinations for which changing the value of $PO_{appr,i}$ does not violate the quality constraint. Therefore, they can be used to simplify the approximate circuit, reducing area, power, and delay. Both the individuation of ODCs and the following netlist simplifications are performed with commercial logic synthesis tools. The process is repeated on all bits of PO_{appr}, progressively updating the approximate circuit in the QCC. Several optimizations are proposed in the paper to reduce the computational complexity of the algorithm.

In the work of [79], the SALSA approach is extended to consider also sequential circuits. The resulting tool is called *Automatic methodology for Sequential Logic ApproximatioN* (ASLAN). In ASLAN, a *Sequential Quality Constraint Circuit* (SQCC) is constructed, conceptually similar to the one in Figure 2.3. However, in this case, the quality function also receives as inputs the state registers of both accurate and approximate circuits, and contains an internal state machine to determine *when* quality must be evaluated. Rather than don't care propagation methods, sequential model checking (implemented with commercial formal verification tools) is then used to simplify the approximate circuit.

The *Substitute-and-Simplify* (SASIMI) technique, proposed in [81], is one of the most notable approaches for synthesizing both statically approximate and quality-configurable circuits. This solution identifies pairs of internal netlist signals which have similar logic functions. It then replaces one of the signals, called *Target Signal* (TS) with the other, labeled *Substitute Signal* (SS). This substitution allows to then simplify the circuit, by eliminating the cone of logic needed to generate TS, and possibly relaxing the timing constraints of the transitive fan-out of both TS and SS. The process is repeated iteratively, as long as the input quality constraint is respected, using merit functions to select the optimal SS and TS among all circuit nets.

When targeting a quality-configurable design, the cone generating TS is just disabled rather than eliminated, so that accurate computation can be performed when needed, at the cost of one additional clock cycle. In this case, additional logic is used to selectively reconfigure the netlist depending on the active quality mode. More than two quality modes can also be obtained in a similar way.

2.2 Architecture-Level Techniques

Architecture-level techniques for EQ scalable design work at a higher abstraction level compared to the previously described gate/circuit level approaches. They target relatively larger processing blocks, ranging from simple *Multiple and Accumulate* (MAC) units to full co-processors, and they perform their optimizations without directly modifying the individual logic gates that compose the circuits.

A first-order classification of these methods separates those that achieve energy savings by modifying the functionality of a block from those that pursue the same goal leveraging its operating conditions. The first family includes approximate behavioral synthesis algorithms and tools, while the second is composed by all those works that leverage *Voltage Over-Scaling* (VOS) for approximation.

Clearly, there is not a well defined distinction between this family of techniques and the previously mentioned gate-level ones. For example, some papers on VOS perform experiments single functional units, and could therefore be considered at the same level as the solutions of Section 2.1.1.

2.2.1 Approximate Behavioral Synthesis

Approximate behavioral synthesis methods address the automated synthesis of inexact circuits, starting from a functional specification of their behavior (typically graph-based) rather than from a structural netlist of logic gates. These approaches are sometimes also referred to as *approximate architectural synthesis* or more generally *approximate high-level synthesis*. With respect to standard high-level synthesis algorithms, approximate versions aim at simplifying the functionality of a design by introducing errors or precision reductions, mostly statically at synthesis time, under a minimum quality constraint. However, simplifications do not involve modifications of the internal netlist of a block at gate level, as in approximate logic synthesis. Instead, FUs are considered as atomic building blocks, and approximations are realized by their substitution, deletion, transformation, etc.

An early effort in this field is documented in [29], where the authors present a tool named *Automated Behavioral Approximate CircUit Synthesis* (ABACUS). ABACUS produces multiple inexact versions of a design, originally specified in a behavioral *Hardware Description Language* (HDL). The specification is first translated into an *Abstract Syntax Tree* (AST) model; the latter is then transformed by a series of *operators*, each of which introduces some approximations in the output, while preserving syntactic correctness. The proposed operators include precision reduction (i.e. LSB-truncation), variable to constant substitution, loop unrolling, etc. Additionally, ABACUS also permits the usage of approximate FU models (such as those presented in Section 2.1.1). In this case, an additional operator leverages inexact FU implementations as atomic replacements for their accurate counterparts.

The transformed ASTs are then converted back to HDL and processed with a standard synthesis and simulation flow to evaluate their cost (in terms of area and power) and their output quality. Therefore, ABACUS operates only by preliminary AST optimizations, in the so-called front-end part of a high-level synthesis flow, rather than on the actual synthesis steps (i.e. scheduling, binding, etc.).

To avoid an exponential number of synthesis and simulations, ABACUS does not apply all its operands to all feasible locations in the AST, in an exhaustive way. Instead, the tool generates a number of approximated ASTs choosing operands and locations at random, then greedily selecting the *best* solution according to a cost function. This process is repeated for a fixed number of iterations, where the input to each iteration is the best AST selected in the previous step. The cost function is expressed as a weighted combination of cost (power and area) and quality. A hard threshold on the allowed error is also set, so that solutions that affect quality too much are not synthesized.

The work of [82], differently from ABACUS, considers approximations within the scheduling and binding phases of high-level synthesis. The proposed algorithms take as input a *Data Flow Graph* (DFG) composed of additions and multiplications and concurrently perform its scheduling and binding, while optimizing the precision of each operation under an output error constraint. In this work, approximations are introduced as bit-width reductions. The authors use a simple (and approximate) analytical model of the DFG output quality, that does not require simulations to be evaluated. This model greatly improves the performance of the following algorithms, since the latter must consider quality during scheduling and binding optimizations, and time-consuming simulations would dominate their execution time. Then, they propose two methods to perform approximation-aware scheduling and binding: an exact solution based on *Integer Linear Programming* (ILP) and a heuristic alternative based on iterative *list scheduling*. Both approaches also consider the asymmetric compatibility of approximate operations, i.e. the fact that a low precision operation can be mapped to a high-precision FU (for improving sharing), while the opposite is not possible.

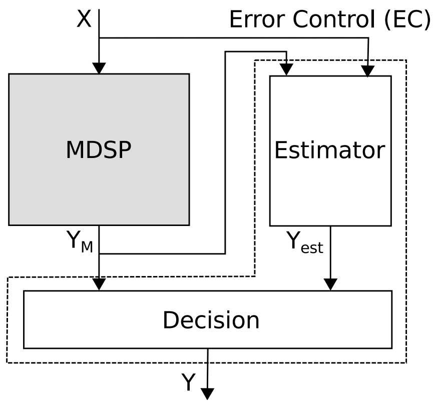


Fig. 2.4 General scheme of an ANT architecture.

2.2.2 Voltage Over-Scaling

The second group of architectural solutions for EQ scalable design is composed by all those works that leverage VOS for approximations [24]. In these works, the supply voltage V_{dd} of a system is lowered below the critical value $V_{dd,crit}$. The latter is defined, for a synchronous digital circuit, as the lowest voltage that guarantees timing correctness even in worst-case conditions of temperature, process variability, etc. Despite using $V_{dd} < V_{dd,crit}$, the clock frequency f_{clk} of a VOS system is kept at the nominal value, to maintain the original system latency and throughput.

VOS yields significant dynamic and static power reductions due to their strong dependence on V_{dd}, but also also induces *timing errors* due to setup constraints violations. The rate and impact of timing errors depend both on the circuit topology and on the characteristics of input data [17]. However, as long as these violations only occur occasionally (i.e. for few input combinations), so that the circuit can still output correct results most of the times, VOS operation can yield a very advantageous energy versus quality tradeoff.

One of the first proposals to leverage VOS for designing energy-quality scalable architectures is found in [14]. In this work, an arithmetic or DSP architecture operating in VOS conditions is combined with an additional *Error Control* (EC) block, with the purpose of detecting and *mitigating* the effect of timing errors. This general scheme is shown in Figure 2.4, and takes the name of *Algorithmic Noise Tolerance* (ANT), or sometimes more generally of *soft-DSP*. In this figure, the gray box represents the main functional hardware block, e.g. an arithmetic unit or a digital filter, and is referred to as *Main DSP* (MDSP) block. The EC block is delimited by the dashed box.

The purpose of the EC block is to detect timing violations, and limit their effect by providing an *approximation* of the correct results. Thus, this block does not fully *correct* errors, as in a fault tolerant system, and its design can be consequently made simpler, reducing the area and power overheads. This is fundamental to ensure that the entire ANT system in VOS consumes less power than the sole MDSP at nominal V_{dd}. At the same time, the simplicity of the EC block also allows to make this component timing compliant even at reduced V_{dd}, so that it is not affected by timing errors itself, thus loosing its purpose.

More specifically, the EC block is composed of two main elements, an *estimator* and a *decision* block. The former produces the output approximation using (in the most general case) information from the MDSP input and output signals. The decision block is then in charge of selecting among the MDSP and estimator outputs, based on the possible occurrence of a timing error. MDSP and estimator outputs are latched before being fed to the decision block. This prevents the decision logic from ending up in the critical timing path, thus becoming prone to timing errors, and rendering the entire architecture useless.

ANT architectures are based on two assumptions on the nature of VOS-induced timing errors. First, as anticipated, these violations must be *rare*, so that the MDSP produces a correct output in most cases, and the overall quality of the system remains comparable to that of the MDSP at nominal V_{dd}. The validity of this assumption depends on a number of factors, including the selected VOS voltage, the sequence of input patterns fed to the system, and the distribution of timing *slacks* in the MDSP netlist. As explained in the following, some works propose circuit-level methods to enforce this assumption [83, 84]. Second, the errors produced by timing violations must be *large* in magnitude, to be easily detectable by the decision block. This is theoretically true, at least for arithmetic and DSP systems, due to the fact that critical timing paths are those related to the computation of output MSBs (see Section 2.1.1).

The assumption of large error magnitude allows to use the *absolute value difference* between MDSP and estimator outputs as a discriminating factor for the occurrence of timing errors. Indeed, as the estimator provides an approximation of the correct result, if its output is significantly different from that produced by the MDSP at any given time, then it is highly probable that the latter is subject to

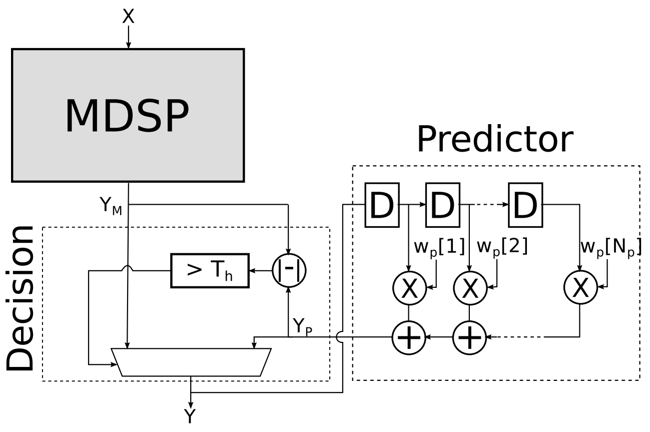


Fig. 2.5 Prediction-based ANT architecture.

a VOS-induced error, and vice versa. This intuition is formalized by the following equation, which is used as decision scheme in ANT:

$$Y[n] = \begin{cases} Y_M[n], & \text{if } |Y_M[n] - Y_{est}[n]| \leq T_h \\ Y_{est}[n], & \text{if } |Y_M[n] - Y_{est}[n]| > T_h \end{cases} \tag{2.1}$$

The variables used in this equation refer to the scheme of Figure 2.4, and the symbol $[n]$ is used to indicate the value of a variable at clock cycle n.

Two main types of ANT architectures have been proposed, under the names of *prediction-based* ANT [14, 85] and *Reduced-Precision Redundancy* (RPR) ANT [86, 28]. The differences between the two mainly reside in the implementation of the estimator block.

Prediction-based ANT

In prediction-based ANT [14, 85] , a *forward predictor* is used as estimator. In its simplest form, this block approximates the correct output as a linear combination of the recent output values history. Thus, the estimator output is computed as:

$$Y_p[n] = \sum_{k=1}^{N_p} w_p[k] Y[n-k] \tag{2.2}$$

where regression weights $w_p[k]$ are chosen to minimize the *MSE* of the predictor in absence of VOS-induced timing violations, and N_p is the width of the selected window of previous outputs. The size of this window creates a tradeoff between the complexity of the estimator block and the prediction accuracy.

The hardware implementation of prediction-based ANT is depicted in Figure 2.5, where boxes labeled with D are *delay* elements, i.e. flip-flops. The figure also shows one possible implementation of the decision scheme of (2.1).

One critical issue with this type of estimator is the choice of the decision threshold T_h. In [85] the authors propose to use a multiple of the standard deviation of the prediction error. However, this method does not ensure that the MDSP output is *always* selected in absence of timing errors. For example, an abrupt variation in the MDSP output may cause a large error in the linear predictor. Consequently, depending on T_h, the decision block may assume that an error has occurred, when in reality it has not. In general, prediction-based ANT works well when the outputs of the MDSP are strongly *temporally correlated*, i.e. when subsequent values of Y_M are correlated with each other. Another limitation of this architecture is that the predictor output is biased by previous errors occurring within the considered window. In particular, if *error bursts* occur at the MDSP outputs, the detection and output estimation capabilities of this scheme worsen. For this reason, aggressive V_{dd} scaling is not possible in prediction-based ANT, as this would cause accumulation of errors in the prediction window.

Reduced-Precision Redundancy ANT

In RPR ANT [86, 28], the EC block is constructed as a lower precision replica of the MDSP, with its inputs connected to the MSBs of the MDSP inputs. A scheme of this architecture is shown in Figure 2.6. The usual decision scheme of (2.1) is used also in RPR, since replica outputs have a small magnitude error with respect to the correct results (they differ only for some LSBs), and can therefore be used to identify large magnitude errors caused by timing violations. Thanks to reduced-precision, the replica hardware is smaller and faster than the MDSP, and therefore can be designed to be timing compliant in VOS. As for the window length in prediction-based ANT, the input bit-width of the replica (B) can be used to explore the tradeoff between approximation error and EC block overheads.

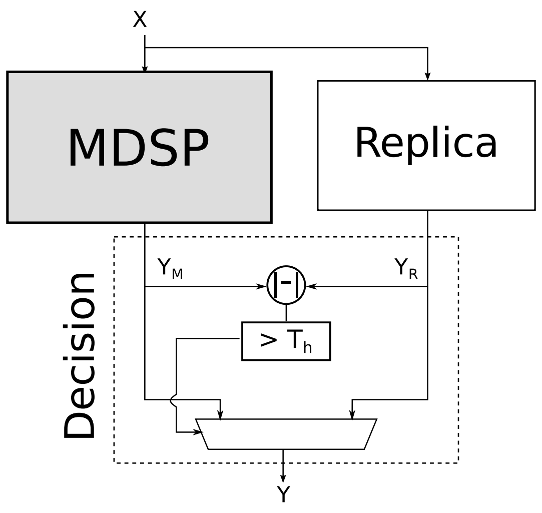


Fig. 2.6 Reduced-Precision Redundancy (RPR) ANT architecture.

Differently from the case of prediction-based ANT, the decision threshold T_h in RPR architectures can be computed in a way that ensures that MDSP outputs are always selected in absence of timing errors. Specifically, T_h can be chosen as:

$$T_h = \max_{\forall input} ||Y_{M,0}[n] - Y_R[n]|| \tag{2.3}$$

where $Y_{M,0}[n]$ and $Y_R[n]$ are the error-free MDSP outputs and the Replica outputs at time n respectively. Theoretically, evaluating this equation exactly requires to consider the entire output images of $Y_{M,0}$ and Y_R. In the case of complex MDSPs, for which such images are not easily computed, T_h can be approximated by simulation or statistical measures.

One advantage of RPR is the higher flexibility compared to prediction-based schemes. Indeed, approximate outputs in RPR are computed only based on the value of inputs at the same clock cycle. Therefore, this architecture is not negatively influenced by the lack of correlation among subsequent MDSP results [3]. Similarly, the error mitigation in RPR ANT is not influenced by recent errors, except for the cases of MDSPs (and replica) that include internal feedback networks. On the other hand, RPR EC blocks tend to have larger overheads compared to precision-based solutions, especially for complex MDSPs.

[3]In practice, different input sequences do cause different timing error rates and magnitudes, thus resulting in a different output quality. However, they do not have a direct impact on the error mitigation capabilities of the EC block.

Other forms of ANT

In hybrid ANT [28], the EC block is designed combining the ideas of prediction based ANT and RPR. In practice, the estimator includes both a linear predictor and a reduced-precision replica, and a *Finite State Machine* (FSM) is used to select between the two depending on an *operating mode*. By default, the output approximation is obtained with the linear predictor, while the replica is turned off by means of power/clock gating, thus eliminating most of its overheads. When a timing error is detected, the FSM then turns on the replica, and shuts down the predictor. In this way, prediction divergence due to accumulation of errors is avoided. The replica is kept active until no errors are detected for a given number of clock cycles (set as a parameter of the system), then the default mode is restored. With this dual operation, hybrid ANT outperforms both prediction-based an RPR solutions for many designs, at the cost of a larger area overhead [28].

More recently, ANT solutions have been also considered for near-threshold and sub-threshold operation in [27], targeting *Minimum Energy Point* (MEP) operating conditions. Finally, the work of [87] has proposed *Embedded-ANT* (E-ANT). The key idea of this approach is to reuse part of the MDSP output to build the EC block, in order to reduce the (normally very large) overheads of conventional RPR ANT. This goal is achieved by means of *datapath decomposition*, a technique that converts the overall MDSP output function $Y_M = f(x)$ into the combination of two functions that depend only on MSBs and LSBs respectively, i.e. $Y_M = g(f_{LSB}(x_{LSB}), f_{MSB}(x_{MSB}))$. As long as the critical path of the hardware that implements f_{MSB} is shorter than the one of the new global function g, the former can be used as output approximation, similarly to the replica output in RPR. While obtaining a significant reduction of overheads compared to RPR, E-ANT is a less general approach. Indeed, exact datapath decomposition can only be performed for simple MDSPs. While the authors of [87] propose different ways to achieve an *approximate* datapath decomposition, including Taylor expansion and piecewise linear approximation, those can be applied only if some pre-conditions on f hold (e.g. differentiability). Moreover, using these methods implies that the MDSP will never produce fully correct outputs, even in absence of timing errors.

Voltage Over-Scaling Beyond ANT

The so-called *Significance Driven Design* (SDD) family of approaches [88–91] exploits VOS operation using a different perspective. Rather than reducing timing errors induced by VOS, SDD approaches try to confine their occurrence to low-significance computations, i.e. those that do not impact drastically the output quality. This is achieved combining algorithm-level and hardware architecture-level measures. For example [88] applies SDD to a *Discrete Cosine Transform* (DCT) accelerator. In this work, some of the DCT function coefficients are slightly altered (algorithm-level modification), in order to improve hardware sharing (architecture-level modification). These modifications allow to compute the most significant DCT outputs (from the point of view of the visual characteristic of the output image) with a smaller delay. Consequently, when the hardware is operated in VOS, timing errors will only affect less significant components.

As previously mentioned, VOS-based approaches require that timing errors are *rare* events. In practice, the occurrence of such errors is determined by the activation of long combinational paths in the hardware, caused by pairs of input vectors. Consequently, the distribution of path lengths in the circuit clearly impacts the frequency and effect of timing errors errors. Different works have considered the problem of redistributing the timing slack among paths, in order to allow a more *graceful* degradation of the output quality of a system under VOS [83, 84]. Specifically, the authors of [83] propose a gate-sizing algorithm aimed at changing the path distribution to enable a more aggressive over-scaling. In [84], some general architectural modifications for arithmetic circuits are proposed, that produce the same kind of path distribution reshaping. The latter work focuses on the so-called *meta-functions*, i.e. operations that are commonly found in many different error tolerant applications. In particular, two techniques for building operators that can support aggressive VOS are proposed, called *dynamic segmentation* and *delay budgeting*. The former corresponds, in practice, to *carry chain segmentation*, described in Section 2.1.1. Delay budgeting, instead, is used when multiple operators are chained together, and is based on inserting latches in order to obtain an optimal share among the operators of the delay budget available in each clock cycle.

For completeness, notice that VOS is also frequently leveraged for power reduction outside the context of EQ scalable design. In [92–94], among many others, the authors proposed techniques for *Better than Worst-Case* (BWC) design. In these paradigms, design margins are reduced below the limits that ensure wort-case timing correctness, and VOS is by far the most common technique employed to do so. However, rather than simply being *accepted*, timing errors are first *detected* by means of special devices (e.g. Razor or Canary flip-flops), and then *corrected*, normally allocating an additional clock cycles to let the longest combinational paths stabilize (e.g. by means o clock gating or pipeline stalling).

2.3 Subsystem-Level Techniques

All the EQ scalable design techniques proposed in this thesis for processing elements work at the circuit or architecture abstraction levels, detailed in Section 2.1 and Section 2.2 respectively. Therefore, the state of the art for higher levels of abstraction, from subsystem-level onward, will be treated in a less detailed way. Nonetheless, in this and in the next sections, we will try to provide a general overview on the full spectrum of existing techniques.

EQ scalable design techniques that consider entire hardware subsystems as their target can be broadly divided into two groups: those that focus on *General Purpose* (GP) processors, and those that concentrate on domain-specific or application-specific hardware accelerators.

2.3.1 Processor Design

One of the most interesting proposals for the design of an EQ scalable GP processor is the so called *Error Resilient System Architecture* (ERSA) [95]. ERSA is a multi-core platform composed of some *Super Reliable Cores* (SRCs) and some *Reduced Reliability Cores* (RRCs). SRCs are made reliable by appropriately setting margins on the operating conditions (voltage, clock frequency, etc.) and possibly using fault tolerance mechanisms. RRCs, instead, are subject to errors of various nature such as aging, temperature, external radiations, etc. Evidently, the lack of error correction in RRCs allows them to have significantly lower cost and power consumption.

The execution of an application on ERSA is orchestrated by SRCs, that execute the main control flow and dynamically offload error-resilient tasks to RRCs. SRCs are also in charge of performing some low-cost checks on the execution of RRCs, including liveness checks via watchdog timers, and minimum sanity checks on results (e.g. whether an output belongs to the expected range, whether successive iterations converge, etc.). Whenever a RRC hangs or crashes, the offloaded task is restarted, possibly scheduling it to a different core. This architecture protects the system against catastrophic errors on control and other non-tolerant computations, while exploiting error resilience on those computations that can be approximated.

In [96] a different error-resilient processor architecture is proposed, which attempts to reduce the overheads of ERSA removing the need of an entire reliable core. Instead, the authors analyze the minimum micro-architectural checks required to avoid catastrophic errors. They propose to implement with reliable hardware only the blocks required to perform these controls (e.g. instruction sequencer, memory fence unit, etc.) while the most part of the core remains error-prone.

The work of [47] describes a further programmable vector processor for error resilient applications denominated Quora. The main difference with respect to ERSA and similar architectures is that Quora offers the possibility of setting the desired level of quality via a *Quality-Programmable Instruction Set Architecture* (QP-ISA). The software designer can use the QP-ISA to constrain the quality of each individual data instruction, for example in terms of the maximum error magnitude that it can produce. This enables higher flexibility with respect to ERSA, in which tasks are executed either reliably or completely unreliably. Quora is designed as a vector processor, based on the *Single Instruction Multiple Data* (SIMD) paradigm. This choice is motivated by two reasons. On the one hand, many error resilient applications (e.g. from the RMS domain) are dominated by vector and matrix operations. On the other hand, a vector architecture reduces the ratio between control hardware, which is inherently error-intolerant, and data hardware, in which resilience can be exploited for power saving. Similarly to ERSA, an accurate *Processing Element* (PE) is used for control instructions. Quality programmable PEs, of two different types, are instead used for data instructions. In the latter, different quality and power configurations are achieved via reduction of operands bit-widths combined with power and clock gating.

2.3.2 Accelerator Design

The application-dependent nature of error tolerance makes the EQ scalable design paradigm perfectly suited to the design of specialized hardware accelerators. Many researchers have used EQ scalable design techniques for accelerating error resilient tasks. In particular, the domain that has received the most attention from this point of view is certainly machine learning. Notice that, in the case of hardware accelerators, there is not a clear boundary between architecture-level techniques (analyzed in Section 2.2) and subsystem-level ones, considered here. In general, the works analyzed in this section refer to bigger designs, that can independently execute large-scale tasks (e.g. a full neural network inference), typically offloaded from a CPU.

The authors of [97] propose a framework for accelerating general purpose programs using *Neural Processing Units* (NPUs). To this end, they introduce an algorithmic technique named *parrot transformation*, which automatically converts error tolerant code sections into a *Neural Network* (NN) representation, specifically a *Multi-Layer Perceptron* (MLP). To achieve this transformation, the inputs and outputs of the target section are profiled, and used to train several MLP models; the network topology that produces the lowest error is selected. Once the topology and network parameters are fixed, the original code is automatically modified to replace calls to the target function to corresponding invocations of the MLP model. At runtime, the network is accelerated on a digital NPU, implemented as a tightly-coupled accelerator. Thanks to the higher specialization and smaller control overhead of the NPU compared to a CPU, performance and energy efficiency gains are simultaneously obtained.

The work in [98] introduces a design methodology called AxNN to run approximate versions of NN models onto a specialized hardware accelerator. The methodology is composed of three main steps, which are executed cyclically in order to progressively reduce the estimated energy consumption of the network, until a user-provided quality constraint is respected. The first step is a *characterization* of the output error sensitivity to the computations performed in each neuron (i.e. node) of the network. Then, computations within nodes labeled as error tolerant are approximated by means of bit-width reduction. Finally, the network is incrementally retrained to partially restore the original output quality (e.g. classification accuracy) despite the previously introduced approximations, thanks to the self-healing properties of NN training. Once the approximate NN model has been generated, the authors of [98] propose to offload its evaluation onto an accelerator named *quality-configurable Neuromorphic Processing Engine* (qcNPE). In this hardware, energy reductions when precision is downscaled are achieved by means of input and clock gating.

Similar considerations and analyses are performed in the work of [99]. In this case, however, network *quantization* (i.e. bit-width reduction) is performed on a per-layer basis, rather than per-neuron. Moreover, quantization is combined with *computation skipping*. This additional optimization is motivated by the fact that many of the trained parameters in a neural network have magnitude close (or equal) to zero. Considering the very large number of parameters of high-end NN models, computation skipping can provide relevant energy benefits, especially by avoiding unnecessary memory accesses. The hardware accelerator proposed in [78] by the same authors exploits both these algorithm-level optimizations to improve energy efficiency. Specifically, one important improvement in this system with respect to [98] is the usage of DVAFS (introduced in Section 2.1.1) for quantized operations. A similar framework is introduced in [100], where quantization and computation skipping are combined with further approximations by means of inexact functional units.

The bit-width reduction concept of network quantization has been brought to the extreme in *Binary Neural Networks* (BNNs) [101], i.e. neural network models in which the weights and internal variables are constrained to assume one of two values (e.g. -1 or 1). As long as the training process is opportunely modified, these networks have been demonstrated to suffer very small accuracy degradations compared to reference floating point models, at least for simple classification tasks (e.g. handwritten digit recognition). At the same time, they allow orders of magnitude energy benefits, thanks to the replacement of the basic MAC kernel involved in neurons computations with binary operations, and to the dramatic reduction of the weights memory footprint. An example of a complete hardware accelerator for BNNs can be found in [102].

The works of [103] and [104] propose *dynamically* EQ scalable NN accelerators. Indeed, previous techniques only involved static optimizations: although different parts of the network (e.g. neurons, layers) were treated differently, optimizations (e.g. quantization, skipping, etc.) were done independently on runtime inputs. In contrast [103] proposes *big/little* neural networks for classification tasks, in which energy and quality are traded off by using two NN models of different sizes (obtained by changing the number of layers, neurons, etc.) for different inputs. The "little" NN is used by default, in order to classify as many inputs as possible with a reduced energy consumption. The "confidence" of this classification is then evaluated by means of a metric called *score margin*, and if it is lower than a pre-defined threshold, the classification is repeated with the "big" NN. This allows to obtain an accuracy comparable to that of the big NN, with a consumption comparable to that of the little NN. The work of [104] further improves this concept, by designing the little network to be a subset of the big, hence reducing the overheads associated with two complete models (e.g. memory occupation for weights). The same concepts are extended to other machine learning algorithms besides neural networks in [105], which introduces a general framework for designing *scalable effort classifiers*. Specifically, the authors describe general techniques for building cascades of increasingly more complex and more accurate classifiers, and propose an algorithm to automatically determine the proper number of cascade elements and the decision criteria for accepting or rejecting the output of each.

An example of EQ scalable accelerator design not targeting NNs (but still in the domain of machine learning) is presented in [106]. The authors of this work use the design of a hardware accelerator for supervised classification using *Support Vector Machines* (SVMs) as a case of study to demonstrate a general principle, named *Scalable Effort Hardware* (SEH) design. The fundamental idea of this technique is to combine EQ scalable design techniques at different abstraction levels. Specifically, in [106], the SVM accelerator is designed combining computation skipping (algorithm-level), variable operand precision (circuit-level) and voltage-scalable meta-functions (architecture-level).

2.4 Software-Level Techniques

Energy-quality scalable design techniques at software-level mostly deal with algorithms that exhibit the error tolerant computational patterns described in Section 1.1.3 (iterative refinement, statistical aggregation, self-healing, etc.). Once again, there is not a clear distinction between these approaches and those listed in Section 2.3. For example, many of the algorithmic solutions adopted for hardware acceleration (e.g.

quantization and computation skipping in NNs) also provide energy and performance benefits when applied in software.

Best-effort computing is a technique for software performance improvement and energy consumption reduction, applicable to iterative refinement-based algorithms [30, 15]. The original idea comes from the *Internet Protocol* (IP) in the networking domain, in which packets are transmitted with a best-effort contract, i.e. they are not guaranteed to be correctly received [10]. Transport-layer protocols of the Internet stack provide both reliable (TCP) and unreliable (UDP) services on top of the unreliable layer provided by IP. This duality allows to separate transfers that need to be delivered correctly from transfers that can accept some data loss.

The same concept is explored in the context of parallel computing platforms for iterative error resilient algorithms. A best-effort layer is implemented on top of the existing operating system functionalities, in form of a software library. The algorithm is then divided in *guaranteed computations*, which are fundamental for the final result, and *optional computations*, which can tolerate errors without a significant impact on output quality. The relaxed correctness requirement on the latter group is exploited in the best-effort library to perform different types of optimizations:

- These computations can be skipped entirely, for example to reduce the system workload or to avoid exceeding the power budget.
- The synchronization between parallel computations can be eliminated, facilitating concurrency at the cost of possible dependencies violations.
- They can be performed on EQ scalable hardware (e.g. a module operated in VOS or a RRC core in ERSA).

All these mechanisms are managed by the best-effort layer, and are transparent to the software engineer.

A simple yet very effective software approximation method is *loop perforation*, introduced in [107]. In this work, the authors show that, for many applications, significant software performance improvements can be obtained by selectively skipping some iterations of looping code (e.g. for or while loops). For example, convergence of an iterative refinement algorithm can still be achieved despite skipping some iterations; similarly, aggregate metrics computed on a set of partially redundant data may not be dramatically affected by ignoring few samples, etc. In order to take advantage from this property, the authors propose compiler optimizations that first split the main software loops into *critical* (i.e. that cannot be approximated) and *tunable* groups. Then, for tunable loops, they propose algorithms for exploring the performance versus output quality design space achieved by skipping the iterations of each loop at different "rates". This simple technique allows up to 7x performance improvements for some applications. Moreover, reducing the computation time also yields corresponding energy savings.

Finally, programming languages to support approximations (at both software and hardware levels) have been developed. Examples include EnerJ [46] and Rely [108]. EnerJ is a Java extension that lets a programmer annotate variables and operations as either *approximate* or *correct*. At compile time, the information flow of variables is checked to prevent illegal operations (e.g. the content of an approximate variable cannot be assigned to a correct variable, unless the former is explicitly *endorsed*). At runtime, approximate variables can be optimized in several ways, for energy savings and performance improvements. For instance, they can be stored on unreliable memories (see Section 2.6), or updated with the result of an inexact operation. Finally, the programmer can also provide two different (software-level) implementations of a function for correct and approximate arguments respectively. The Rely language [108] allows a programmer to specify the desired *reliability* of each operation result (defined as the probability of correctness), as a function of the reliability of its inputs. Given this information, and a model of the target EQ scalable hardware, the Rely compiler can provide statically-computed statistical tests results on the probability that the program can be executed on that hardware with the required reliability.

2.5 System-Level Techniques

System-level EQ scalable design literature deals with two main problems. The first is the semi or fully automatic *assessment of the reliability* of an application. This type of analysis, performed in the first design stages, is fundamental to drive the design of lower level components to target the most relevant error tolerant parts of an application, and consequently obtain the largest possible benefits.

The second problem is the *dynamic management* of the output quality at runtime. This objective must be achieved by *synergistically* combining multiple EQ knobs (i.e. parameters that affect the error and energy consumption of a system), that could operate at different abstraction levels, and is aimed at maximizing the *total* savings while respecting the imposed quality constraints. Dynamic quality management further involves two main aspects: light-weight *quality estimation* by means of checkers, and *quality control* through the configurable knobs.

Concerning the first objective, one of the most complete works is the one of [9]. Therein, the authors describe the so-called *Application Resilience Characterization* (ARC) framework. This tool takes three inputs: a high-level software model of the target application, a set of representative input stimuli, and a user defined output quality metric. It then uses *Dynamic Binary Instrumentation* (DBI) to inject errors in different elementary sections of the application (e.g. inner loops bodies), referred to as *computational kernels*. The impact of injected errors on each section is evaluated, and kernels are split into resilient and not resilient. This phase is denoted as *resilience identification*, and is similar to the analysis performed in [46] for loop perforation.

Kernels identified as resilient then undergo a following step, named *resilience characterization*. In this phase, the impact of different types of EQ scalable design techniques on each kernel is analyzed. This is achieved again through DBI; in this case, however, errors are injected in accordance to a model of the output effects produced by the considered technique (*approximation model*). The ARC framework supports the characterization of techniques that work both at the circuit-level (inexact FUs and precision reduction) and at the algorithm-level (e.g. loop perforation and other algorithm manipulations). Different techniques are modeled differently in terms of injected errors, and high-level statistical approximation models are also available to test the impact of errors without referring to one particular technique. For each kernel and approximation model, an output *quality profile* is produced, i.e. a table reporting the value of the user-defined continuous quality metric as a function of the model's parameters.

A complete framework for runtime quality management is proposed in [109, 39], with the name of *Dynamic Effort Scheduling* (DES). The motivation for DES (and similar approaches) is that the tolerable amount of error varies throughout the operating life of a system, as explained in Section 1.2.4. Therefore, setting the amount of error of each EQ scalable component statically may result in violations of the minimum quality constraints, or missed opportunities for energy saving. In contrast, DES allows to change the amount of error (and power consumption) at runtime, by dynamically tuning a set of software and hardware knobs.

In [39], in particular, the same three approximation techniques proposed in [106] are considered, i.e. computation skipping, reduced precision and voltage over-scaling. Each of the three is paired with a mechanism for dynamically tuning the amount of error, or *effort* (e.g. supply voltage is the knob associated to voltage scalable meta-functions). At runtime, knobs are set automatically to achieve the optimal quality/power level, using a control loop scheme, very similar to the one shown in Figure 1.3. To this end, a method for runtime quality estimation is needed in order to obtain information on the current level of output quality, and to correct the values of the knobs accordingly. For each of the three approximation techniques, a corresponding quality sensor is therefore defined. Specifically, the authors propose to use Razor flip-flops to measure errors due to voltage scaling, comparators on input variables to evaluate the truncation error introduced by precision reduction, and checks on application-specific variables to estimate the impact of algorithm-level simplifications. Depending on these sensor outputs, a *Proportional Integral Derivative* (PID) controller is used to determine a correction factor on the optimal amount of error. The latter is in turn used to select a combination of knobs values from the set of Pareto optimal settings. The previously described ARC framework is employed to optimize a DES architecture similar to the one described above in [12].

A number of other frameworks has been proposed to achieve the same objective as DES, including *Green* [110], *Sage* [111], *Rumba* [112, 113], and *Mithra* [114]. These works do not use the term DES explicitly, but they all target the problem of online *quality management*. Each framework uses different types of checkers and quality knobs (hardware and/or software). Nonetheless, the general architecture invariably follows the organization shown in Figure 1.3, which can be therefore considered as the most general view of an EQ scalable system.

For example, *Sage* [111] approximates GPU code for performance by skipping expensive computations (quality knob). It uses a tree-based greedy algorithm to select the optimal quality mode (quality control) and periodically checks the output error by performing an accurate execution every N invocations of the target code (quality estimation). Similarly, *Rumba* [112] selectively schedules accurate computations on a CPU, and error tolerant computations on an approximate accelerator, depending on a user-defined quality constraint. Accelerators outputs that exceed the maximum desired error are detected by means of lightweight checks (quality estimation). These checks can be implemented using linear regression or decision trees trained on the accelerator inputs, or comparing the current accelerator output with an exponential moving average of its previous outputs. The thresholds used in these checks to detect an error can be tuned at runtime (quality knob), depending on the target output quality or energy consumption. Consequently, the number of iterations accurately executed on CPU is increased or decreased (quality control). All other frameworks [110, 113, 114] follow similar principles.

2.6 The Energy-Quality Tradeoff Beyond Processing

As mentioned at the beginning of the chapter, EQ scalable design techniques that work on other components of a system besides processing are much less explored by the literature. In this section, we review some of the most relevant proposals on this subject, considering the three main components of a digital system besides processing, with reference to the architecture of Figure 1.5: *memory elements*, *interconnects*, and *peripherals*. In particular, while in the context of an embedded system, peripherals include all kinds of *sensors* and *actuators*, in this thesis we only focus on *Organic Light-Emitting Diode* (OLED) displays. The reason for this interest is twofold. On the one hand, displays are often among the most relevant contributors to the total energy consumption of embedded systems; on the other hand, the emissive nature of OLED devices offers unique opportunities to explore the energy versus quality tradeoff.

In the rest of this section, we briefly analyze existing techniques for EQ scalable design in memories. We then go into more details for what concerns works on interconnects (specifically serial I/O buses), and OLED displays, as these are related to our techniques proposed in Chapter 4 and Chapter 5.

2.6.1 Memories

Techniques that exploit applications error resilience in *Static Random Access Memories* (SRAMs) are mostly based on applying aggressive supply voltage scaling on memory cells, thus reducing both leakage and dynamic power. However, voltage scaling also reduces the *Static Noise Margin* (SNM) of SRAM cells, and may cause errors when data is read from the memory, hence generating a typical EQ tradeoff. One of the earliest works that explores this tradeoff is found in [115], where the authors propose a *hybrid* SRAM architecture, composed of standard *6-Transistor* (6T) cells combined with robust 8T cells. While the latter can be operated at lower voltage without incurring in read/write errors, their larger size induces relevant area overheads. Therefore, the hybrid memory uses 8T cells for MSBs, and 6T for LSBs. A similar idea is adopted in [36], but using low-overhead *Error-Correcting Codes* (ECCs) instead of larger memory cells. In this work, an unequal error-protection ECC scheme is proposed, in which higher-order bits are protected against a large number of voltage scaling-induced errors, whereas a simpler correction scheme is used for LSBs, thus reducing the associated overheads. A dynamic programming approach is proposed to find the optimal error protection for different groups of bits within a memory word, according to an error metric that takes bit-significance into account. More recently, the work in [116] proposes an architecture in which different bits of a memory word receive different supply voltages. The voltage applied to each bit is changed dynamically depending on the operation (read or write) and on the required quality. This paper also provides a good survey of other similar techniques for SRAMs.

Differently from SRAMs, *Dynamic RAMs* (DRAMs) must be constantly refreshed in order to avoid data losses due to the discharge of cell capacitors. EQ scalable design techniques for these memories focus on reducing the refresh rate, which is a relevant contributor to the energy consumption of DRAMs. In [117], a technique called *Flikker* is proposed, in which a DRAM is split in two parts: a critical region, which is refreshed normally, and a non-critical region, which is refreshed at a lower rate. The latter is used to store data that can tolerate errors. At the software level, error tolerant data are identified through annotations. In the work of [118], the entire memory is refreshed at a low rate. The rationale of this approach is that the capacitor charge retention time is different for every cell, but the normal refresh rate is calculated using worst-case assumptions. Therefore, even if the memory is refreshed with longer intervals, only few of its cells will actually generate errors. Thus, the authors propose to perform an extensive characterization of the memory, and to rank its pages depending on the number of errors. Data are then stored in different pages depending on their criticality. With respect to Flikker, this solution has the advantage of not requiring DRAM hardware modifications, as a single refresh rate is used for the entire chip. The work of [119] goes even further, by completely disabling DRAM refresh. With examples from graph analysis and LDPC decoding, the authors show that as long as the lifetime of data is shorter than the DRAM retention time, or the application can tolerate errors for a given time-span, DRAMs with disabled refresh can still be used successfully. A summary of EQ scalable DRAMs is provided in [120].

Finally, the work in [121] applies similar ideas to non-volatile memories. The paper focuses on *Multi-Level Cell* (MLC) *Phase-Change Memory* (PCM) technology, but claims that similar techniques can be used for MLC Flashes as well. Two methods to exploit application error tolerance on these devices are proposed. The first involves reducing the number of current pulses required to program a PCM (or Flash) memory cell. This affects the robustness of the written datum, possibly causing read errors, but also reduces the time and energy required for writing. The second method involves using blocks of memory affected by static faults (caused by wear out) to store error tolerant data. This solution increases the lifetime of the memory, which can be used as long as its fault-free portions are large enough for storing critical data.

2.6.2 Serial Interconnects

Interconnects can be relevant contributors to the total power consumption of an embedded device. In particular, long off-chip interconnects, such as cables or microstrips implemented on *Printed Circuit Boards* (PCB), do not follow the same technology scaling trends of integrated circuits over time. Consequently, the energy consumption of these connections, which is a function of their electrical characteristics, is becoming increasingly more relevant compared to that of ICs. This is especially true for ultra low-power devices, such as IoT sensor nodes, which may be equipped with very simple (and low power) processing elements, while being connected to tens of sensors, each through a dedicated off-chip bus.

Serial links are by far the most common choice for interconnecting peripherals (sensors, actuators, and I/O controllers) in digital systems, due to their numerous advantages over parallel interfaces. Thanks to the absence of skew, crosstalk, and jitter issues they allow higher signaling frequencies than their parallel counterparts. Moreover, they result in cheaper implementations thanks to the lower pin count and easier routing [122]. Standards protocols for this kind of connection include *Serial*

Peripheral Interface (SPI) [123], *Inter Integrated Circuit* (I2C) [124], *Controller Area Network* (CAN) [125], and others.

The main source of energy consumption when transmitting data over long serial interconnects is the charging and discharging of large capacitances (due to PCB strips, cables, or integrated metal routes), occurring in correspondence of changes in the transmitted voltage. In the digital domain, these voltage variations correspond to changes in the transmitted logic value. Therefore, most literature on energy reduction for serial interconnects focuses on *encoding* the data sent on the bus, so that the number of logic transitions in *codewords* (i.e. the outputs of encoding) is smaller than that of the original *data words*. In general, both *inter-word* transitions (i.e. bit changes within a single serialized word) and *intra-word* transitions (i.e. bit changes between the last transmitted bit of the previous word and the first of the current word) are considered. The sum of all transitions during the operation of a bus is referred to as the *Transition Count* (TC).

Classically, literature on bus encodings focused mainly on parallel buses (e.g. for the connection between a processing element and main memory). However, due to the aforementioned diffusion of serial interconnects for interfacing with sensors and actuators, recent years have seen an increasing interest on low-power serial bus encodings. These proposals can be broadly categorized into two main groups: *lossless* or *accurate* encodings, and *lossy* or *approximate* encodings. The former are solutions for which, in absence of channel errors (e.g. bit-flips caused by external interference, shorts, etc.), the transmitted data can be decoded exactly by the receiver. Approximate solutions, in contrast, explore the energy versus quality tradeoff, by sacrificing perfect decoding in exchange for larger TC reductions, and consequently larger energy savings. The techniques proposed in this thesis, being EQ scalable, belong to this last category. Nonetheless, in this section, we also briefly review some of the most relevant accurate serial bus encodings, as these will then be used as starting points or comparisons for our proposals.

Lossless Serial Bus Encodings

Lossless bus encodings can be adopted for all types of I/O, i.e. data, addresses and instructions. Some of the most relevant techniques belonging to this family have been proposed in the context of *Network-on-Chip* (NoC) links, but their scope of application can be extended also to off-chip buses. The common approach followed in lossless encodings is to reduce the number of transitions for transmitting the *most probable* data words, following the typical *common case optimization* principle of low-power design. This requires to have some form of a priori knowledge (typically statistical) on the data being encoded.

The work of [126] leverages statistical correlation among subsequently transmitted words (i.e. the *temporal correlation*) to obtain TC reductions. The proposed approach takes the name of *Serialized Low-Energy Transmission* (SILENT) coding, and its basic underlying principle is that of *Differential Encoding* (DE). In DE, rather than the original binary representation of a word, what gets sent on the bus is the bitwise difference with respect to the previous word. As detailed in Chapter 4, DE is based on the observation that temporally correlated data tend to have a small *Hamming Distance* (*HD*) on average, with differing bits located in the LSB-part of words. Therefore, when these data are encoded as bitwise differences, the resulting codewords will often contain a long string of 0s in the MSBs. Once serialized, the latter does not incur any value transition, and hence contributes to reducing the TC.

While SILENT reduces the TC when data are strongly correlated, it actually incurs significant power overheads for uncorrelated data [126]. In order to overcome this limitation, the authors of [127] propose a hybrid encoding scheme, called *adaptive low-power transmission coding*. In this solution, the hamming distance between subsequent data is estimated using a simple FSM-based prediction scheme. Depending on the outcome of this prediction, one of two possible encoding schemes is selected: for low estimated HD, the same DE-based encoding proposed in SILENT is used. Vice versa, when the bitwise difference between words is estimated as large (i.e. when SILENT would generate overheads), the codeword is generated taking the difference among the n-th word, and the $(n-2)$-th one. An additional flag bit is used to inform the decoder about the chosen encoding scheme.

In [128], *transition inversion coding* is proposed. In this scheme, the number of intra-word transitions generated by serializing the current data word is computed within the encoder. If this number is less than half of the word length, the data is transmitted as is (i.e. the codeword is equal to the data word). In the opposite case, the bits in positions with and even index (0, 2, etc.) are inverted before transmitting. As before, an additional *invert* flag bit is added as a header to the codeword, so that the decoder can correctly recover the original datum. An advanced variant of this algorithm is also proposed in the same paper, to account also for inter-word transitions. Finally, the authors also show how their encoding can be enhanced with a parity check bit, to achieve single-bit error detection.

In [129], the proposed encoding achieves TC reduction by swapping the positions of some bits within words. Specifically, whenever a 3-bit sequence with two transitions is found within a serialized data word (i.e. 010 or 101), the last two bits are swapped, thus reducing the number of transitions to 1 (obtaining 001 and 110 respectively). Such swaps are signaled to the decoder by means of two additional lines, encoding the position (or positions) of the swapped triplet(s) within each byte. While this solution allows TC reduction regardless of the temporal correlation among subsequent data, it incurs significant overheads due to the aforementioned need for additional physical lines.

In general, all algorithms that extend SILENT to reduce its overheads require some form of *redundancy*, in order to achieve successful decoding. This additional information is either transmitted on a separate physical line, as in the last example, or added as header to the encoded words. Differently from the case of parallel buses, this type of redundancy is extremely costly for a serial connection. Indeed, adding physical lines effectively doubles (in the worst case) the cost of the connection in terms of area, used pins, etc [4]. On the other hand, adding redundant bits on the same line as the data reduces the effective bandwidth of the bus, as a smaller number of data bits are transmitted for a given signaling frequency.

One important lossless scheme that avoids redundancy is proposed in [130]. As for SILENT, this technique makes important assumptions on the statistics of the transmitted data. Specifically, this encoding targets the data transmitted serially between a processing element and a display controller, using protocols such as *Digital Visual Interface* (DVI) and others. Exploiting the locality of image data (see Chapter 4), arithmetic *differences* between subsequent pixels are transmitted rather than their binary values, and a *Look-Up Table* (LUT) is used to map the most commonly occurring differences (i.e. the smallest ones), to codewords with a small number of serial transitions.

[4]This is true if the original serial bus only contained one physical line. For standards that have more than one connection (e.g. I2C [124] has separate data and clock lines), the cost is not doubled, but still significantly increased.

Lossy Serial Bus Encodings

Lossy encodings introduce approximations in the communication, making the received data words potentially different from the transmitted ones. For this reason, these solutions are only suitable for *data* buses, and cannot be applied to addresses and control. This is analogous to the case of processing hardware, were approximations are applicable in the datapath but mostly not in the control path. The somewhat more limited scope of applicability of lossy encodings is compensated by the larger achievable power savings.

One of the first proposals in this field is found in [131]. As for [130], this encoding targets the serial transmission of data to a standard display interface. For the two most common physical level protocols used within display standards (e.g. DVI), that are *Low-Voltage Differential Signaling* (LVDS) and *Transition-Minimized Differential Signaling* (TMDS), the authors propose two corresponding lossy encodings that exploit the limited human perception of small color differences. Specifically, both techniques approximate the transmitted colors, under a maximum deviation constraint. Two versions of each encoding are proposed: a *static* one, in which the amount of error is fixed at design time, and a *dynamic* one, in which the quality constraint can be changed at runtime. In the former case, the optimal codeword for each data word is obtained through an offline search process, before being stored in a table for runtime lookup. Dynamic encodings, instead, rely on a smart rounding of some LSBs of the transmitted pixels, which accounts for the underlying physical layer, and also considers inter-word transitions.

The work of [132] is the first to formally consider serial bus encodings from the point of the energy versus quality tradeoff, as an instance of the approximate computing paradigm. The proposed encoding, called *Rake*, heuristically tries to minimize the TC for each transmitted word, under a maximum value deviation constraint, analogous to the one of [131]. This TC reduction is obtained by inverting a sequence of ones or zeros within the data word. The error introduced by this inversion is then compensated by also inverting sequences of lower-order bits of opposite polarity. The entire process is performed by scanning the bits of the data word twice, resulting in a linear time complexity with respect to the word length. Interestingly, Rake does not make any assumption on the data statistics to reduce the TC. Nonetheless, the authors demonstrate that when considering uniformly distributed data, their algorithm achieves better TC reductions for the same average error, compared to a simple rounding of LSBs (i.e. the approach taken in [131]).

Finally, a recent and very effective lossy encoding has been proposed in [133]. This work has been published after the first dissemination of the techniques described in Chapter 4 through our papers found in [51, 52], and is heavily inspired by the latter. Therefore, we postpone the detailed analysis of this technique to Chapter 4.

2.6.3 OLED Displays

Displays are often among the most power hungry peripherals in digital systems (see Figure 1.6b). In classic *Liquid Crystal Displays* (LCDs) the light produced by an external source is transmitted or reflected by pixels, which filter the input wavelengths, in a way that depends on the electric field applied on their terminals. Studies on these displays have shown that the *backlight* (i.e. the external source) can account for up to 80% of the total power consumption [134]. Conversely, the intensity of the pixels in the displayed image has a limited impact on the consumption of the display. Therefore, the vast literature on LCDs energy efficiency mostly leverages *backlight scaling* as a power reduction knob [134–139]. However, backlight scaling also causes a reduction of the overall brightness in the displayed image. To compensate for this unwanted effect, many works use some kind of *image transformation* (i.e. a modification of the pixels intensities), so that the perceived image quality is preserved as much as possible, despite the diminished brightness.

OLED displays have a radically different working principle, that opens new possibilities for exploring the energy versus quality tradeoff. Indeed, these devices are *emissive*, i.e. light is directly produced by the panel pixels, without any external source. In particular, each pixel is formed by a combination of three emissive devices, corresponding to the red, blue and green components of the RGB color space respectively [140]. The intensity of each component depends in a nonlinear way on the current flowing through the corresponding device. Therefore, the total power consumption of the panel is strongly dependent on the *brightness* of the displayed image, and secondarily on the balance between color components.

Power reduction techniques for OLEDs are tailored to this image-dependent power consumption. In particular, one important novel possibility, with respect to the case of LCDs, is that of altering power consumption simply by *transforming the* *displayed image*, changing the intensities of pixels. These transformations do not require any type of hardware control, and can be implemented fully in software. Thus, OLEDs enable a much larger space of possible power saving solutions compared to LCDs. In general, methods presented in literature can be broadly split in two categories: those targeting *Graphical User Interfaces* (GUIs), and those applicable to general images.

Techniques for GUIs are based on the observation that, for a user interface, the most important requirement is *usability*, rather than visual *fidelity*. In other words, making sure that the user is able to distinguish text or icons from the background is more important than preserving the exact colors of the page. Following this principle, the authors of [141] propose a technique that dramatically alters the color palette of a GUI, exploiting the color dependence of power. In practice, GUI elements colors are shifted towards black and green (the latter is the least consuming RGB color component for many real devices), to obtain progressively higher power savings, e.g. when the battery state of charge of a device goes below a given threshold. This work is then extended in [140], where a web browser application able to optimize OLED power consumption is proposed, called *Chameleon*. This browser lets a user select among multiple possible power saving transformations, including the previously mentioned color shift, as well as color inversion, global *darkening* (i.e. luminance reduction), etc. The selected transformation is then applied to each new displayed web page.

A different approach for GUIs is proposed in [142]. The authors motivate this work with the consideration that, in many types of interfaces, the user typically focuses only on a small portion of the display (e.g. the foreground window in a window-based graphical environment, the selected item in a settings list, etc.). Therefore, they propose to selectively darken the pixels outside of the area of user interest. This area is determined by the GUI software, either by approximating it with the active window, or by monitoring screen usage patterns. A more modern application of the same principle is found in [143]. Differently from the previous approach, the authors propose to automatically identify the degree of user attention for each region of the screen using the so called *saliency map* method. Then, pixels are dimmed depending on their region, using a progressive scaling factor that avoids the creation of artifacts at region boundaries.

Approaches that target GUIs cannot be directly extended to general images or videos. Indeed, for the latter, visual fidelity is a fundamental aspect of perceived quality, and drastic modifications of color and/or luminance such as those proposed in the aforementioned works would be unacceptable for a user.

An interesting solution for general images, proposed in [144], consists in applying *Dynamic Voltage Scaling* (DVS) to the OLED panel. The aim of OLED DVS is that of reducing the *maximum current* that can flow through pixel devices, by means of a custom driver circuit. Clearly, this has a direct impact on the total power consumption of the display, but also reduces the brightness of the displayed image. Therefore, with a rationale similar to that of LCD backlight scaling techniques, brightness reduction is then compensated in software, modifying pixel values. In the work of [145], the same principle is extended to *fine-grain* DVS. In this solution, a different supply voltage is applied to different areas of the display panel; specifically, areas containing *major objects* (identified as groups of pixels with a similar RGB histogram), are scaled less aggressively than the rest of the display. Similar to the previous proposal, image compensation is leveraged to reduce the visual impact of supply voltage scaling.

In order to achieve power savings without resorting to custom OLED panel drivers, many techniques have been proposed that act *solely* on pixel values. Differently from the previous group, these approaches are also applicable to to off-the-shelf OLED displays, that may not support DVS.

Some works in this group propose to transform images in the most straightforward way that provides power benefits, i.e. a linear reduction of all pixel values. Two variants of this *brightness scaling* are described in [146] and [147]. In both cases, the goal of the proposed algorithms is to minimize power consumption under a minimum quality constraint. Importantly, rather than using a constant scaling factor for all images, these methods compute a *different factor* for each new input. This allows to scale the brightness of an image more or less aggressively, depending on the effect that this transformation has on its content. Specifically, the solution in [146] only scales brightnesses higher than a minimum knee-point, and quantifies the acceptable quality reduction in terms of maximum *MSE* between input and output images in RGB (which is equivalent to a minimum *PSNR*, as explained in Section 1.2.2). The work in [147], instead, uses a uniform scaling for all brightnesses, and its quality constraint is specified as a minimum threshold on the *MSSIM* between input and output images, which correlates better with the subjective perception of image alteration compared to the *MSE*.

Other OLED image transformations try to concurrently reduce power and *enhance* the target image, by applying a nonlinear transformation. A fundamental work in this category is *Power-Constrained Contrast Enhancement* (PCCE), first described in [148]. Starting from the observation that *chrominance* variations in a photo or video strongly alter the perceived quality (differently from the case of GUIs), the authors propose to only transform the *luminance* of each pixel. In particular, for each target image, the PCCE algorithm computes a customized and nonlinear pixel luminance mapping, i.e. a function that associates the luminance of a pixel in the input image to a corresponding luminance in the output image, in a nonlinear way. This mapping is determined according to an objective function that concurrently optimizes two objectives: while attempting to minimize the output image power consumption, it also tries to maximize its contrast. By balancing these two objectives appropriately, the inevitable brightness reduction needed to save power is compensated by the contrast increase, thus preserving the perceived visual quality. Many variants of the original PCCE have been proposed in following works. For example, the solution in [149] uses multi-scale retinex to generate a variant of PCCE, while the authors of [150] propose a similar transformation that does not require an iterative optimization procedure. Finally, transformations with similar objectives to PCCE, but leveraging a different internal procedure, are described in [151] and [152].

One common problem of the aforementioned transformations is that their complexity is too high for real-time implementation, unless they are executed on powerful processing hardware (e.g. a high-end CPU). However, in the latter case, the energy benefits on the display are likely to be offset or even nullified by the overheads for implementing the transformation itself. This is the main motivation for our proposed low-overhead transformations for OLEDs, described in Chapter 5, and will be presented in more detail therein.

To the best of our knowledge, only two previous works have tackled this issue directly. In both cases, overheads are reduced by moving the majority of the computational effort to the *image acquisition phase*, and in particular to the shooting of a picture. The work of [153] proposes a custom camera application for mobile devices (smartphones, tablets, etc.) that directly transforms the images it acquires, in a way that preserves contrast and concurrently reduces power. By means of a slider, the user of this application can set the amount of contrast enhancement and power reduction for each knew image. In [154], instead, the acquisition phase is used to analyze the image and store some metadata, such as focus length, exposure, and white balance. These data are then used at display-time to optimize power and image quality. The advantage of these solutions is that the image acquisition phase does not have the same real-time constraints of the display. Therefore, the complexity of the image analysis and/or transformation can be higher. However, both these methods only work if the images are acquired through a modified camera application. Hence, they cannot be applied to synthetic images (e.g., GUIs, computer graphics, etc.) nor to photos obtained differently (e.g., downloaded from the web, scanned, etc.).

Chapter 3

Automated Design of Energy-Efficient Quality-Configurable Datapaths

In this chapter, a set of techniques is presented for designing energy-efficient, quality-configurable digital datapaths. These proposals focus in particular on automation aspects and on generality, having as main objective that of being easily integrated into existing EDA flows. The first technique described consists of a framework for the automatic design of datapath units (or operators) based on *Reduced-Precision Redundancy* (RPR), introduced in Section 2.2.2. Next, we propose two different techniques for designing the same kind of units, but using the principles of *Dynamic Accuracy Scaling* (DAS) and its variants, described in Section 2.1.1, to achieve multiple energy-quality configurations. The content of this chapter is an extended and improved version of our previous publications found in [50, 33, 56, 57].

3.1 Motivation

As mentioned in Chapters 1 and 2, a large body of previous literature has focused on designing processing elements that exploit the error resilience of applications. However, the state of the art is still immature on two important aspects: industrial-level *automation*, and *generality*.

One first fact that shows the lack of generality is that, as mentioned in Chapter 2, a great number of literature works propose designs in which the output quality is *fixed at design time*. These solutions can only be applied to very specific scenarios, since for most realistic applications, the necessary quality level varies significantly at runtime (see Section 1.2.4).

All techniques presented in this chapter, on the contrary, allow to reconfigure the amount of error at runtime. Precisely, the approach based on RPR allows a minimal form of reconfigurability, by supporting two *energy-quality modes*, i.e. accurate and approximate, depending on the supply voltage applied to the circuit. The techniques based on DAS/DVAS, instead, permit a much larger set of quality modes, at most equal to the number of input bits of the unit.

Although there have been previous proposals for the design of processing hardware with configurable quality at runtime, these still lack in generality, being normally tailored to one specific architecture (e.g. one type of adder or multiplier). Moreover, most solutions tend to be based on manual design, and the authors of these works do not consider issues related to the integration of their techniques within standard EDA tools and flows. Thus, industrial-level automation is still mostly unexplored for the EQ scalable design paradigm.

In summary, in the works described in this chapter, we tried not to propose “yet another” EQ scalable design, but rather some general and automatic flows that could be leveraged to design real integrated circuits.

3.2 EDA Flow for Reduced-Precision Redundancy

RPR is an EQ scalable design technique based on VOS, that has been shown able to provide large power savings when applied to several datapath units for digital signal processing applications [28]. The general background on this approach was provided in Section 2.2.2.

Previous literature on RPR only focuses on small families of units, and is mostly based on manual design. In particular, the reduced-precision replica used as EC block is designed by hand for each new circuit. Moreover, the output quality of the complete RPR block is estimated with a theoretical analysis based on statistical assumptions on the distribution of input/output data. As explained in the following, these assumptions are oversimplified, and neglect important aspects, such as the temporal correlation of inputs. Finally, the design of an optimal RPR architecture (most importantly the replica bit-width) is performed only considering dynamic power, and leakage is only evaluated a posteriori; however, considering that RPR introduces redundant logic, hence increasing leakage power, we claim that the latter should be considered as part of the optimization cost function.

In this section, we propose a new generalized approach to RPR, that allows to overcome these limitations. By relaxing some theoretical assumptions in favor of an empirical input-based analysis, and by leveraging state-of-the-art EDA tools, we construct a framework able to automatically add RPR to a pre-existent gate-level netlist. Due to its more realistic assumptions compared to the classic RPR formulation, our approach allows to obtain a much closer matching among the power savings considered during optimization and those of the final output design. We show that, when tested on the same families of designs treated with ad-hoc procedures in previous works, our approach reaches comparable savings. Moreover, we apply it to several architectures that were not considered previously, confirming its generality and flexibility.

3.2.1 Classic RPR Problem Formulation

In this section, we present a recap of the theory needed to formalize the problem of designing an optimal RPR architecture. A more exhaustive treatment of this subject can be found in [28].

With reference to Figure 2.6, in a RPR design, the MDSP output at time n and at nominal voltage (i.e. in absence of timing errors) can be expressed as:

$$y_{M,0}[n] = s[n] + \eta[n] \tag{3.1}$$

where $s[n]$ is the signal component, and $\eta[n]$ is a generic noise component, that may be due to different physical sources (e.g. environmental noise on inputs sampled from analog sensors). RPR saves power by accepting to introduce errors due to timing violations, which are only partially corrected by the EC block. These violations can be modeled as an additional source of additive noise, as explained in Section 1.2.

Therefore, in VOS conditions, the MDSP output becomes:

$$y_M[n] = y_{M,0}[n] + \gamma[n] = (s[n] + \eta[n]) + \gamma[n] \tag{3.2}$$

where $\gamma[n]$ is the additional noise due to timing errors.

The MDSP output y_M and the replica output y_R are then subtracted and compared against a threshold T_h. The global output y is selected according to the following decision scheme, first introduced in Section 2.2.2:

$$y[n] = \begin{cases} y_M[n], & \text{if } |y_M[n] - y_R[n]| \leq T_h \\ y_R[n], & \text{if } |y_M[n] - y_R[n]| > T_h \end{cases} \tag{3.3}$$

Timing errors only occur when the critical paths of the MDSP are activated. Therefore, a desirable property of this architecture is that $y = y_M$ in all other conditions (i.e. when no violations occur within the MDSP). To ensure this property, the threshold must be set as in (2.3). In this scheme, the residual VOS-induced error at the output of the decision block is:

$$\gamma_r[n] = y[n] - y_{M,0}[n] \tag{3.4}$$

When designing an RPR architecture, the goal is to make sure that the global output y satisfies a quality constraint. In general, the form in which such constraint is expressed is strongly *application-dependent*. However, as ANT designs (such as RPR) are typically used in signal processing applications, a commonly found expression of the output quality in literature is the *SNR*. In the context of RPR, the latter is defined as:

$$SNR = \frac{\sigma_s^2}{\sigma_{\gamma_r}^2 + \sigma_\eta^2} \tag{3.5}$$

where σ_s^2 is the signal power, σ_η^2 is the external noise power and $\sigma_{\gamma_r}^2$ is the noise power due to residual errors (assumed uncorrelated from σ_η^2):

$$\sigma_{\gamma_r}^2 = \mathbb{E}[\gamma_r[n]^2] = \mathbb{E}[(y[n] - y_{M,0}[n])^2] \tag{3.6}$$

A typical approach followed in ANT designs is to make the errors due to VOS-induced timing violations negligible with respect to external noise, i.e. $\sigma_{\gamma_r}^2 \ll \sigma_\eta^2$, as the latter imposes an upper bound on the achievable output quality.

The two design parameters that affect the residual error (and consequently the output quality) are the VOS supply voltage V_{vos} and the replica implementation. In particular, since the replica is a reduced width version of the MDSP, the possible implementations vary in terms of their error correction capabilities and power/area overheads, both function of the number of bits (B).

The identification of the best RPR architecture can be informally defined as: *finding the VOS voltage and feasible replica implementation that minimize total power consumption, including EC block overheads, under a minimum quality constraint, and so that the entire RPR system consumes less than the single MDSP in nominal conditions.* In formal terms, this corresponds to the following optimization problem:

$$\begin{cases} \min[P_{rpr}(V_{vos}, B)] \\ Q(V_{vos}, B) > Q_{des} \\ P_{rpr}(V_{vos}, B) < P_{mdsp}(V_{nom}) \end{cases} \tag{3.7}$$

In these equations, Q is a generic quality measure (e.g. *SNR*) and Q_{des} is the desired minimum quality constraint for the target application. V_{nom} is the nominal MDSP voltage, i.e. the minimum voltage that ensures timing-error-free operation at the target clock frequency. P_{mdsp} is the total power consumption of the MDSP (including leakage and dynamic power), which only depends on the supply voltage, as the hardware of the MDSP is fixed. Finally, P_{rpr} is the total power consumption of the RPR architecture (including the EC block overheads), which depends both on the VOS operating point and on the size of the replica, determined by B:

$$P_{rpr}(V_{vos}, B) = P_{mdsp}(V_{vos}) + P_{ec}(V_{vos}, B) \tag{3.8}$$

3.2.2 Limitations in the Classic Formulation

As mentioned in the introduction to this section, while the classic formulation of the RPR optimization problem is theoretically solid, it makes several simplified and unrealistic assumptions, which limit its generality of application.

A first limitation is in the way in which the decision threshold T_h is computed. Indeed, in order to use the expression in (2.3), the output values produced by the MDSP and replica for the entire set of possible inputs must be known. The computation of these *output images* requires a complete knowledge of the MDSP functionality, which is only possible if RPR is implemented manually. In contrast, an automatic tool that constructs a RPR architecture on top of an existing design (e.g. specified as a HDL description), cannot in general determine the functionality of the MDSP, except for few simple cases (e.g. adders, multipliers, etc.). Moreover, real applications often use only a portion of the input domain of a hardware block. Therefore, the theoretical threshold might correspond to an input combination that never occurs during real operation.

The most important issue, however, is related to the evaluation of the final output quality of the RPR architecture, which depends on (3.6) and (3.5) in the previously described formulation. In particular, some aggregate statistics on uncorrected VOS-induced errors, such as their variance $\sigma^2_{\gamma_r}$, computed in (3.6), are involved in the evaluation of most quality metrics. When this is the case, the overall output quality depends on two quantities: the distribution of output values, and the distribution of timing-errors.

The distribution of outputs depends, in turn, on the ones of all MDSP inputs. In [86], some relevant simplifications were made on the latter, i.e. a uniform distribution was assumed for each input. However, in realistic applications, the distribution of inputs is almost invariably not uniform, and in general does not follow any closed-form expression. In these cases, especially if the MDSP functionality is complex, an analytical evaluation of the distribution of outputs is unfeasible.

The distribution of errors (i.e. the probability and magnitude of each error) is required to evaluate the global RPR output y. In fact, the value of the latter depends both on the approximate outputs provided by the replica, and on the rate at which the replica output is selected.

In previous works, the error probability was estimated as the percentage of *static timing paths* that violated the positive-slack constraint under VOS. This percentage was computed based on simple CMOS delay models [28].

However, timing-errors are determined by *dynamic path activations*, i.e. they occur when a specific pair of inputs is applied to the MDSP, causing a critical path to be activated. Therefore, computing the error probability as the number of violating static paths implicitly assumes a *uniform distribution of all input pairs*. As mentioned, in most applications the probability of occurrence of input vectors is not uniformly distributed, let alone that of input pairs. For example, inputs are often temporally correlated, i.e. the value of an input at a given time is statistically influenced by the values occurred in the recent past, which violates the uniformity assumption. The assumption of uniformly probable path activation biases also the estimation of error significance, since every possible VOS-induced error is considered equally likely. This, in turn, affects the evaluation of which errors are detected and mitigated by the EC block, and of the corresponding approximation accuracy.

All these aspects influence significantly (3.6), and might produce misleading results (both overly optimistic or pessimistic) on the number of replica bits required to ensure a minimum quality at a given V_{vos}. Notice that these issues are not specific to the SNR, but apply to the majority of quality metrics (e.g. $PSNR$, MED, MSE, etc.).

Another important aspect that is partially neglected by literature is the effect of voltage scaling on the replica. In Section 2.2.2, we have mentioned that the replica must be timing compliant in VOS, in order to let the RPR system function correctly. However, this is not trivially achieved, and depends strongly on the topology of the circuit. It does not suffice to assume that the reduction in critical delay of the replica is proportional to the number of removed bits in its operands. Timing compliance must be checked accurately, and can also be enforced, for instance with gate re-sizing. The latter approach, however, may increase the power overhead and reduce the total savings of the RPR system. All these details cannot be taken into account by optimization methods such as the one proposed in [86], which use abstract high level models of MDSP and replica.

3.2.3 Overview of the Proposed Methodology

Because of the limitations described in the previous section, RPR designs based on the theoretical formulation are limited to a small class of MDSPs, and can only be optimized correctly under the unrealistic assumptions of inputs uniformly distributed and completely uncorrelated in the time domain. Our flow for the automatic design of RPR architectures addresses all the aforementioned issues. The following is a list of its main features:

- All design metrics (timing, power and area) are estimated accurately. To this end, we operate on a *gate-level* model of the MDSP, as opposed to the high-level models adopted in previous works. We evaluate power, timing and area, using $N+1$ *Liberty format* characterizations of the same gate library, corresponding to the nominal supply voltage V_{nom} and to a series of decreasing VOS operating points $[V_{vos,1}...V_{vos,N}]$. This method provides much more accurate estimates of savings and timing degradation with respect to the basic delay and power models used in previous works. Moreover, it allows to consider all power components, including the overhead introduced by the EC block, during optimization.

- Rather than the theoretical analysis used in the classical RPR formulation, our tool extensively relies on simulations, performed with realistic input stimuli, for evaluating all the quantities involved in the optimization problem of (3.7), and in particular the output quality. This choice is based on the assumption that a designer wanting to add RPR on top of an existing hardware has a good knowledge of the inputs that will be applied to the circuit during its operation. With respect to the theoretical formulation, simulations are never 100% accurate in evaluating quality, being based on a finite-length sequence of inputs. However, in practice, they permit a much more faithful estimation of the rate and magnitude of timing-errors. In particular, they allow to consider temporal correlation and non-trivial distributions of inputs, two aspects that could not be treated by simple statistical models.

- In our flow, the replica circuit is constructed automatically, starting from the gate-level model of the MDSP, and leveraging standard EDA tools directives for reducing its bit-width. This is done in an application agnostic way, i.e. without requiring any knowledge on the MDSP functionality. Our method is automatically able to discern whether the addition of RPR is advantageous or not, and to synthesize the replica configuration that maximizes power savings under a user-defined quality constraint. The entire process is automated, so that no additional coding effort is needed by the user in order to implement RPR on top of an existing design.

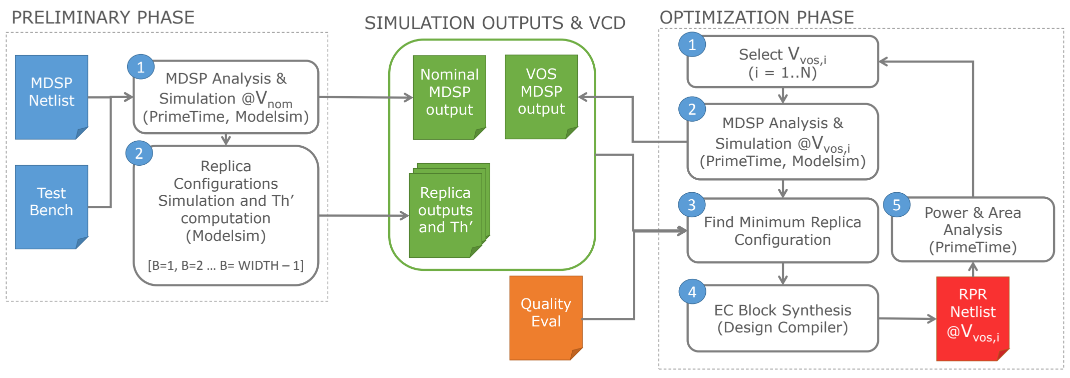

PRELIMINARY PHASE
MDSP Netlist
Test Bench
1 MDSP Analysis & Simulation @V_{nom} (PrimeTime, Modelsim)
2 Replica Configurations Simulation and Th' computation (Modelsim) [B=1, B=2 ... B= WIDTH – 1]
SIMULATION OUTPUTS & VCD
Nominal MDSP output
VOS MDSP output
Replica outputs and Th'
Quality Eval
OPTIMIZATION PHASE
1 Select $V_{vos,i}$ (i = 1..N)
2 MDSP Analysis & Simulation @$V_{vos,i}$ (PrimeTime, Modelsim)
3 Find Minimum Replica Configuration
4 EC Block Synthesis (Design Compiler)
5 Power & Area Analysis (PrimeTime)
RPR Netlist @$V_{vos,i}$

Fig. 3.1 Flow of the proposed RPR automation tool.

3.2.4 Design Flow

Figure 3.1 shows the flow of our proposed RPR automation tool. One important prerequisite that must be satisfied before running this tool is that the HDL file describing the gate-level MDSP must be annotated by the designer with some *pragmas*, specifying *input and output buses*.

With the term *bus*, in this context, we refer to a coherent set of interface wires in the MDSP, within which each wire represents a digit in a fixed-point number. A bus may often correspond to a *port* in the HDL specification, but this is not a strict requirement. A single port may contain two sets of wires which are logically grouped in different buses, and vice versa. An example of Verilog module definition annotated with pragmas is shown in Figure 3.2. Notice that clock and reset signals are not annotated, being error intolerant. The information on buses is needed by the tool in order to automatically perform the necessary simplifications when synthesizing the replica netlist starting from the MDSP.

```
module fir ( reset, clk, u, y );
  input [11:0] u;
  //pragma rpr bus(u) port(u) signed(N) weight(0) msb(11) lsb(0)
  output [23:0] y;
  //pragma rpr bus(y) port(y) signed(Y) weight(0) msb(23) lsb(0)
  input reset, clk;
```

Fig. 3.2 Example of a Verilog module annotated with the pragmas required by the RPR automation tool.

As mentioned, our optimization relies on simulations with realistic input data. The latter are provided to the tool in the form of a Verilog or VHDL testbench for the MDSP. Similarly, the choice on the most appropriate quality metric Q and on the corresponding minimum desired quality Q_{des} is also left to the user. To this end, the tool accepts a custom quality evaluation function, currently provided in the form of a Perl script. A different quality constraint can be specified for each MDSP output bus, and all of these constraints must be respected at the same time, in order for the tool to consider a given implementation valid. The quality evaluation function must receive two arrays of binary words for each output bus, representing the sequences of error-free and erroneous outputs respectively, and must return a single boolean output, indicating weather the desired quality constraint for that bus was respected or not. Standard quality functions (e.g., $Q \equiv SNR$) are provided in the tool, and can be used in place of custom ones if needed.

In summary, the inputs that a designer should provide to the tool are:

1. The gate-level netlist of the MDSP, annotated with bus pragmas.
2. A testbench generating realistic input stimuli.
3. Optionally, a custom quality evaluation Perl module.

All simulations, synthesis and netlist analysis steps within our tool are performed with commercial EDA software.

Preliminary Phase

The execution of the tool is split in two phases, grouped by dashed rectangles in Figure 3.1. In the first step of the preliminary phase (① in the figure), the MDSP netlist is simulated at the nominal voltage V_{nom}, in order to gather the set of *golden outputs* needed to measure quality during the optimization. The power and area of the MDSP at V_{nom} are also evaluated and stored. These metrics will be used as reference, and will be compared to those of each new RPR implementation. In particular, this allows to make sure that the last constraint of (3.7) is respected, i.e. that the final power of the RPR architecture is smaller than that of the single MDSP at nominal voltage. Power analysis is performed using *switching activity* information, collected during the previous simulation.

Another set of simulations is then run in order to determine the outputs produced by different replica configurations for the provided simulation inputs (step ②). In doing this, we exploit the fact that in a fixed-point datapath circuit, removing some LSBs from an input bus is functionally equivalent to setting them to logic-0. Consequently, we can obtain the same outputs that would be produced by a reduced bit-width replica, simply by forcing the corresponding number of LSBs to zero in the simulation. This does not require the availability of the actual replica netlist, and can be done directly on the MDSP.

During step ② we also make the optimistic assumption that in the final design, the considered replica implementation will be timing compliant. Because of this assumption, we can run *functional* simulations in this phase, without the need of accurate timing information, which greatly reduces the execution time. Moreover, we can execute a single set of replica simulations, rather than repeating them for all V_{vos} voltages. In the following optimization phase, this assumption will be checked for each combination of replica bit-width and VOS supply voltage; if it turns out to be false, the corresponding combination will be discarded.

Simulation in ② are performed in decreasing order of replica overheads. This is achieved by progressively zeroing out some bits of each input bus, in increasing bit-weight order. Although precise power and area information are not yet available, it is safe to assume that at a given $V_{vos,i}$, a larger number of removed bits corresponds to a smaller overhead.

When golden outputs and replica outputs for all configurations have been gathered, the decision threshold T_h' for each output bus, and for each version of the replica is computed as:

$$T_h' = \max_{\text{all input vectors}} |y_{M,0}[n] - y_R[n]| \leq T_h \tag{3.9}$$

This expression is in accordance to (2.3), where in this case $y_{M,0}$ represents the set of MDSP golden outputs obtained in ① and y_R represents the set of replica outputs obtained in ②.

The empirical threshold T_h' is in generally smaller than the theoretical one, since in most cases, the testbench provided by the user does not cover all possible input combinations. Therefore, for some combination of inputs not present in the simulated vectors, the decision block may select the replica output even though the

MDSP is correct. However, if the test stimuli are representative of real usage, this occurrence will be rare. Furthermore, using simulations to determine T_h' does not require knowledge of the MDSP functionality.

Optimization Phase

In the optimization phase, the tool iterates over the N standard-cell library characterizations in VOS (step ①). For each $V_{vos,i}$, the MDSP first undergoes a timing analysis, and the computed propagation delays are annotated in a *Standard Delay Format* (SDF) file. The SDF is then used to execute a back-annotated timing accurate simulation of the MDSP in VOS conditions (step ②). This simulation may produce erroneous outputs due to the activation of paths that violate the timing constraints and are sampled wrongly by memory elements (flip-flops or latches). Given that a realistic input sequence is used, the error rate and magnitude are estimated accurately, also accounting for temporal correlation.

With erroneous MDSP outputs available, our tool computes the minimum replica configuration that satisfies the quality constraints (step ③). To do so, it iterates over replicas in descending order of bit-width. For each bit-width, the outputs of a hypothetical RPR architecture are generated merging the erroneous MDSP outputs from ② and the replica outputs gathered in the preliminary phase, according to (3.3) and to the previously computed threshold T_h'.

The sequence of outputs obtained from this merging is then fed to the quality evaluation module, where it is compared with the error-free version of the MDSP outputs. If the quality constraints are satisfied for all output buses, the evaluation is repeated with a smaller replica, until a configuration that violates at least one constraint is found.

When the minimum size replica configuration has been identified, the tool proceeds to generate its netlist (step ④). To this end, it leverages the netlist simplification directives embedded in all commercial logic synthesis tools (e.g. `remove_net` in Synopsys Design Compiler). With these commands, the desired number of input bits is removed from the MDSP netlist. Then, an incremental re-synthesis is performed in order to propagate these simplifications, so that the tool can automatically eliminate all cones of logic that have become useless. Importantly, this re-synthesis is performed under the currently considered $V_{vos,i}$. Therefore, the synthesis tool also performs all the necessary optimizations (e.g., gate resizing, remapping, etc.) to make the replica timing compliant at the target supply voltage. As previously mentioned, this is fundamental to correctly evaluate the power and area overheads of RPR. In fact, to be error-free, the replica may contain much larger and power consuming gates than those that compose the MDSP.

Once the replica is generated, the rest of the EC block must be synthesized. In particular, each output bus requires a proper *decision block*. The latter has a fixed structure, composed of a subtractor, a comparator and a multiplexer, as shown in Figure 2.6. The only variable elements of its architecture are the input bit-widths and the hardwired threshold T_h'. Therefore, our tool simply sets these parameters within a generic HDL model before synthesizing the decision block at $V_{vos,i}$.

The final step performed by the tool is the estimation of the total power and area costs of the RPR configuration (step ⑤). In this phase, the tool also checks the actual timing compliance of the replica and of the decision block, necessary for the correct behavior of the entire architecture. This additional check is needed because the previous synthesis may sometimes terminate despite having generated a circuit with negative slack (e.g. when the maximum number of optimization iterations has been reached). The considered configuration is considered feasible if the total power (including dynamic and leakage contributions) is less than the power of the MDSP at nominal voltage, and if the EC block meets timing.

The entire loop of the optimization phase is then repeated considering the next VOS voltage.

The flow described above represents the *normal mode* of operation of our tool. However, a *full mode* of operation is also available, in which the complete tradeoff between power and quality is explored. In this mode, rather than just extracting the minimum replica size for each $V_{vos,i}$, all configurations that meet the quality constraint are synthesized and analyzed. Clearly, with respect to the normal operation, this mode greatly increases the execution time; however, it is sometimes useful to perform preliminary experiments (e.g. when the desired quality level is not yet defined). Finally, our tool also allows to set a hard constraint on the maximum error rate allowed for the RPR architecture.

3.2.5 Experimental Results

Setup

We have validated the tool described in Section 3.2.4 with experiments on a set of digital datapath IPs publicly available on OpenCores [155]. We have synthesized the original RTL models with Synopsys Design Compiler F-2011.09, targeting a commercial 45nm standard-cell library from ST Microelectronics. For all designs, we have set the nominal (error-free) supply voltage to $V_{nom} = 1.1V$, and we have selected a synthesis frequency that produces a critical positive slack of less than $100ps$ in that condition. We have characterized the standard cells library for timing and power in a set of VOS points going from $1.05V$ to $0.50V$ in steps of $0.05V$. The flow depicted in Figure 3.1 has been integrated also with Synopsys PrimeTime F-2011.12 for timing and power analyses and with Mentor Graphics ModelSim SE 6.4a for simulations.

We have selected the following five designs for our experiments:

- A pipelined 16th-order low-pass *Finite Impulse Response* (FIR) filter in direct form (12-bit in, 24-bit out) [156]
- A pipelined 4th-order low-pass *Infinite Impulse Response* (IIR) filter in direct form I, modeled after a Butterworth analog filter (16-bit in, 32-bit out) [156]
- A “butterfly” unit employed within an accelerator for *Decimation-In-Time Fast Fourier Transform* (DIT-FFT) [156]
- A rotation mode (RM) pipelined *Coordinate Rotation Digital Computer* (CORDIC) accelerator (16-bit in, 16-bit out) [157]
- A fixed-point 32-bit *Square Root Unit* (SRU) implementing the Goldschmidt recursive equation [32].

FIR and butterfly have been selected to compare our results with those of [28], since those two benchmarks were also analyzed using the classic RPR formulation. The remaining circuits serve to show the generality of our algorithm, and to discuss the feasibility of RPR in relation to the characteristics of a given hardware block.

For each circuit, we have generated a testbench in VHDL that stimulates it with realistic inputs extracted from a suitable error-resilient application. Depending on the nature of inputs, we have adopted two different quality metrics. For communication applications, quality has been evaluated using the *SNR*, defined as in (3.5), while for arithmetic units we have used the inverse of the *MSE*, or *PSNR*:

$$Q_{mse} = \frac{n}{\sum_{i=0}^{n}(\overline{y}[n] - \overline{y_{M,0}}[n])^2} \tag{3.10}$$

where $\overline{y}$ and $\overline{y_{M,0}}$ are y and $y_{M,0}$ normalized to the maximum value expressible on each output bus, in order to meaningfully compare errors on buses with different bit widths [1]. The discriminant for the usage of one quality metric or the other is the presence of external noise. By combining the effects of timing-errors with a pre-existent source of inaccuracy (noise), the *SNR* is effective in showing how RPR does not affect the overall quality of processing in communication systems, if the replica is sufficiently large. The *PSNR* instead, is suitable to consider the impact of VOS-induced errors alone, for systems in which no information on input data noise is available.

All reported results on power savings and area overheads are normalized to the metrics of the MDSP in nominal conditions.

Analysis of Input Data Dependence

One feature of our methodology that was not present in prior art on the subject is the possibility of taking into account the characteristics of inputs (value distribution and temporal correlation) when implementing a RPR architecture. In order to show the importance of this feature, we have analyzed the impact of different input datasets on the RPR power savings for two circuits.

As a first motivating example, we have applied RPR to a simple fixed-point signed array multiplier, with 16-bit inputs and 32-bit outputs. *PSNR* has been used as the quality metric, setting an arbitrary constraint $Q_{mse} \geq 60\ dB$ (i.e. $MSE \leq 10^{-6}$).

We have executed our flow four times on this design, in identical conditions, except for the sequence of input vectors applied to the circuit by the testbench. In particular, we have generated two sets of 5000 inputs: in the first, both operands of the multiplier are uniformly distributed in the range $[-2^{15} : 2^{15} - 1]$, while in the

[1] We use the subscript *mse* for indicating quality expressed by the *PSNR* to avoid confusion with the *SNR*.

second, they follow a Gaussian distribution with mean $\mu = 0$ and standard deviation $\sigma = 2^{10}$. Moreover, we have repeated the two experiments after sorting the stimuli in ascending order, thus enforcing a strong temporal correlation among subsequent vectors. The maximum error rate has been set to 0.2, which, for all inputs, has led the tool to discard configurations with $V_{vos} < 0.7V$.

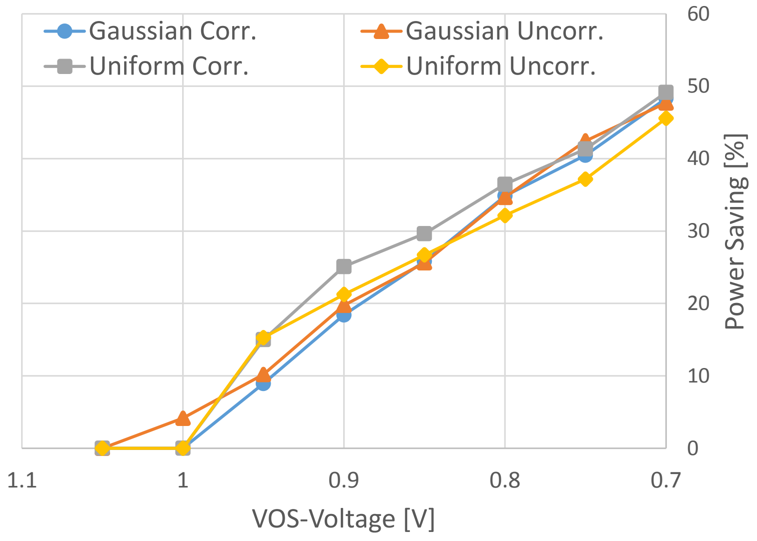
Figure 3.3a shows the results produced by our tool on a power saving versus voltage plane. Power saving follows a similar monotonically ascending trend for all data sets. However, the exact power reduction achieved varies significantly as a function of the inputs: the maximum difference between the most and least favorable input sets is around 8% (at $V_{vos} = 0.9V$). Furthermore, the relation among input set and power saved is not trivial, and there is no evident way in which it could be modeled analytically.

These results are expected, and can be motivated considering the working principles of RPR. Indeed, different inputs stimulate the internal paths of the multiplier differently. Because delay faults depend on *pairs of stimuli*, both the distribution of values and their temporal sequence are relevant. Consequently, the percentage of violating paths at each V_{vos} (hence the error rate) changes for different data sets. Due to this difference, our tool will often find that, at a given voltage, the minimum required quality is obtained using *different replica bit-widths* for different stimuli. Clearly, the smaller the replica, the smaller the associated leakage and dynamic power overheads, and the larger the achieved power savings.

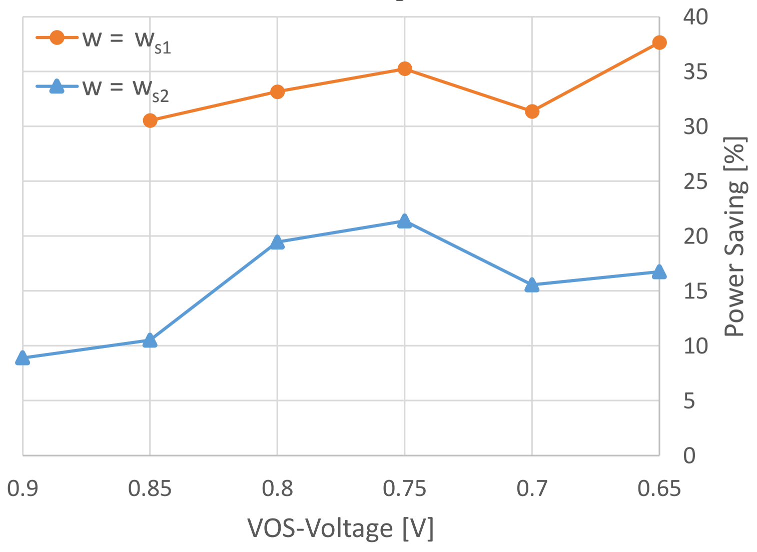
Secondarily, the sequence of input vectors also directly affects the *switching activity* of the gates both in the MDSP and in the replica, which in turn influences the dynamic power consumption of the entire circuit.

As an example, at $V_{vos} = 1V$ the correlated Gaussian inputs produce an error rate of $1.5 \cdot 10^{-3}$ while the error rate for uncorrelated data is only $0.4 \cdot 10^{-3}$. However, the total power consumption *of the MDSP* at this voltage is $0.6mW$ for the correlated inputs, and $1.7mW$ for the uncorrelated ones, due to internal switching activity differences. Overall, the combination of these two effects is very difficult to model analytically, whereas it is automatically accounted for by our simulation-based approach.

The impact of inputs on the effectiveness of RPR is even more evident for a larger circuit, such as the FIR. The results of an experiment similar to the one described above on this benchmark are reported in Figure 3.3b. In this case, we



(a) Multiplier.



(b) FIR filter.

Fig. 3.3 Examples of RPR power savings versus V_{DD} for different input sets.

have fed the FIR with two sets of full-swing sinusoidal samples, with frequencies of $w_{s1} = 0.01\pi rad/s$ and $w_{s2} = 0.1\pi rad/s$ respectively, both affected by *Additive White Gaussian Noise* (AWGN) with a *SNR* of 25dB. The cutoff frequency of the filter has been set to $w_c = w_{s2}$. Notice that, in this case, the two sets of stimuli have the same distribution of values; only the rate of variation over time (hence the temporal correlation) differs. We have instructed the tool to maintain the effect of VOS-induced errors on output quality comparable to that of input noise, by setting the quality constraint to $Q_{snr} \geq 25dB$. We have also set the maximum error rate to be 0.2.

In Figure 3.3b, we only report results starting from $V_{vos} = 0.9V$ because for higher voltages, even though the critical path slack is negative, no input pair generates a timing violation. This means that the critical path is never actually activated by the considered inputs, a condition that, by the way, can only be verified by a simulation-based analysis. When applying the lower frequency input (w_{s1}), the circuit is still error-free also at $0.9V$.

Importantly, in the case of the FIR, power savings do not increase monotonically when voltage is decreased. This phenomenon is typical for complex designs, especially in deep VOS conditions, and is due to the optimizations performed during the automatic synthesis of the replica. In fact, in order to make the replica timing compliant despite the lowered supply voltage, our tool must optimize its netlist until a positive slack is obtained (by modifying gate sizes etc.), thus increasing the associated power overheads. This contribution may be larger than the power reduction in the MDSP with respect to the previous VOS point, causing an increase in the total consumption of the RPR architecture. When voltage is reduced further, if the error rate does not increase significantly, allowing to maintain a similar replica size, total savings may start to increase again.

The main difference between this experiment and the previous one on the multiplier, however, is the clear influence of input/output transfer function of the MDSP on the RPR output quality. In fact, since the FIR is configured as a low-pass filter, the attenuation introduced on the input signal is different at w_{s1} and w_{s2}. This causes a reduction in signal power for the higher frequency waveform, which translates in a significant *SNR* difference, even under the same conditions of input noise and rate of VOS-induced errors. Consequently, a significantly larger replica ought to be used when targeting the higher frequency waveform, in order to maintain the desired quality level. This dependency adds up with the influence of vectors sequences on activated paths, resulting in a power saving difference between the two inputs sets larger than 20%.

These two examples clearly highlight that the assumption of uniformly distributed inputs can lead to sub-optimal RPR architectures (either overly optimistic or pessimistic depending on the considered circuit). They also show that considering the sole distribution of values is not sufficient to overcome this issue, and that accurate simulations, accounting also for input temporal correlation, can provide much more realistic results.

Flexibility of the Approach and Comparison with Ad-Hoc Solutions

For each of the five target benchmarks we have identified an error-resilient application that includes it, and profiled such application to generate the input stimuli required in the simulations performed by our tool.

For the FIR filter and FFT-butterfly, we have considered the same applications examined in [28], i.e. the low-pass filtering at the receiver of a QPSK communication link, and the modulation/demodulation in *Orthogonal Frequency Division Multiplexing* (OFDM) WLAN respectively. We have also assumed the same channel SNR conditions ($21.5dB$ for the FIR and $55dB$ for the butterfly). As above, we have set the quality constraints of the tool to the same SNR values, in order to make VOS-induced errors comparable to input noise. To generate the signal component, we have used a MATLAB model of the IEEE 802.11g WiFi standard, from which we have extracted the input data for the two circuits. We have then converted these data to the appropriate binary format and added AWGN. The produced sets of vectors has been loaded in the VHDL testbench passed as input to the automation tool.

RM-CORDIC can be used in place of the classical butterfly unit in FFT accelerators [158]. Similarly, the IIR filter can replace the FIR in a QPSK receiver. Therefore, we have decided to test these two new circuits for the same applications of the previous two, and under the same quality constraints. This shows how our tool can be used to determine empirically the most suitable architectures for a RPR-based realization of a given task.

Finally, SRUs are used in many error-resilient applications, from FFT magnitude evaluation to SVD-based image compression. For our experiments, we have targeted the *Automatic Gain Control* (AGC) task present in many DSP systems [156]. We have generated the inputs to the SRU from a set of base-band audio samples. In this case, in absence of precise data on noise, we have used the *PSNR* quality metric, setting the maximum acceptable *MSE* to 10^{-12}. Table 3.1 summarizes the obtained results for all five benchmarks.

Circuit	Net Count	f_{clk} [MHz]	Optimal V_{vos} [V]	Replica / MDSP Bits	Tot. Power Saving [%]	Area Ovr. [%]
FIR	3379	330	0.60	6/12	44.96	82.39
Butterfly	3840	200	0.55	13/16	49.66	133.20
IIR	3556	250	0.50	9/16	8.82	181.82
CORDIC	4939	400	0.55	11/16	42.05	127.64
SRU	21267	125	0.75	17/32	47.91	143.32

Table 3.1 Summary of the RPR automation tool results for five benchmarks.

The large power saving values confirm the generality of the tool, since almost 50% reduction in total power can be obtained for 4 of the 5 benchmarks, with the notable exception of the IIR filter, which is however due to intrinsic features of the RPR approach (explained later).

The main drawback of RPR is the large area overhead introduced by the EC block (replica and decision circuitry). For all of our benchmarks, except the FIR filter, this overhead is larger than 100%, i.e. the area of the circuit is more than *doubled*. This is mainly due to the gate resizing performed by Design Compiler during the replica synthesis, whose impact is made more relevant by our application-agnostic approach. In fact, since we create the replica from the MDSP netlist, using Design Compiler directives, we only allow *logic-level* optimizations to be performed on the reduced unit. DC cannot perform higher-level optimizations such as the instantiation of different DesignWare macro-blocks. As an example, consider that under the same synthesis constraints, an 8x8-bit multiplier generated removing inputs from a 16x16 netlist is $1.3x$ larger than one generated directly from behavioral VHDL, in our target technology node.

One way to reduce these overheads would be to limit the applicability of our methodology only to RTL-level designs defined *parametrically*, in which the input bit-width is set through a generic construct of the HDL. This would allow to generate the replica with a full synthesis run, simply by changing the value assigned to the generic, thus enabling all kinds of optimizations by the synthesis tool. However, doing so would strongly affect the generality of our tool. Given that generality was one of our main objectives, we have decided to keep our code as flexible as possible, and to accept the additional overheads that derive from this choice.

Importantly, the power savings obtained for the FIR and butterfly with our method are comparable to those reported in [28]. In particular, [28] achieves 60% power reduction for the filter (versus our 44.96%) and 44% (versus 49.6%) for the FFT block. Differences are partly related to the different testing conditions (especially the technological node), partly to the different timing and power models adopted. For instance, the leakage power overhead is much more significant at 45nm than at 250nm. However, the main contributor to these discrepancies might be the fact that we take into account input sequences in our optimization, obtaining more realistic estimates of the real timing and power behavior, as demonstrated in Figure 3.3. In summary, the comparison with a classic RPR approach clearly shows the competitiveness of our methodology.

We have also profiled the execution of the RPR automation tool, in order to assess its complexity, and its applicability to real-life scenarios. Rather than absolute values, which depend on the platform on which the code is run, we have collected the *ratio* between the execution time of our tool and that of a normal design and verification flow (i.e. a single sequence of synthesis, timing annotation, timing simulation and power estimation). This metric is in first approximation independent from the platform, as well as from the circuit complexity and input stimuli length; it only depends on the number of VOS voltage points and on the number of feasible solutions found by the tool. The result of this profiling is that the execution time of the RPR automation tool for the five benchmark circuits is between 6x and 9x longer than that of a standard flow. If needed, this overhead could be easily reduced by selecting a smaller number of VOS points. Moreover, notice that the time used as reference does not consider any power optimization. In practice, many low-power design flows often require several iterations to converge to an optimal implementation, and differently from our tool, may also require the manual intervention of the designer for tuning parameters. In summary, although the iterative nature of the optimization performed in our methodology clearly requires a not negligible execution time, we believe that this overhead is acceptable when balanced with the convenience of the “push-button approach” offered by our tool.

Impact of Architectural Features

The IIR benchmark deserves a separate analysis. This circuit has a significant difference with respect to all others, that is the presence of a *feedback network* from the outputs [156]. This is clearly an architectural drawback from the point of view of RPR. In fact, when a timing error is generated in the IIR, it remains in the system for more than a single clock cycle due to the feedback network, worsening the output quality. In other words, even if the rate of timing violations is low, the error rate becomes extremely large. Since the “memory” of the IIR is limited in time by the finite data precision, the error gradually reduces in magnitude and eventually becomes comparable to the input noise. This is the reason why our tool is still able to produce feasible solutions. However, due to the negative effect of feedback on the outputs in presence of timing errors, a very large replica is needed to respect the quality constraints, making the total savings achieved for the IIR significantly smaller than those of all other circuits. Finally, the presence of feedback also limits the simplifications that can be performed by Design Compiler when inputs are removed during the replica generation, which also increases the area overheads.

The IIR example highlights some of the characteristics that a circuit should possess to be an ideal candidate for RPR. However, results show that our tool is able to deal also with unfavorable input netlists, generating all feasible solutions. The assessment of the convenience of such solutions is clearly left to the designer.

Quality versus Power tradeoff

As a last result, we present in Figure 3.4 an example of the output produced by our tool when executed in *full mode*, for the FIR filter benchmark. To generate this graph, we have set a very low quality constraint (0 *dB*), and no error rate constraint. In practice, this forces the tool to produce the entire set of feasible configurations for each V_{vos} point in which the error rate is not 0. Input vectors and noise parameters are the same used in previous experiment.

Two trends can be identified in the graph: on the left side (correspondent to large replica implementations), the effects of input noise dominate the *SNR*, and the

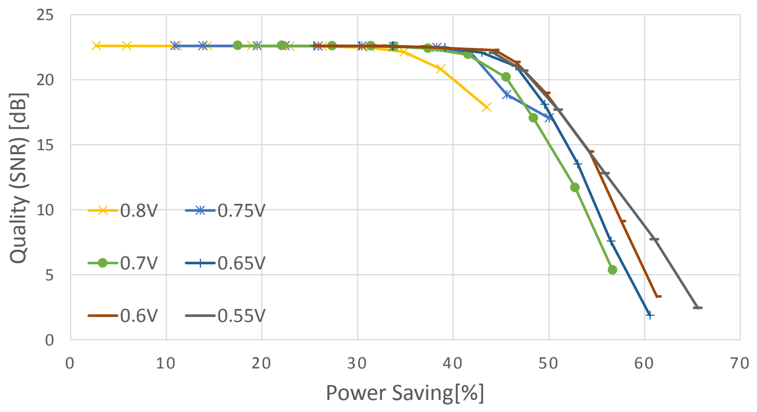


Fig. 3.4 Quality versus power tradeoff curve generated by the RPR automation tool in full mode for the FIR Filter.

quality saturates at the 21.5 dB limit; as soon as the RPR effects start to become relevant due to the reduction in replica size, the tradeoff between power and quality becomes evident. Note that different curves have different numbers of points. This happens because the tool still eliminates those replica implementations that either produce a negative power saving, or cannot be synthesized with positive slack at the given V_{vos}.

3.3 EDA Flow for Precision-Configurable Operators Based on Combined Voltage Scaling and Adaptive Back Biasing

DAS and its variants are design techniques for quality-configurable datapath units that rely mostly on technological knobs, with minimal architectural modifications [34, 48, 78]. The general background on this family of approaches has been presented in Section 2.1.1. As mentioned in that context, DAS-based approaches have been proven superior to most of the other techniques for the design of quality-configurable hardware, both in terms of sheer power savings, as well as in terms of associated overheads and generality of application [34, 77]. Among the different variants, DVAS achieves the best balance between effectiveness (its savings are significantly larger than those of DAS thanks to voltage scaling) and generality (differently from DVAFS, it does not require architectural modifications to support subword parallel operation, and it allows a finer granularity of precision configurations).

However, similarly to the case of RPR, the theoretically sound principles that motivate DVAS-based design do not match the characteristics of real implementations, which are optimized using standard EDA flows. In particular, the assumption that gating some LSBs produces a reduction in the critical delay of an operator, thus allowing voltage scaling, is not valid for most real circuits. In contrast, the most common scenario is to have a large number of timing paths whose delay is close to the critical one. In that case, even if a large number of bits is gated, the critical delay is likely to remain very close to the nominal one. This phenomenon is known as the *wall-of-slack*, and will be analyzed in greater depth in Section 3.3.1.

This limitation can be solved either by modifying the synthesis process, or by means of a fine-grain delay and power consumption tuning of the circuit internals at runtime, through an additional technological knob. The second option allows to reshape the distribution of timing paths within the operator, depending on the selected precision configuration. Because of this reshaping, as shown in the following, the power saving opportunities offered by input gating can be fully exploited, even in a circuit synthesized with a standard EDA flow. This is the solution adopted in this section. In Section 3.4, we propose a flow that modifies the synthesis process to achieve a similar result, and we also compare the two options.

In particular, the technique described in this section combines DVAS with dynamic threshold voltage (V_{th}) scaling. By acting on the V_{th} of standard cells within a quality-configurable operator at fine grain, our method selectively speeds up the timing-critical regions of the hardware, thus increasing the usable dynamic when the supply voltage is scaled. In principle, our solution can work with any technological knob for dynamic V_{th} tuning. However, in this work, we demonstrate it on state-of-the-art *Fully-Depleted Silicon On Insulator* (FDSOI) technology using dynamic back-gate biasing [159, 160].

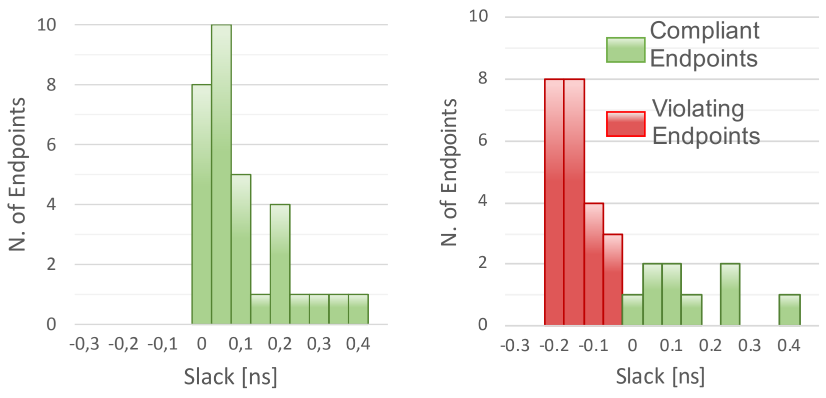
Because of the much finer spatial granularity at which the tuning of V_{th} can be applied, compared to dynamic voltage scaling, our method allows to overcome the main limitations of DVAS. Our proposed flow is fully automated and integrated with state-of-the-art tools, and yields a power reduction of about 40% with respect to standard DVAS.

3.3.1 Limitations in Classic DVAS Design

As discussed above, DVAS can be regarded as one of the state-of-the-art techniques for the implementation of energy-quality scalable hardware operators. However, DVAS incurs in a major issue when implemented in combination with a standard EDA flow.

As a matter of fact, synthesis and *Place and Route* (P&R) tools normally optimize the most timing-critical paths of a circuit for performance, whereas less critical paths are exploited for area and power recovery. Therefore, gates belonging to short paths are mapped to smaller and less power hungry standard cells, which are invariably also *slower*. This produces a reduction in area and power at the expense of an increase in delay [161]. Because of this optimization, the delay of non-critical paths tends to increase, to the point where their slack is comparable to that of critical ones, causing the wall-of-slack phenomenon [83].

An example is shown in Figure 3.5a, which reports the slack histogram of the endpoints of a 16x16-bit multiplier. The histogram is obtained after P&R, at the nominal implementation supply voltage ($1V$).



(a) $V_{DD} = 1.0V$ (b) $V_{DD} = 0.8V$

Fig. 3.5 Endpoint slack histogram for a 16x16-bit multiplier implemented in 28nm UTBB FDSOI technology, with a target clock frequency $f_{clk} = 1.25GHz$.

The effect of the wall-of-slack on DVAS is that the input bit-width usable without incurring in timing violations decreases rapidly when V_{DD} is scaled. As an example, Fig. 3.5b shows the multiplier endpoint slack histogram, after the supply voltage has been downscaled to $0.8V$. Red bars correspond to endpoints violating timing constraints. Such a large amount of violating paths indicates that, in order to restore timing compliance, the input bit-width should be decreased drastically, even for such a limited V_{DD} reduction. Reversing the perspective, in order to work at high-precisions with DVAS, V_{DD} cannot be scaled more than few tens of mV.

Another limitation of DVAS occurs when multiple operators within the same integrated circuit must work with independently tunable precisions. In such case, each operator must receive its own supply voltage, and therefore it must be placed in a dedicated *voltage domain*. In MOS technology, voltage domains are separated inserting level shifters, which introduce significant power overheads, as detailed in the next section.

3.3.2 Fine-Grain Speed/Power Control Using Back Bias

Using supply voltage (V_{DD}) scaling as a knob to regulate power, in response to a precision reduction, has some technological downsides. First, V_{DD} Scaling has a relatively coarse granularity and cannot be applied to arbitrarily small regions, due to the power and area overheads of level shifters: practical voltage domains have sizes in the order of hundreds of standard cell rows [162–164]. Second, the supply voltage cannot be tuned in arbitrarily small increments; indeed, to contain generator costs, V_{DD} is normally selected among a pool of few (≤ 10) pre-generated voltages, using the so-called V_{DD} hopping [164].

A finer-grain knob that can be exploited to tune the power/precision operating point is the regulation of the threshold voltage V_{th}. At runtime, this can be achieved with adaptive *Body Biasing* (BB), that is by modifying the voltage applied to the body contact of a MOSFET transistor. Detailed analyses of the relations between BB and the delay and power consumption of a MOSFETs can be found in [165, 166].

In classical bulk CMOS, adaptive BB has limited effectiveness, since the applicable voltage ranges are limited to $\approx \pm 300mV$ by the presence of parasitic P/N junctions between the body and source terminals of a transistor. Consequently, the range of variation of V_{th} is also limited, and transistors power and delay are only moderately affected.

Conversely, runtime V_{th} tuning is much more effective in advanced technologies like FDSOI. Specifically, in the *Ultra-Thin Body and Box* (UTBB) FDSOI technology used in this work [159], the applicable BB voltage range is approximately $\pm 2V$, and the *body factor* (i.e., the sensitivity of V_{th} to BB) is as high as $85mV/V$ [160].

The larger range of applicable biasing is due the presence of a *Buried Oxide* (BOX) layer, which removes body-source P/N junctions and acts as a back-gate. For this reason, BB is referred to as *back-gate biasing*, or simply *back biasing* in the context of FDSOI technology.

Such a large BB voltage range can be leveraged for a very effective dynamic tuning of the performance/power tradeoff. Specifically, *Forward Back Biasing* (FBB) lowers the V_{th} and reduces the switching delay of a transistor, at the expense of an increase in sub-threshold leakage currents. *Reverse Back Biasing* (RBB) has the opposite effect. In literature, adaptive body/back biasing has been widely used in combination with *Dynamic Voltage and Frequency Scaling* (DVFS) for power management, to improve the Pareto frontier of power/performance points [160, 167, 166]. However, to the best of our knowledge, BB has never been used before as a knob for implementing energy-quality scalable systems.

With respect to V_{DD} scaling, adaptive BB can be implemented at much finer granularity, since no level-shifters are required. However, there are still technological constraints that prevent its application to single devices. As a matter of fact, gates with different BB voltages must be separated by *guardbands* that have a size comparable to that of a standard cell [160]. Further technological details about the specific technology used in our work will be given in Section 3.3.5.

3.3.3 Fine-Grain Back Bias in Precision-Configurable Datapath Operators

The methodology proposed in this manuscript uses the same approach of DVAS to generate multiple quality configurations in a circuit: the latter are obtained by varying the operator *precision*, which is achieved by simply zeroing some of the input LSBs. However, our method employs fine-grain tuning of speed and power to contrast the wall-of-slack problem of DVAS.

To motivate our choices, we first describe the situation in which this kind of fine-grain tuning is realized using multiple V_{DD}s within the same operator. We then explain how the same result can be achieved with V_{th} tuning (via back biasing), albeit with smaller overheads.

In a reduced-precision operator, many timing paths are *disabled* as a consequence of zeroing the LSBs, and therefore they are irrelevant for timing. These, plus the few paths that are not critical despite the wall-of-slack, could theoretically receive a smaller V_{DD}, while the rest of the circuit is supplied with a "conservative" (higher) V_{DD} to prevent violations. This subdivision of paths is exemplified in Figure 3.6. Notice that the membership of a path to one of these three groups depends on the selected supply voltage, as well as on the number of gated LSBs.

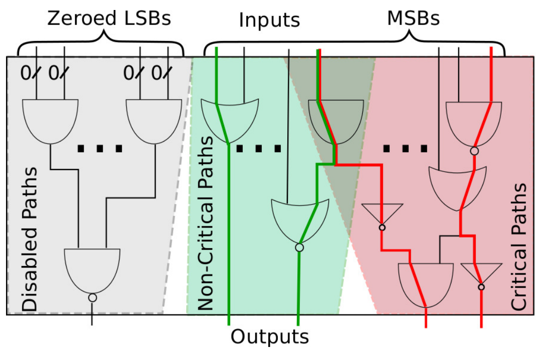


Fig. 3.6 Conceptual subdivision of the timing paths within an operator working at reduced bit-width. Some of the paths in each group are highlighted.

Using multiple V_{DD}s within an operator would permit significant power benefits, despite the fact that the critical delay of the design is unchanged at reduced precision (due to the wall-of-slack). In fact, disabled and non critical gates would consume much less than critical ones, in terms of both dynamic and leakage power, thanks to the lowered supply voltage.

Unfortunately, such a partitioning of the circuit in multiple V_{DD} domains is only feasible at the SoC-level, and not at the fine granularity required for the relatively small hardware operators considered in this work; in particular, the insertion of level shifters between domains would have a relevant impact on power consumption [162]. Therefore the "conservative" value of V_{DD} must be applied to the *entire circuit*, resulting in less power savings than what could potentially be achieved.

The speed and power knob offered by dynamic V_{th} tuning can be leveraged to solve this issue, offering an additional degree of freedom. For example, the entire operator can receive a *lower* V_{DD}, thus reducing power in the disabled and non critical parts. Then, *only the gates belonging to critical paths* can be speed up by lowering the threshold voltage (e.g. through FBB), in order to recover the timing violations. The main difference with the multi-V_{DD} approach is that, as anticipated, V_{th} tuning by means of adaptive BB can be applied at a much finer granularity.

Indeed, as explained in Section 3.3.2, topologically close cells in a circuit must still be grouped into areas with a common body/back biasing control, in order to contain area overheads associated with guardbands. However, given the typical dimensions of guardbands, the minimum acceptable size of one of these regions is only in the order of few tens of rows. Moreover, no level shifters are needed to separate one region from its neighbors. Consequently, while fine-grain V_{th} tuning still incurs some area overhead, it has a much smaller power cost compared to using multiple V_{DD}s, which is what enables its application at finer granularity.

In the rest of this manuscript, we will refer to groups of cells that receive the same body/back biasing with the term *V_{th} domains*, or equivalently *BB domains*.

3.3.4 Overview of the Proposed Methodology

A conceptual depiction of our methodology is shown in Figure 3.7. We propose to split an operator in multiple V_{th} domains, each with an independent back bias control, whereas the entire circuit shares a single V_{DD}. The additional fine-grain back biasing knob is leveraged to selectively speedup the (precision dependent) timing critical paths of the circuit, by applying a lower V_{th} to a subset of domains (shown as shaded red areas in the figure).

At a given V_{DD}, thanks to back biasing, the circuit will therefore be able to work at a higher precision compared to DVAS, without incurring timing violations. Consequently, it will also achieve higher power efficiency for most precisions.

To better highlight this advantage, Figure 3.7 reports the Pareto frontiers obtainable with DVAS (i.e. without fine-tuning) and with our method. These frontiers are constructed connecting the minimum power points for each bit-width, considering all possible combinations of knobs available in the two solutions (DVAS and ours).

As an example, in the figure DVAS allows using the first lowered supply voltage below the nominal one (i.e. $V_{DD,2}$) only for a precision of 7-bit or lower. Higher precisions cause timing violations at reduced V_{DD} due to the wall-of-slack effect. In contrast, the selective speed-up offered by BB allows the design to be timing compliant at $V_{DD,2}$ already at 14-bit.

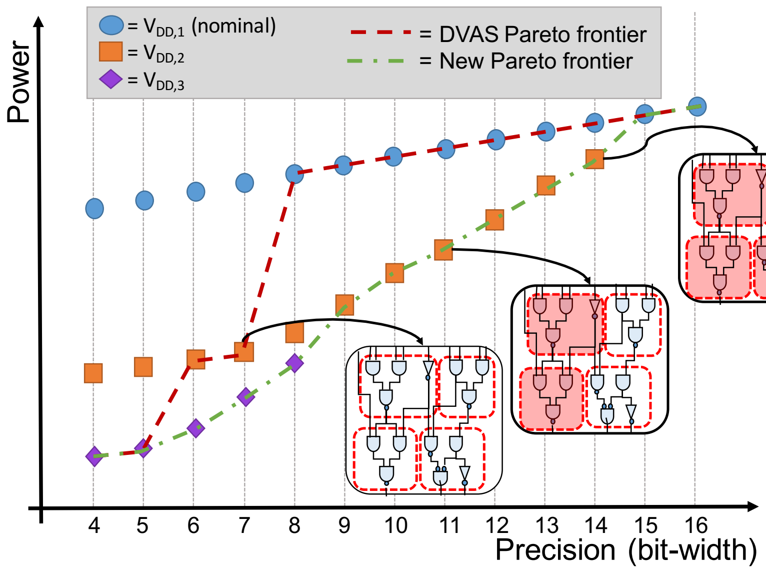


Fig. 3.7 Conceptual scheme of the proposed methodology for fine-grain speed and power control of a precision-configurable operator by means of adaptive V_{th} tuning.

In this work, we assume that each domain can be assigned one out of two possible V_{th} values: *Standard* V_{th} (SVT), which is considered the nominal condition, and *Low* V_{th} (LVT), which is the "boosting" (faster and more leaky) condition. In 28nm FDSOI with flip-well devices [160], we map SVT to *No Back Bias* (NoBB) and LVT to *Forward Back Bias* (FBB).

Limiting the choice to two V_{th} values has several advantages with respect to more complex assignments. First, it simplifies the search of the optimal V_{th} assignment to domains, for a given precision. Second, it makes the generation of back bias voltages easier, as we will show in Section 3.3.5. Nonetheless, our methodology can also be applied to more than two V_{th} values, with increased flexibility at the expense of higher complexity. Similarly, it can also be implemented using other forms of V_{th} tuning (e.g., body biasing in bulk CMOS).

It is worth re-emphasizing that our approach is not alternative but rather *complementary to DVAS*, since we still apply voltage scaling to the entire design, and then use BB to fine-tune the consumption and speed of different parts of the circuit. Indeed, the effect of our method is to make DVAS compatible with the optimizations performed by standard synthesis tools, which cause the wall-of-slack.

The same is true also for DVAFS, which benefits from a reduction of the wall-of-slack as well, and can be implemented *on top* of our technique. Of course, the effect of DVAFS on timing paths slightly differs from the one discussed in Section 3.3.3. In particular, disabled paths are not simply those driven by a zeroed input bit, but rather those that are never activated due to the "split" of the operator into two sub-circuits [48].

The Circuit Partitioning Problem

The use of back biasing for V_{th} tuning brings about new design issues. First of all, we need to determine the maximum number N of V_{th} domains that can be inserted in a given operator. This number depends on a user-defined acceptable area overhead, due to separation guardbands. Even once N is defined, however, there are still many degrees of freedom available regarding the possible shapes of the V_{th} domains. The definition of these regions is a fundamental issue in the proposed methodology.

The basic problem of partitioning a circuit into multiple regions for selective application of a knob (V_{th} tuning, in our case) to control some design tradeoffs (power/precision, in our case) has been already studied the literature for various knobs and tradeoffs [168]. However, this particular instance, i.e. the application to precision-configurable designs, introduces several new elements of complexity.

The ideal goal of the partitioning process can be formulated as follows: *build a partitioning that ensures timing compliance with minimum power consumption, considering all relevant precision configurations (i.e. bit-widths) and V_{DD} values*. Here, a relevant configuration is one that will be actually used in applications (e.g. 1-bit precision might not be interesting).

This goal is achieved if the partitioning can isolate exactly the cells that define the critical paths (the red part of the circuit in Figure 3.6). This must be true for all relevant bit-widths at their corresponding optimal V_{DD}, i.e. the supply voltage that, after boosting critical cells with FBB, allows to reach timing compliance with minimum total power for that bit-width.

However, finding this optimal solution is in general impossible. In fact, the physical partitioning of the circuit in domains must be done at design time, and is *single* for a given operator. In contrast, each bit-width mode implies a different set of critical paths, and a partitioning that is optimal, (i.e., that allows to speed up *only* the critical cells) for a given precision, might not be optimal for others. The only solution that would always guarantee perfect isolation of critical cells for all bit-widths is the independent back biasing of each cell, which is technologically unfeasible.

Moreover, the partitioning process also impacts the other figures of merit of the circuit (timing, dynamic power consumption, etc.), due to the fact that V_{th} domains are physically isolated regions in the die. In the partitioned circuit, cells should be kept as close as possible to their optimal locations (i.e., those determined by a standard placement algorithm) in order to minimize this impact [162, 168].

Partitioning Strategies

The fundamental criterion used in this work to drive the circuit partitioning into BB domains is to keep the figures of merit of the precision-configurable operator as close as possible to those of a standard design, when the former is used at maximum precision. To this purpose, we only consider partitioning solutions in which cells are *minimally displaced* with respect to a standard P&R.

The default choice in this work consists of the simplest possible partitioning, i.e. a *regular tiling* of the operator into N BB domains. In this solution, each domain is assigned an identical area of rectangular shape, equal to a fraction $1/N$ of the total operator surface. Starting from the result of a standard P&R, cells are mapped to the closest domain in the newly created grid of tiles, as shown by the arrows of Figure 3.8 for the case $N = 4$. This solution has the desirable property of producing a regular structure, which eases the physical implementation [169]. Moreover, it is easy to automate and embed in a standard EDA flow.

Of course, the drawback of regular tiling is that it does not consider the spatial distribution of timing paths. Therefore, for a given precision and V_{DD} combination, each BB domain will generally include a mix of critical cells (highlighted in red in the figure for one particular combination) and non-critical/disabled cells. In order to meet timing constraints, the *entire* BB domain has to be assigned to LVT/FBB, resulting in a power overhead due to the unnecessary speed-up of non-critical and disabled cells. However, thanks to the relatively small granularity of BB domains, this overhead is normally smaller with respect to that of increasing V_{DD} in the entire circuit. Domains that must be boosted to “cover” all the critical cells in the example of Figure 3.8 are shown with a red-shaded background.

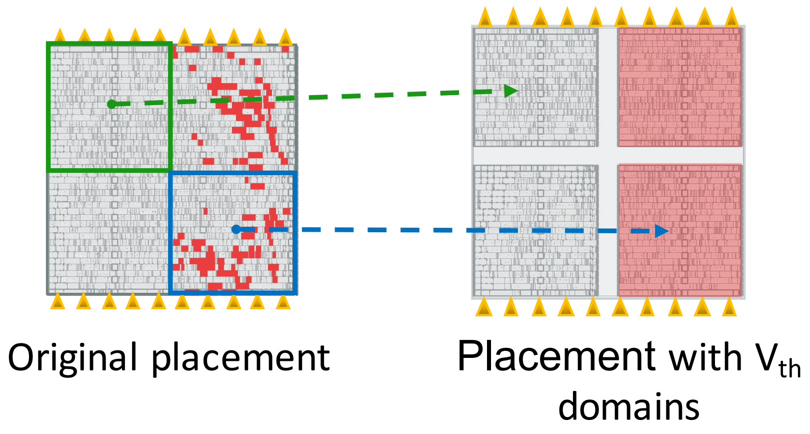


Fig. 3.8 Regular tiling partition of an operator into multiple V_{th} domains, and example of critical cells and boosted domains, for one particular V_{DD} and precision combination.

In general, the smaller the V_{th} domains, the more it is possible to identify a set of tiles to be boosted that approximately isolates the critical paths from the rest of the circuit. In fact, the smaller the tiles, the fewer cells are grouped together; at the limit, this reduces to the case in which each domain contains a single cell, which (although unfeasible due to area overheads), would allow perfect isolation of critical paths.

One benefit of a regular tiling is flexibility: since the partitioning is determined a priori, without a specific target precision or V_{DD}, it will perform reasonably well on average for all bit-width and V_{DD} values. Conversely, an irregular tiling that tries to better match the spatial distribution of the critical cells will be very effective for the specific precision/V_{DD} these cells refer to, but it will simply be inappropriate for other precision/V_{DD} points. Moreover, since critical cells also depend on clock frequency, process, temperature, etc, the irregular solution will also not adapt to changes in the operating conditions.

Nevertheless, in the following, we also propose one possible irregular partitioning approach. Specifically, we obtain this partitioning using a regular grid as a starting point, and merging some of the BB domains according to their criticality in different precision configurations. The description of this *criticality-driven partitioning* is postponed to Section 3.3.7, as it requires some of the concepts introduced in Section 3.3.6.

3.3.5 Enabling Technologies to Support Multiple BB Domains and Associated Overheads

As anticipated in Section 3.3.2, the practical implementation of multiple BB domains requires two main enabling technologies, i.e. separation guardbands and back bias generators. Each of these elements introduces some overheads compared to a standard design, as discussed below.

Guardbands: Areas with independent BB voltage require a separation (guardband), in order to insert deep well trenches. In UTBB 28nm, the minimum BB domain guardband width is $2\mu m$, but in our work we chose a more conservative width for better quality of results. Specifically, we chose a width equal to three times the height of a standard cell row, i.e. $3.6\mu m$. More information on the technological details of our target process node can be found in literature [159, 160].

Back Bias Generators: Figure 3.9 shows a block diagram of the circuit used to assign the back biasing voltages to each BB domain at runtime, in the particular case of a dual assignment (NoBB or FBB). Each domain must receive two voltages, $V_{BB,N}$ for NMOS transistors and $V_{BB,P}$ for PMOS.

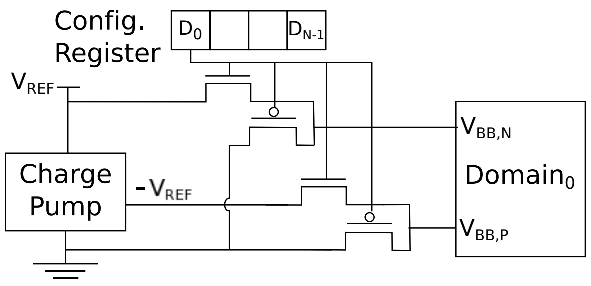


Fig. 3.9 Block diagram of a dual back bias generation circuit. FBB biasing voltages: $V_{BB,N} = V_{REF}$, $V_{BB,P} = -V_{REF}$.

Since $V_{BB,N}$ and $V_{BB,P}$ can only assume two values, back biasing voltages can be pre-generated, without the need of a controllable voltage generator, and can be shared by multiple operators. Charge pumps can be used to generate the negative voltage required for $V_{BB,P}$, as shown in the figure. Two power switches allow the connection of either NoBB or FBB voltages to the bias pins in each domain (shown only for one domain in the figure). Bias voltages are then routed to N-wells and P-wells of transistors through *well tap* cells placed at regular intervals on the die. Power switches are controlled by a configuration register, whose width is equal to the number of domains N. The register is memory-mapped, and is written when the bit-width of the operator must be modified. Thus, the area overheads for generating back bias voltages are limited to a few switches and flip-flops, and are negligible compared to those caused by the insertion of separation guardbands. Notice that we do not consider V_{DD} generation, as the latter is already present in most modern systems.

Runtime assignment of back bias voltages also introduces a *time* overhead, which makes the transition between two different assignments not instantaneous. This overhead is due to the loading of well capacitances by the circuit of Figure 3.9. As reported in Section 3.3.8, thanks to the small size of the BB domains considered in this work, the transition time is in the order of tens of nanoseconds. This overhead is acceptable, considering that the precision requirements are expected to vary for different applications (or portions thereof) and not for single instructions [9, 8].

3.3.6 Design Flow

In this section, we describe a fully automated EDA flow to implement precision-configurable operators based on our proposed fine-grain V_{th} tuning approach. A high-level block diagram of this flow is shown in Figure 3.10. The entire process internally leverages commercial EDA tools for P&R and timing analysis.

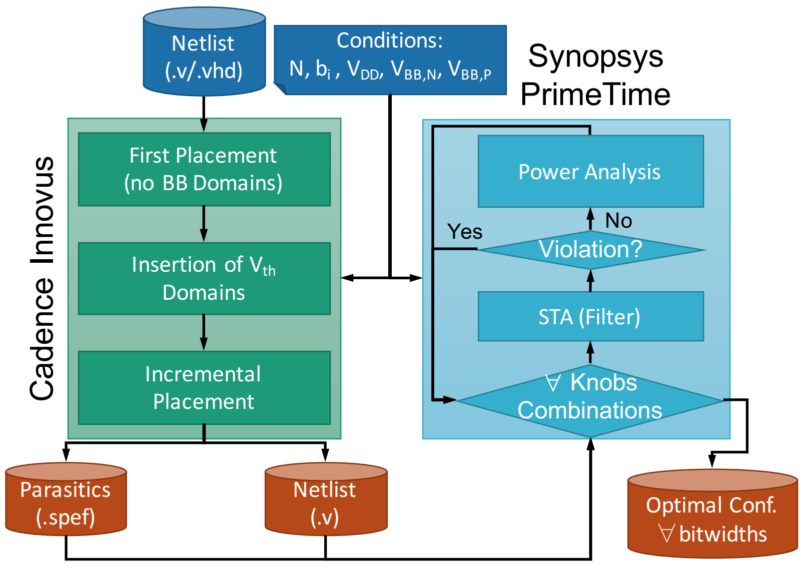


Fig. 3.10 Proposed EDA flow for the implementation of precision-configurable operators based on DVAS and fine-grain V_{th} tuning.

The main input to the flow is a gate-level netlist describing a standard implementation of the target hardware operator, whose precision is equal to the maximum desired precision of the output design. Moreover, the user needs to specify four additional parameters:

1. the V_{DD} values that can be applied to the entire operator (one or more)
2. the FBB voltages for N-well and P-well.
3. the relevant precision configurations to be considered throughout the flow (b_i).
4. the number of V_{th} domains N, which determines the acceptable area overhead.

The sequence of operations can be split in two main parts. The first *implementation phase* (green background, on the left in the figure), consists of the physical design of the operator, with the insertion of V_{th} domains. The result of this implementation then undergoes a second *analysis phase* (blue background, on the right), in which the best assignment of technological knobs for each bit-width is determined.

Implementation Phase

First placement: The implementation phase begins with a first placement of the operator at nominal voltage, without V_{th} domains. This operation is performed by the P&R tool using its standard algorithms. Hence, cells are placed according to the usual constraints of timing, area and power. The information gathered in this step is used during V_{th} domains creation to assign the cells to the closest domain, as shown in Figure 3.8.

Insertion of $\mathbf{V_{th}}$ domains: With the placement available, V_{th} domains can be created. The die area reserved to the operator is also enlarged to insert separation guardbands. The details of this phase depend on the selected circuit partitioning technique. When using regular tiling, domain creation is straight-forward and simply consists of the insertion of guardbands at equally spaced intervals. The details for an irregular partitioning are provided in Section 3.3.7.

Incremental placement and routing: Finally, an incremental placement is executed on the modified circuit. In this step, the tool takes into account the newly inserted V_{th} domains and their possible operating modes (NoBB or FBB) and consequently modifies gate sizing, position, etc., in order to meet all timing (setup and hold) and DRC constraints. The tool is also instructed to insert well taps for the connection of back bias voltage rails to the different domains. Finally, the implementation is completed with routing.

The outputs of the implementation phase are the placed and routed netlist of the operator, including V_{th} domains, and the relative parasitics file to be used in the following analysis.

Analysis Phase

We propose two algorithms to identify the combination of knobs (i.e. back bias voltage assigned to each domain and global V_{DD}) that minimizes power for each bit-width b_i. An optimal method based on exhaustive exploration, which has exponential complexity in the number of domains N, and a greedy heuristic with linear complexity. The two methods are detailed in the following. Regardless of the chosen search algorithm, the flow of the analysis phase is the same and is composed of two main steps.

Static Timing Analysis (STA) Filter: For a given combination of knobs, the operator is analyzed using static timing analysis, checking for setup and hold timing compliance. In case a violation is found, that combination is discarded. For example, a configuration in which NoBB (high V_{th}) is assigned to all domains and V_{DD} is set to a low value (e.g. $0.6V$) will probably incur in violations at maximum precision (e.g. $b_i = 32$ bit). STA is quite fast (in the order of 0.1s for the scale of our operators) and even faster for low precision netlists, thanks to the fact that many paths are disabled. On average, we have observed that about 75% of the configurations are filtered by STA, hence this step allows to greatly reduce the total analysis time.

Power Analysis: Configurations that pass the STA filter are then analyzed for total power. In this phase, switching activity data from simulations can be optionally loaded to increase the estimation accuracy. If the configuration consumes less power than the previous best for the same bit-width, it is saved in an internal database. When all configurations have been analyzed, the tool produces in output a list of configurations (one for each relevant precision), with the corresponding knobs settings.

Note that this process (specifically the STA filter) is dependent on the target clock frequency f_{clk}. If the operator is to be used at multiple frequencies, the analysis must be repeated.

Exhaustive Search

The easiest method to identify the optimal knobs configuration is to analyze all possible combinations of domains V_{th} and global V_{DD} exhaustively. This corresponds to evaluating all 2^N combinations of NoBB/FBB assignment to the N domains, for each V_{DD} and b_i. An example is shown in Figure 3.11, where red squares represent FBB domains.

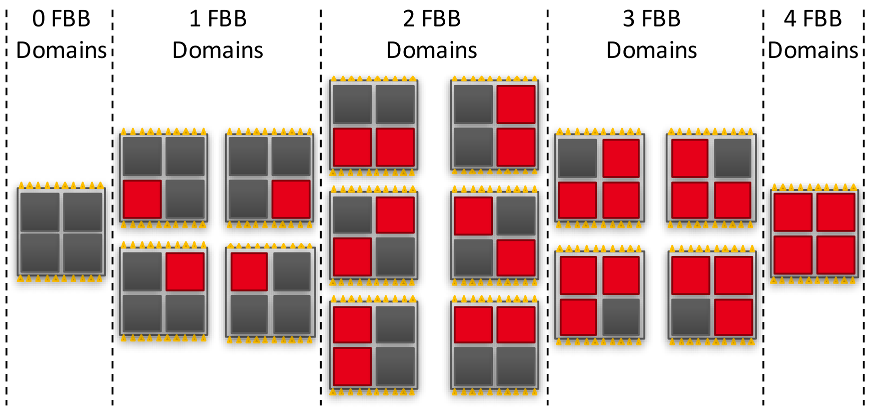


Fig. 3.11 Example of the back biasing configurations considered in exhaustive search, with $N = 4$, for a given (b_i, V_{DD}) point. Red squares correspond to boosted domains.

Therefore, the overall analysis complexity is $O(2^N \cdot B \cdot N_{V_{DD}})$, where B is the number of considered bit-widths, and $N_{V_{DD}}$ is the number of supply voltages to be explored.

Since these values are all relatively small, and since STA filters out most of the combinations, exhaustive exploration is feasible. Indeed, N is in the order of a few tens (more domains would generate unacceptable area overheads). B in the worst case coincides with all possible bit-widths (e.g. from 1 bit to 32 bit) but is typically smaller. $N_{V_{DD}}$ depends on the step and range of variation of V_{DD}: assuming a 100mV step and a range between 0.6V and 1.0V, $N_{V_{DD}} = 5$. With these values, the number of points to consider is in the order of thousands. Power estimation including importing of VCD traces takes about 1s on our target platform (Intel Xeon E5-2630@2.4GHz with 128GB DDR3), hence the entire analysis can be carried out in tens of minutes or few hours. This time is acceptable for a final stage of design, and it ensures that the *optimal* combination of knobs is detected. However, during exploratory phases (e.g. when deciding the best value of N for a given operator), a faster heuristic solution, described in the next section, is preferable.

Dual-V_{th} Guided Search

Dual-V_{th} assignment (multi-V_{th} in general) is a static design-time solution for leakage power optimization, applicable when the target standard-cell library includes devices designed to have different V_{th} values [170, 171]. The basic idea is that of mapping gates in critical timing paths to low-V_{th} devices for performance maximization, and the remaining gates to high-V_{th} devices for leakage recovery. Differently from dynamic V_{th} assignment (e.g. back biasing), static dual-V_{th} can be applied with a single cell granularity.

We exploit the algorithms normally used for static dual-V_{th} as a guidance for our analysis. To do so, we provide a commercial P&R tool with library characterizations in NoBB and FBB conditions, mapping them to high-V_{th} and low-V_{th} respectively. With these libraries available, we instruct the tool to implement static dual-V_{th} assignment, exactly as if the different V_{th} values were obtained technologically (rather than through biasing).

After running the algorithm, the circuit contains a mix of NoBB and FBB cells, with the latter mostly located in critical paths. We then use the *concentration of FBB cells* as an indication of the domains that are most "important" for a particular bit-width and V_{DD}; this allows the designer to drastically reduce the number of configurations to test. The overall algorithm can be summarized as follows. For a given b_i and V_{DD}:

1. We disable the appropriate timing paths for b_i, and have the P&R tool execute a dual-V_{th} assignment on the operator.
2. If dual-V_{th} did not insert any FBB cell, we just analyze the configuration in which all domains are in NoBB.
3. Otherwise, we start boosting the domain which contains the highest number of FBB cells.
4. We run STA and possibly power analysis on this configuration.

5. If timing compliance is reached, we stop. Otherwise, we apply FBB to the next domain with the highest number of FBB cells, and repeat from step 4, until one configuration is compliant, or all domains have been boosted.

An example of this procedure is shown in Figure 3.12, where blue squares represent individual FBB cells identified by dual-V_{th}. With this method, the number of analyses performed in the worst case, for a given b_i and V_{DD}, is linear in the number of domains. Thus, the overall complexity is $O(N \cdot B \cdot N_{V_{DD}})$, and with the same setup described for exhaustive search, the number of configurations to test reduces to few hundreds. The time to execute single-cell dual-V_{th} with our P&R tool is less than 10s, and this step is only executed once for every b_i and V_{DD}.

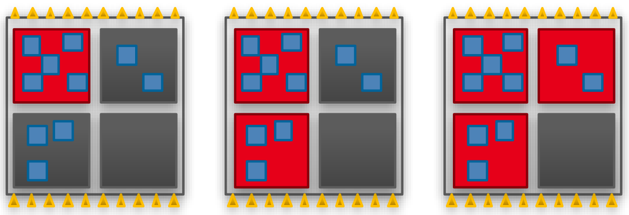


Fig. 3.12 Example of the back biasing configurations considered in the proposed dual-V_{th} guided search, for a given (b_i, V_{DD}) point. Red squares correspond to boosted domains, while blue squares correspond to individual boosted cells according to a dual-V_{th} assignment algorithm.

It is worth emphasizing that we do not boost all domains that contain at least one FBB cell in a single step. This is because when we apply FBB to an entire domain, more cells than those strictly needed (blue squares) will be sped-up. In practice, we observed that these additional cells often "compensate" for the fact that some of those identified by dual-V_{th} in other domains are not boosted. Thus, timing compliance could be reached with a smaller number of domains.

This procedure does not necessarily identify the optimal configuration of knobs. Firstly, because the single-cell dual-V_{th} algorithms embedded in EDA tools are heuristic [170]. Secondly, because the concentration of FBB cells might not be sufficient to identify the optimal domain to boost. For example, if the concentration of FBB cells in two domains is similar and small, boosting either one might be sufficient for timing compliance, due to the "compensation" effect described above. However, the power increase caused by forward back biasing will generally be different among domains, even in the case of regular tiling, due to different leakage consumption of each standard cell. Therefore, the least leaky domain should be selected in this case, rather than the one containing more FBB cells. Nonetheless, we show in Section 3.3.8 that this heuristic identifies the optimal design point in most of the cases and is therefore interesting for early design stages, or for cases in which N is too large for exhaustive exploration.

3.3.7 Irregular Partitioning

Besides being used to guide the selection of back biasing configurations as described above, dual-V_{th} assignment can also be used during implementation, as a way to construct irregular V_{th} domains.

Specifically, we propose to run dual-V_{th} *before* the insertion of back bias domains (after the first placement of the operator), in all precision and V_{DD} conditions. This allows to identify the *minimum set of cells* that should ideally be boosted to reach timing compliance in that condition (i.e. the critical cells described in Section 3.3.3). This information is then used as basis to determine domain boundaries.

As explained in Section 3.3.4, the key issue is merging the results of different dual-V_{th} runs into a single physical partitioning. As a first heuristic simplification, when multiple V_{DD} values are considered, we select, for each precision b_i, the supply voltage that produces the minimum power consumption according to the dual-V_{th} output. In this way, we are left with B different sets of FBB cells to be considered for the construction of domains.

Classic clustering approaches (k-means, dbscan, etc.) cannot be applied straightforwardly to this multi-objective problem. We propose instead a simple aggregative algorithm to generate the domains, named *criticality-driven* partitioning hereinafter. The approach is depicted in Figure 3.13 for a simplified example in which $B = 2$. We first subdivide the placement of the operator in a virtual grid of identical regions, each of which represents a *potential* V_{th} domain. As for regular tiling, the maximum number of regions is determined by the acceptable area overhead. Then, we apply the analysis described in the previous section on the virtual grid, i.e. we identify the regions that should be boosted for each considered bit-width, using the dual-V_{th} output as guidance. A possible result is shown in the two leftmost drawings of Figure 3.13. Finally, we generate V_{th} domains merging regions that, for all bit-widths, always receive the same BB voltage (rightmost drawing in the figure). The merging

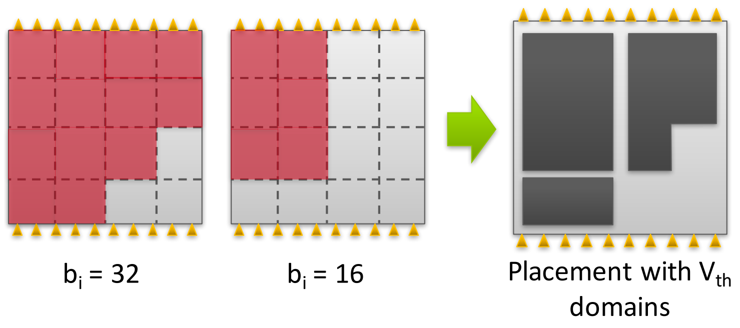


Fig. 3.13 Example of the proposed criticality-driven partitioning to build irregular V_{th} domains.

phase only deals with N configurations in total, therefore its complexity is negligible compared to the previous guided search.

The advantage of this solution with respect to regular tiling is a possible reduction of the number of domains, for similar results in terms of power consumption. This translates into smaller area overheads and simpler control. However, irregular partitioning also has a disadvantage in terms of flexibility, as anticipated in Section 3.3.4. First, merging domains as shown in Figure 3.13 is only possible when few bit-width configurations are considered; with too many precision modes, no group of domains *always* receives the same BB voltage, and merging is impossible. Second, all the previous analyses depend on the target operating conditions, and in particular on the clock frequency f_{clk}. Hence, the partitioning determined with the aggregative method will be optimal only for a single clock frequency; modifying f_{clk} would change the way in which virtual regions should be merged into physical domains. Regular tiling, in contrast, uses the minimum acceptable size for each domain, and leaves more freedom to reconfigure the V_{th} assignment depending on operating conditions at runtime.

3.3.8 Experimental Results

Setup

We have tested our method on a set of representative operators, implemented on a 28nm flip-well UTBB FDSOI standard-cell technology library from ST Microelectronics. We have started from designs downloaded from OpenCores [155], specified in behavioral VHDL or Verilog. We have performed synthesis using Synopsys Design Compiler L-2016.06 and P&R using Cadence Innovus 16.1. Post-implementation netlists and parasitics have been loaded in Synopsys PrimeTime L-2016.06 for timing and power analyses.

Throughout our analyses, we have considered a total of five V_{DD} conditions, from $1.0V$ to $0.6V$, in steps of $0.1V$. As FBB voltages, we have used $V_{BB,N} = 1.1V$ and $V_{BB,P} = -1.1V$ for NMOS and PMOS devices respectively. Unless specified differently, analyses have been performed in a slow-slow process corner, at 125C. During the first P&R of the operators (without BB domains), we have considered as nominal operating condition $V_{DD} = 1V$, with FBB applied to *all* cells of the circuit.

We have considered the following four representative operators:

- A 16x16-bit multiplier, with radix-4 Booth encoder and Wallace tree [32].
- A 64-bit Kogge-Stone adder [32].
- A 16-bit, 30-tap FIR filter in direct form [156].
- A 16-bit butterfly unit (the main datapath component of a FFT accelerator) [156].

The adder and the multiplier have been considered because they are the basic building blocks of most datapath circuits. The remaining two designs have been chosen as representative of datapath accelerators for signal processing applications, a typical error-tolerant domain [9]. The latter two benchmarks prove that our method yields good results also when applied directly on bigger and more complex circuits.

DAS and DVAS [34] have already been demonstrated superior to traditional architectural solutions for precision-configurable hardware operators design ([76, 75], etc.). Therefore, in this work we have considered these two techniques as the main references for comparison.

Comparison with standard DAS

Our method can provide power benefits also when using only dynamic V_{th} tuning, independently from the availability of a global supply voltage scaling knob. To show these benefits, we have implemented a precision-configurable version of the Booth multiplier composed of 4 BB domains, using a 2x2 regular tiling grid.

Figure 3.14 shows the power versus bit-width curves obtained by this multiplier at a fixed $V_{DD} = 0.90V$. The graph reports the Pareto frontier, i.e. the *minimum* total power for each bit-width, obtained testing all possible back biasing assignments to the 4 domains, with the exhaustive analysis method described in Section 3.3.6. Power consumption includes both leakage and dynamic components. The clock frequency is set to the maximum value that does not incur timing violations at maximum precision, with FBB applied to all 4 domains (i.e. $f_{clk} = 1.25GHz$). Only configurations that *fully* meet timing constraints, taking into account disabled paths due to reduced dynamic, are reported in the figure. We neglect bit-widths smaller than 4-bit as they are of very limited interest, even in error tolerant applications.

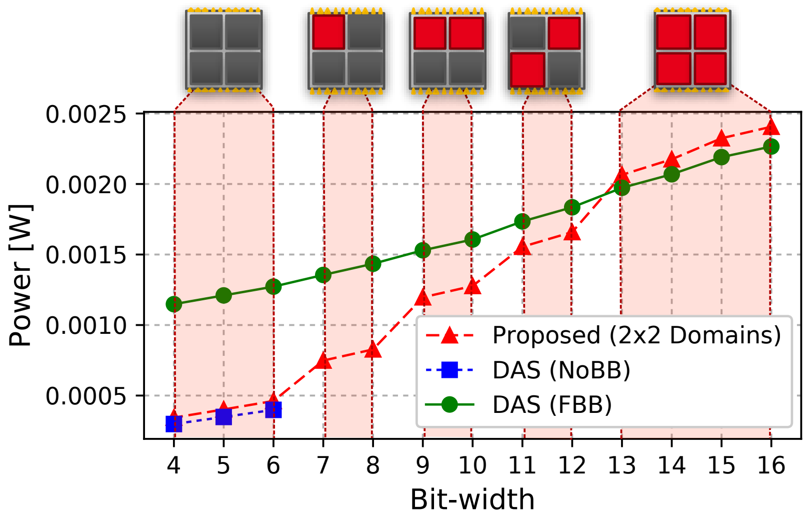


Fig. 3.14 Power versus bit-width curves for a Booth multiplier enhanced with fine-grain V_{th} tuning, at fixed $V_{DD} = 0.90V$.

For comparison, Figure 3.14 also reports the results of simple DAS applied to the original multiplier, without BB domains. Two set of DAS results are shown: one when no back bias is used, the other when FBB is applied “monolithically” to the entire operator. In both cases, the clock frequency is set to the same value used for our method.

At the considered f_{clk}, the original multiplier with NoBB cannot operate at precisions higher than 6-bit, without incurring timing violations. In contrast, the multiplier version modified by our flow with the insertion of BB domains can reach maximum precision (16-bit), thanks to fine-grain FBB. At the same time, our fine-grain solution is (on average) significantly more efficient than using FBB on the entire design. For example, at 7 and 8-bit precision, the power savings with respect to monolithic FBB are 44.8% and 42.3% respectively.

These benefits are paid with a small power overhead with respect to the operator without domains at very low and very high precision (6.2% worst case). This is mostly due to the cells resizing performed in the incremental placement phase, to account for the insertion of separation guardbands. Secondarily, the additional area also causes an increase in the length of metal routes, and consequently in the dynamic power consumption. The area overhead is approximately 14% compared to the operator without domains.

On the top of Figure 3.14, we also report graphically the back biasing configurations used for each bit-width, where red squares represent domains that receive FBB. Notice that, as expected, a smaller bit-width requires a smaller or equal number of boosted domains, and that all domains are boosted in at least one precision configuration.

Comparison and combination with DVAS

In its most effective embodiment, fine-grain V_{th} is combined with voltage scaling applied to the entire operator (i.e. DVAS). To show this scenario, we have implemented both our method and standard DVAS on all four benchmark circuits. We have used a regular grid of BB domains, with the configurations shown in Table 3.2 (column *Domains*). The table also reports the corresponding area overheads (column *Area Ovr.*), and the clock frequencies used for analyses (column f_{clk}).

Figures 3.15–3.18 show the results of this experiment on a bit-width versus power consumption plane. The curves are generated as for the previous DAS experiment, and report the *minimum power* for each bit-width, considering all possible combinations of V_{DD} and (for our method) BB assignment. For a fair comparison, for each benchmark we report both the results of DVAS with NoBB and DVAS with FBB (applied monolithically to the entire design).

The shapes of the curves are similar to the case of simple DAS. Through fine-grain power and speed tuning, our method is significantly more efficient than DVAS with FBB at the same frequency, whereas DVAS with NoBB can seldom reach

Circuit	**Domains**	**f_{clk} [GHz]**	**Area Ovr. [%]**	**Max. Power Saving [%] (Bits)**	**Max. Power Ovr. [%] (Bits)**
Mult.	2x2	1.25	14.5	32.67 (10)	0.26 (9)
Adder	2x2	2.00	11.3	33.78 (12)	0.46 (64)
Butterfly	3x3	1.00	17.0	26.45 (5)	17.15 (4)
FIR	3x3	0.75	15.1	43.06 (16)	0 (-)

Table 3.2 Implementation details and results summary of the comparison between the proposed fine-grain V_{th} tuning method and standard DVAS.

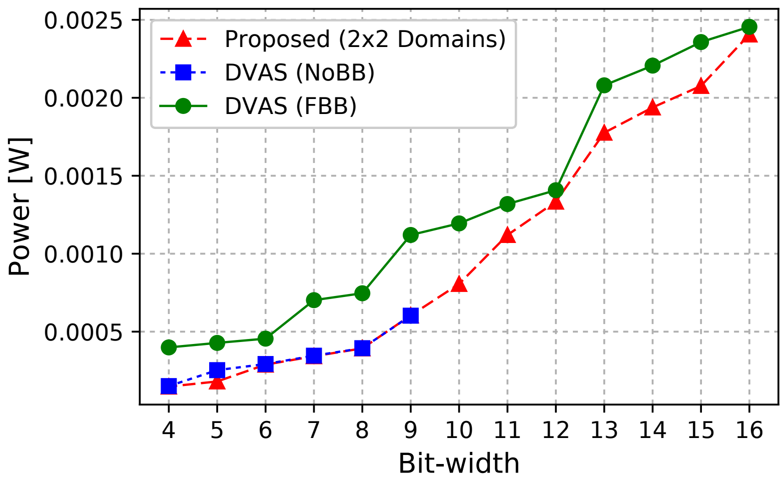


Fig. 3.15 Comparison among standard DVAS and our proposed design method based on fine-grain V_{th} tuning for a Booth multiplier.

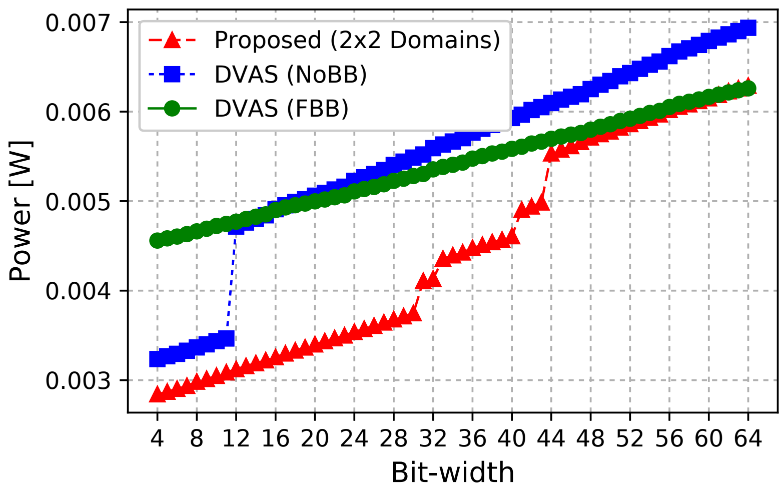


Fig. 3.16 Comparison among standard DVAS and our proposed design method based on fine-grain V_{th} tuning for a Kogge-Stone adder.

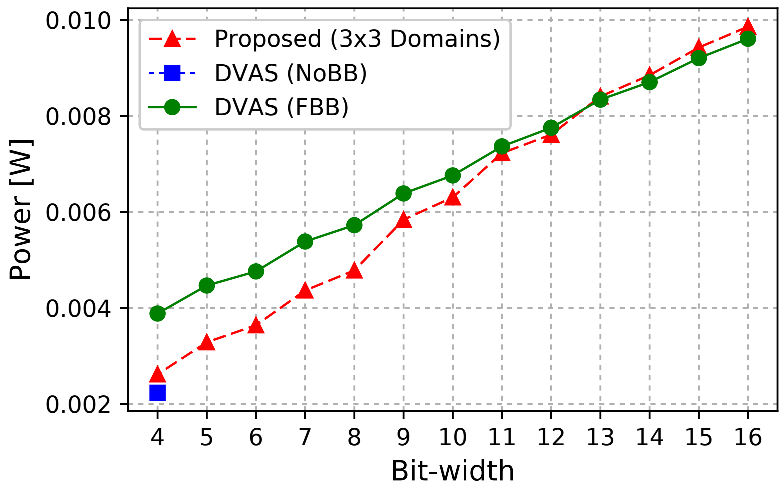


Fig. 3.17 Comparison among standard DVAS and our proposed design method based on fine-grain V_{th} tuning for a FFT butterfly unit.

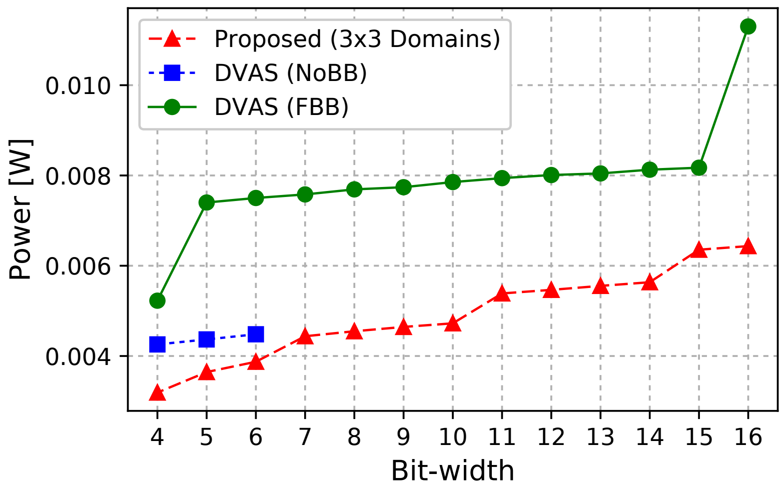


Fig. 3.18 Comparison among standard DVAS and our proposed design method based on fine-grain V_{th} tuning for a FIR filter.

maximum precision without violations. Even when the latter is timing compliant (e.g. in the case of the adder) our method can provide better power efficiency, by working at a lower V_{DD} and *only* speeding up (with FBB) the critical parts of the operator. This comes at the cost of a small leakage overhead, thanks to the fine granularity of BB domains.

Notice that, for adder, multiplier and FIR, the effects of the wall-of-slack are particularly evident looking at the step-wise shape of the Pareto frontier generated by standard DVAS. Each step corresponds to a change of V_{DD}, and occurs because scaling the supply voltage drastically reduces the maximum bit-width that can be used without violations. Therefore, the supply voltage must be kept constant for large ranges of bit-widths (obtaining only a linear power decrease due to the reduction in circuit activity). In contrast, our method can leverage fine-grain back biasing to obtain numerous "intermediate steps", and therefore produces a more graceful power versus precision dependency. In the case of the butterfly unit, the more linear curves of DVAS show that this unit is less affected by the wall-of-slack (probably due to a very relaxed implementation frequency) and consequently the improvement thanks to back biasing is less marked.

The maximum power savings and overheads obtained by our method with respect to DVAS are reported in Table 3.2 (columns *Max. Power Saving* and *Max. Power Ovr.*), together with the bit-widths at which these savings and overheads are obtained. For all designs, the maximum saving is higher than 25%, and reaches 43% for the

FIR filter. When combined with DVAS, the power overheads of our method are generally very small (less than 1%), except for the FFT butterfly. This exception is motivated by the fact that the number of domains used in the FFT experiments is large compared to the size of the circuit. Thus, the compensation of area overheads during incremental placement is more marked. Butterfly implementations with less BB domains were not chosen as they poorly isolated the critical paths of the circuit, and provided very limited power savings compared to DVAS, as shown in Table 3.3.

Impact of the number of V_{th} domains

The number of V_{th} domains has a strong impact on all figures of merit in our method. This is shown in Figures 3.19 and 3.20 using the Booth multiplier as an example. The first figure shows the power consumption obtained at different bit-widths for different configurations of BB domains. For ease of visualization, the graph only reports precisions between 8 and 16 bits. These results still refer to the regular tiling solution, and the only difference among configurations is the number and position of the tiles.

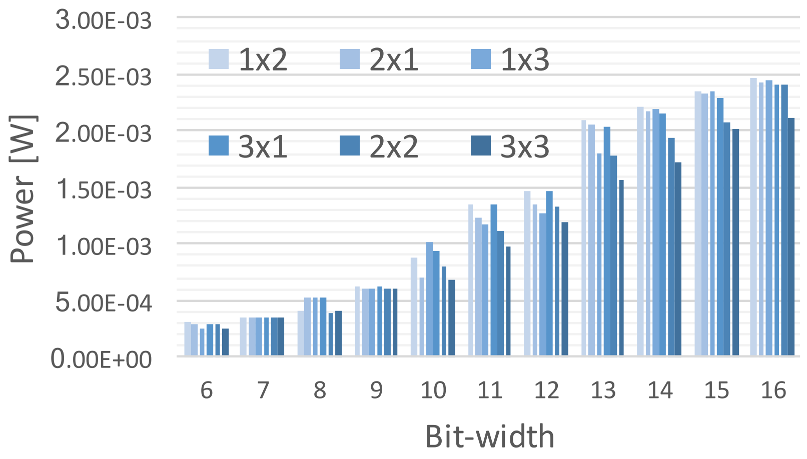


Fig. 3.19 Minimum power versus bit-width for a Booth multiplier implemented with different configurations of V_{th} domains, using a regular tiling approach.

As expected, increasing the number of domains produces a general reduction in power consumption, especially at high bit-widths. Indeed, the presence of more domains allows a finer-grain control of the parts of the operator to boost. In few particular cases, this trend is not respected, and increasing the number of domains has a negative effect of power (e.g. 2x1 versus 3x1 domains at 10-bit precision). This

Circuit	**1x2**	**1x3**	**2x2**	**3x3**
Multiplier	25.90(3.61)	26.82(3.87)	32.67(9.75)	42.56(14.10)
Adder	24.20(14.53)	22.61(15.98)	33.78(19.56)	33.90(22.11)
Butterfly	16.33(2.86)	19.70(2.76)	19.39(3.74)	26.45(9.14)
FIR	23.65(16.89)	34.60(15.31)	41.59(22.74)	43.06(31.40)

Table 3.3 Maximum (mean) power saving [%] of our proposed design method based on fine-grain V_{th} tuning with respect to standard DVAS, for different configurations of regular V_{th} tiles.

is due to the addition of guardbands between groups, which may cause displacements of critical cells that were previously placed close to each other by the tool. As a consequence, the incremental placement must perform substantial modifications to restore timing compliance, causing the trend inversion. Table 3.3 shows the maximum and mean power savings with respect to DVAS obtained by the four benchmark circuits for a subset of groups configurations, in the same conditions of the previous experiment. As for the multiplier, finer granularity generally allows larger power reductions.

Figure 3.20 reports the area and settling time overheads for the groups configurations tested on the Booth multiplier. The settling times refer to a SPICE simulation performed using an implementation of the circuit in Figure 3.9 on the target technology, and the values reported are worst-case, i.e. they refer to the domain with the largest well capacitance. Expectedly, the area overhead increases with the number of domains. On the contrary, the settling time is shorter for a larger number of domains, due to the smaller well area of each region.

In general, the selection of the number of BB domains strongly depends on system-specific constraints; the power reduction in precision modes of interest must be balanced with the area budget. However, as explained in the next section, the design space of possible implementations can be explored exhaustively, at least for a small number of groups (≤ 10).

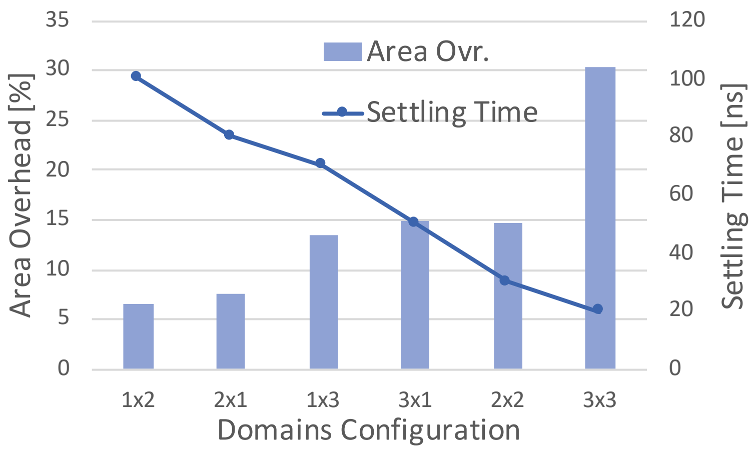


Fig. 3.20 Area and settling time overheads versus bit-width for a Booth multiplier implemented with different configurations of V_{th} domains, using a regular tiling approach.

Exhaustive and heuristic search algorithms

Figures 3.21 and 3.22 show the comparison between exhaustive search and the dual-V_{th} guided heuristic for the identification of the optimal configuration of V_{DD} and BB. The curves referring to exhaustive search are the same as in Figures 3.15 and 3.18, whereas the ones referring to guided search are obtained with the heuristic method described in Section 3.3.6.

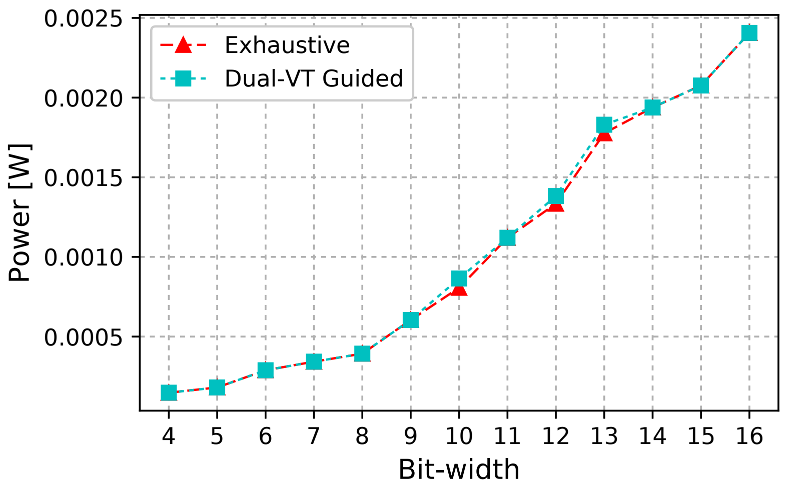


Fig. 3.21 Comparison between exhaustive and dual-V_{th} guided search algorithms for identifying the optimal configuration of fine-grain V_{th} and global V_{DD} in a precision-configurable operator. Example on the Booth multiplier.

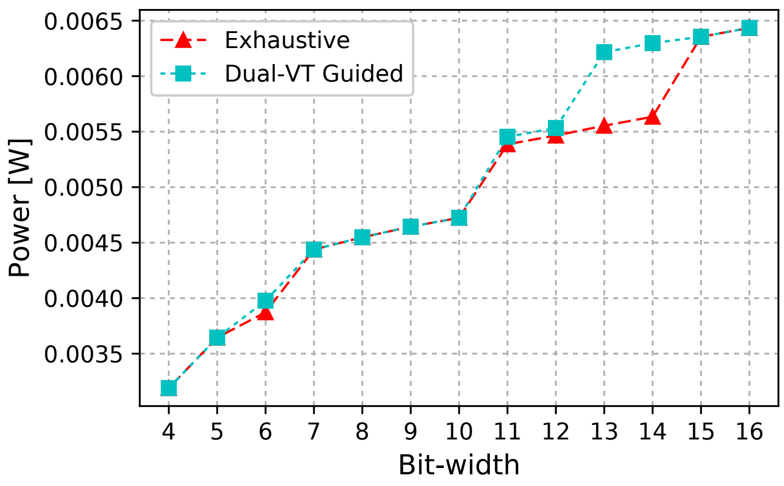


Fig. 3.22 Comparison between exhaustive and dual-V_{th} guided search algorithms for identifying the optimal configuration of fine-grain V_{th} and global V_{DD} in a precision-configurable operator. Example on the FIR filter.

Multiplier	**Adder**	**Butterfly**	**FIR**
7.2(1.0)	9.3(1.4)	6.6(1.5)	11.9(2.2)

Table 3.4 Maximum (mean) difference in power consumption [%] between the exhaustive and dual-V_{th} guided search algorithms for the identification of the optimal configuration of fine-grain V_{th} and global V_{DD} in a precision-configurable operator.

For both circuits, the proposed heuristic is able to identify the optimal configuration in the majority of the bit-widths conditions. Even when it fails to identify the correct combination of knobs, the algorithm still outputs a configuration that is close to the optimal. The maximum and average difference in power consumption between the solution identified by the exhaustive algorithm and the one found by the heuristic is reported for all four benchmarks in Table 3.4.

While these errors might be unacceptable for a final design, the small average deviations prove that the heuristic method can be used to quickly explore the design space (e.g. to identify the optimal number and configuration of BB tiles, or to compare regular with irregular partitioning), providing results that closely resemble the optimal ones, despite the complexity reduction (from exponential to linear). On our testing platform, whose technical specifications have been introduced in Section 3.3.6, the exhaustive search applied to the multiplier requires about 1 hour and 20 minutes, whereas the guided search only takes about 12 minutes.

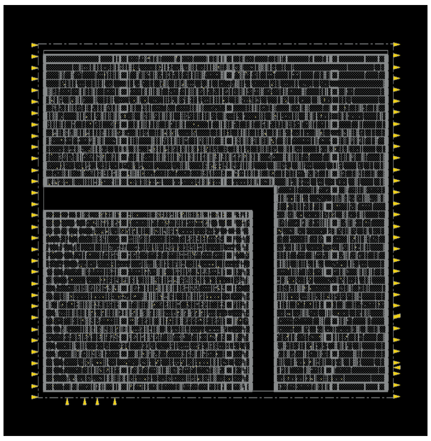


Fig. 3.23 Placement snapshot of a precision-configurable Booth multiplier implemented using the proposed fine-grain V_{th} tuning method, with irregular BB domains.

Irregular Partitioning

In this experiment, we compare regular and irregular partitioning approaches on one example design, namely the Booth multiplier. Specifically, we compare the regular tiling approach with a 2x2 grid with an irregular implementation built using the approach described in Section 3.3.7. To generate the irregular domains, we have started from a virtual 3x3 grid, and we have assumed that the target application only requires three precision modes: 4, 8 and 16 bit. The clock frequency has been set to the same value used for the regular design (see Table 3.2). After the merging phase, the resulting circuit only contains 2 domains; a placement snapshot obtained from the tool is shown in Figure 3.23.

Figure 3.24 shows the power consumption of the two versions of the circuit in the three considered bit-width conditions. The irregular version is slightly more efficient than the regular one due to the smaller number of domains, which eases the placement process. Most importantly, reducing the number of domains simplifies the control logic of the modified circuit (e.g. the size of the configuration register in Figure 3.9) and reduces separation guardbands overheads, while maintaining comparable power. However, these advantages must be balanced with the reduction in flexibility discussed in Section 3.3.7. In fact, the irregular version of the operator is tailored for three precision modes only, and for a *single* clock frequency.

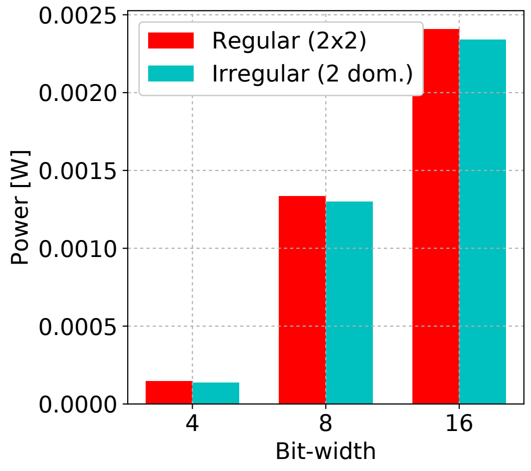


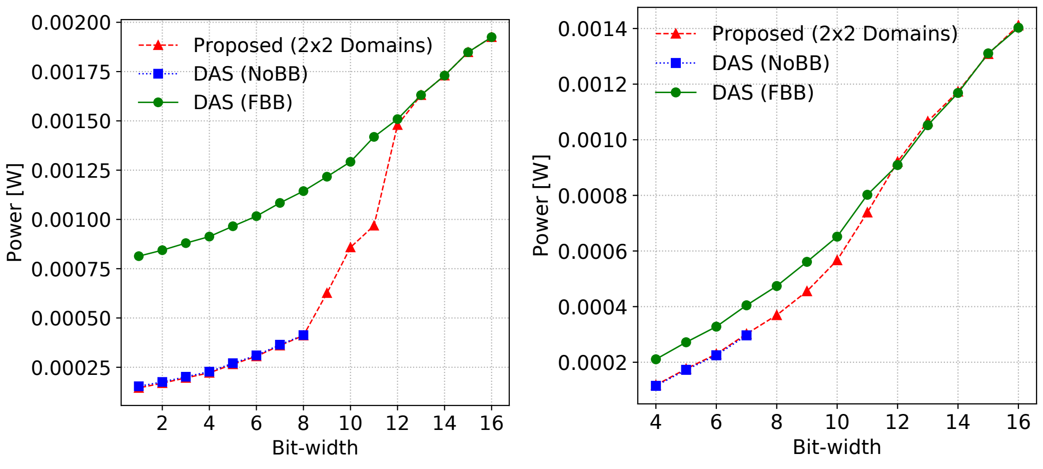
Fig. 3.24 Power versus bit-width for two precision-configurable implementations of the same Booth multiplier, obtained with our proposed fine-grain V_{th} tuning method, using regular and irregular partitioning techniques respectively.

Effects of operating conditions

As a final experiment, we assess how the results of our methodology depend on operating conditions. Intuitively, the larger the share of leakage in the total power, the higher will be the benefits of our approach. In other words, larger savings are expected at high temperature, low voltage, and low frequency [159], i.e. the corners that cause circuits to be most leaky. As a matter of fact, when leakage is negligible, applying FBB to the *entire* circuit has almost no impact on total power, and the difference between DVAS with FBB applied to the entire circuit and our solution becomes less evident. Conversely, the fine-grain approach is more effective when the leakage increase due to FBB has a strong impact on power, since only critical portions of the circuit are boosted.

Two examples are shown in Figure 3.25. Both plots refer to the same circuit as Figure 3.14. In Figure 3.25a, the clock frequency has been reduced from 1.25 to 1 GHz, leaving the temperature at the default 125C. Conversely, Figure 3.25b has been obtained leaving f_{clk} at 1.25 GHz, and reducing T from 125C to 25C. As expected, the savings of our method compared to global back biasing increase at lower frequency (the maximum saving at 1 GHz is 48.5% at 9 bit precision). Conversely, they reduce at low temperatures, although the maximum power saving at 25C is still 22.1% (at 8 bit), a value that can justify the area overhead of our method.

Moreover, notice that 25C is quite a low junction temperature, so this plot refers to a pessimistic condition [2].



(a) $f_{clk} = 1GHz$ (b) $T = 25C$

Fig. 3.25 Effects of operating conditions on the power versus bit-width Pareto curve of a Booth multiplier implemented with our proposed fine-grain V_{th} tuning technique.

In general, low power IoT devices normally operate in conditions were leakage is comparable to dynamic power (e.g. ultra-low voltage and low frequency) [5], which are exactly the operating points where our method performs best.

3.4 Application-Driven EDA Flow for Precision-Configurable Operators

The technique proposed in the previous section of this manuscript has been proven effective in improving the results of DAS and DVAS when they are combined with a standard EDA flow. However, that solution depends on the presence of an effective knob for tuning the speed and power consumption of a hardware operator at fine-grain. In FDSOI, this is achieved thanks to the strong impact of adaptive back biasing on V_{th}. Not all technologies have similarly powerful knobs available; for instance, in bulk CMOS, adaptive body biasing has a much more limited effect.

[2]Intermediate temperatures between 25C and 125C have not been tested due to lack of library characterizations.

In this section, we describe an alternative EDA flow for designing precision-configurable operators that does not rely on the existence of a fine-grain power/delay tuning knob. As such, this second solution can be applied to a wider set of technology targets. The two techniques have in common the way in which multiple quality configurations are obtained, that is analogous to the one of DAS and its variant, i.e. precision modulation by means of LSB gating.

The motivation for this work is the same presented in Section 3.3.1, i.e. the limited compatibility of DVAS with standard EDA approaches, due to the wall-of-slack phenomenon. However, rather than reducing this effect by means of dynamic path reshaping, in this work we propose to *statically* avoid the wall-of-slack by modifying the synthesis process. Therefore, the method presented in this section is more similar to standard DVAS than the previous one. While the previous solution significantly revised the original principles of [34], in this case those same principles are maintained, and our additional contribution focuses on how to actually obtain the theoretical benefits of DVAS within a realistic industrial design flow.

To this end, we describe the first published algorithm for the synthesis of precision-configurable circuits based on DVAS, that can be fully integrated with standard EDA tools. Our algorithm identifies the optimal supply voltage for each target bit-width, and constrains the synthesis process accordingly, preventing the formation of the wall-of-slack. Throughout this process, information on the target application, obtained through off-line characterizations, can be leveraged to drive the optimization. Thus, our method is particularly suited for application-specific hardware accelerators. To show the effectiveness of our solution, we apply it to a Multiply-And-Accumulate (MAC) operator, which is the key data-path component in accelerators for artificial neural networks [48]. On this benchmark, we obtain a total energy saving $> 25\%$ compared to a standard implementation of DVAS.

3.4.1 Multi-scenario optimization in DVAS-based circuits

All state-of-the-art commercial synthesis and P&R tools support some form of *multi-scenario* optimization. In this type of optimization, the tool is instructed to simultaneously consider multiple *corners* (i.e. process, supply voltage and temperature or PVT points) while synthesizing a circuit. Each corner is associated with a corresponding *operating mode*, that is a set of constraints to be used during the optimization of that corner.

In a DVAS-based circuit, multi-scenario optimization can be leveraged to ensure that the circuit does not have timing violations at the desired V_{DD}, for each considered bit-width mode. To do so, a separate scenario for each mode must be created, setting the desired V_{DD} as part of the corresponding PVT corner.

Then, the *case analysis* features of the tool must be used within each operating mode, to mimic the effect of zeroing LSBs. In particular, constant logic-0s must be imposed on the inputs that will be disabled in a given mode, so that the tool can propagate these constraints to the circuit internal signals. Paths that have zeros as inputs are marked as *false*, and the tool ignores them during the optimization of the corresponding scenario. More precisely, false paths are ignored when checking timing compliance and when evaluating the power consumption of a given reduced-precision scenario. However, thanks to the fact that multi-scenario optimization considers all precision modes concurrently, no logic is *eliminated* from the circuit. In fact, the same paths that are false in a certain reduced-precision condition are not false in the maximum precision mode.

Providing the tool with one scenario for each precision mode will force it to concurrently ensure timing compliance in all V_{DD} and bit-width combinations. Thus, the final design will be able to operate at reduced V_{DD} when low bit-widths are used, differently from what happens normally in standard DVAS, thanks to a more graceful distribution of timing paths. In turn, this will yield larger power savings at reduced precision.

The drawback of this solution is an increase in the area occupation and power consumption at maximum precision. In fact, to allow lower V_{DD} operation, gates belonging to true paths (i.e. not disabled) in low bit-width modes will be upsized and/or mapped to lower threshold voltage devices compared to a standard implementation of the same circuit. Fortunately, in many applications, maximum precision is only seldom required [172, 173, 48], hence this solution has the potential to be advantageous in terms of *total energy*.

3.4.2 Proposed Greedy V_{DD} Selection and Synthesis

The multi-scenario methodology described in Section 3.4.1 can be implemented assuming that the best V_{DD} value for each bit-width mode is known. In practice, finding this value is not trivial. In this work, we propose to find an appropriate V_{DD} for each bit-width using the greedy incremental synthesis algorithm shown in Figure 3.26.

```
1: procedure GREEDY INCREMENTAL SYNTHESIS
2:     $V_{start}$ = current supply voltage, initialized to nominal $V_{DD}$
3:     $s_0$ = nominal scenario [$V_{start}$, $b_{max}$]
4:     S = list of scenarios, initialized to $s_0$
5:     C = nominal circuit netlist, synthesized in scenario $s_0$
6:     for all reduced bit-widths b (in decreasing order) do
7:         $s_{new}$ = [$V_{start}$, b] (same $V_{DD}$ as previous scenario)
8:         $P_{new}$ = weighted power of C in scenarios S + $s_{new}$
9:         $V_{new}$ = $V_{start}$
10:        do
11:            $s_{ref}$ = $s_{new}$, $P_{ref}$ = $P_{new}$, $V_{ref}$ = $V_{new}$
12:            $V_{new}$ = decrease $V_{new}$
13:            $s_{new}$ = [$V_{new}$, b]
14:            $C_{new}$ = incr. synthesis of C in scenarios S + $s_{new}$
15:            Ts = worst slack of $C_{new}$ in scenarios S + $s_{new}$
16:            $P_{new}$ = weighted power of $C_{new}$ in scenarios S + $s_{new}$
17:        while $V_{new} > V_{min}$ and $Ts \geq 0$ and $P_{new} < P_{ref}$
18:        S = S + $s_{ref}$
19:        $V_{start}$ = $V_{ref}$, C = $C_{new}$
20:    end for
21: end procedure
```

Fig. 3.26 Proposed greedy V_{DD} selection and incremental synthesis algorithm for precision-configurable operators based on DVAS.

This procedure takes as input a gate-level netlist of the target circuit, synthesized in nominal conditions, i.e. maximum V_{DD} (V_{start}) and bit-width (b_{max}). Synthesis constraints (clock frequency, boundary conditions, etc.) are also received as inputs, as well as the set of reduced bit-width modes to consider during the process, and the available V_{DD}s. In the pseudo-code, the $+$ symbol used with scenarios (lines 8, 14, 15, 16) corresponds to the list concatenation operation.

The core of the algorithm spans the available set of supply voltages, starting from the nominal value and progressively decreasing it. While doing so, it adds one bit-width mode at a time to the considered set, in decreasing order of precision.

For each new bit-width, the first step (line 8) is an assessment of the power consumption of the circuit when the new mode uses the *same* supply voltage as the previous one (i.e. the only difference between the modes is in the number of zeroed LSBs). This power estimation, as well as the one in line 16, takes into account *all bit-width modes* in the considered set, as detailed in Section 3.4.3.

Then, V_{DD} is decreased (line 12), and an *incremental* synthesis (line 14) is performed on the circuit, so that the tool can try to enforce timing compliance at a lower V_{DD} for the considered bit-width. During the re-synthesis phase, the tool also takes into account the previous (larger) bit-width modes, using the multi-scenario functionality described in Section 3.4.1.

In some cases (i.e. when V_{DD} is too low), the tool can fail to resolve timing violations. Therefore, the resulting netlist undergoes a Static Timing Analysis (STA) in line 15, where the worst setup and hold slacks in *all* bit-width modes are evaluated. Then, the power of the modified netlist is evaluated in line 16.

The progressive scaling of V_{DD} is continued until it results in a *reduction of total power*, and the tool is able to avoid timing violations. The lowest V_{DD} that satisfies these two conditions is selected as final supply voltage for the considered bit-width, and the corresponding scenario is saved (line 18). Then, a new precision mode is added, and the procedure is repeated starting from the current V_{DD}, i.e. $\mathrm{V}_{\mathrm{start}}$.

The proposed algorithm requires to re-synthesize the operator several times. Thus, we perform this optimization at gate-level, rather than after P&R. Although the latter solution would yield more accurate results, it would also require a longer execution time. Moreover, as long as power estimations after synthesis and P&R are correlated (although exact values might differ), it makes sense to *select* the appropriate V_{DD}s post-synthesis, and then *enforce* them during P&R.

In particular, the V_{DD} selected at gate-level for each precision mode is then provided to the P&R tool as part of the corresponding scenario. The tool then executes a single multi-scenario P&R, considering all modes in its optimizations.

3.4.3 Application-driven Power Weighting

A key feature of the algorithm in Figure 3.26 is the way in which the total power consumption of the operator is assessed. Specifically, as reported in the figure (lines 8 and 16), we consider a *weighted* contribution from each scenario, that is:

$$P_{tot} = \sum_i w_i P_i \tag{3.11}$$

where P_i is the power in each bit-width mode. P_i depends on the V_{DD} applied to the circuit in that mode, and is affected by the re-synthesis phase, because of the possible gates resizing/remapping performed by the tool when further scenarios are added.

Using $w_i = 1$ for all i, would give the same importance to the power consumption in all modes. However, our main concern is energy efficiency, which also depends on the usage frequency of each mode. For example, if the power consumption of the circuit at maximum precision is high, but that mode is very rarely used in the target application, the impact on total energy will be minimal.

Therefore, we propose to select weights w_i in a way that lets our algorithm minimize a "proxy" of the total energy consumption, leveraging information about the target application. In practice, this can be done assigning to w_i a value proportional to the *probability* that a given operation requires mode i, which can be obtain through an offline characterization of the application.

Notice that the algorithm of Figure 3.26 is greedy, in that it stops decreasing V_{DD} at the first minimum of (3.11), although this might not be the global best. For example, if the nominal V_{DD} and bit-width for a circuit are 0.9V and 16-bit respectively, using 0.8V for the 8-bit mode might cause a slight *increase* in the total weighted power, due to the impact of gates resizing on the consumption at 16-bit. However, decreasing the 8-bit supply voltage further, e.g. to 0.7V, might reduce the power consumption at that precision enough to improve the value of (3.11). The occurrence of this situation depends on the values of weights w_i, on the topology of the considered circuit, and on synthesis constraints (especially f_{clk}).

The advantage of the greedy approach is that it requires a smaller number of re-synthesis. Alternatively, an exhaustive search could also be performed, considering all possible supply voltages for each bit-width (among those smaller than the ones used for higher bit-widths). This would guarantee optimality at the cost of longer execution time, and it would still be feasible if the number of bit-width modes and V_{DD}s is not too large. In practice, in our test cases, the situation described above never verifies, and the greedy and exhaustive solutions coincide.

3.4.4 Experimental Results

Setup

To demonstrate the effectiveness of the methodology described above we have performed some experiments on a 16x16-bit MAC operator, with a 44-bit accumulator. We have described the architecture of the MAC in VHDL, and synthesized it targeting a 28nm FDSOI technology library from STMicroelectronics. The clock frequency and nominal supply voltage have been set to $f_{clk} = 1.1GHz$ and $V_{DD} = 0.95V$ respectively. In all experiments, we have considered supply voltages from $0.95V$ to $0.60V$ in steps of $0.05V$.

For logic synthesis we have used Synopsys Design Compiler L-2016.03, P&R has been executed with Cadence Innovus 16.1, while STA and power analyses have been performed in Synopsys PrimeTime L-2016.06. The algorithm of Figure 3.26 has been implemented in Python 3.5, and internally makes use of both Design Compiler and PrimeTime.

The choice of a MAC operator is motivated by the fact that this circuit is a basic building block of hardware accelerators for the inference phase of neural networks [97–99]. In the following, we have considered the case of designing a MAC operator contained in an accelerator for the popular LeNet-5 NN architecture, originally proposed for handwritten digit recognition tasks [1]. We have selected this benchmark due to its relevance, and the availability of previous bit-width characterizations. However, notice that our method is applicable to any other domain for which the number of operations performed at different bit-widths can be estimated through offline characterizations.

The authors of [48] have studied the impact of using different fixed-point bit-widths for performing MAC operations on the accuracy of popular NN models, including LeNet. They have shown that better accuracy is obtained if a different bit-width is used for each *layer* of the network. In our experiments, we have considered the same bit-widths derived in [48] for the two convolutional layers of LeNet-5.

Bit-Width	**Number of MACs**
16	$0.01 \cdot 10^6$
8	$1.6 \cdot 10^6$
4	$0.3 \cdot 10^6$

Table 3.5 Number of MAC operations performed at each bit-width during the classification of one digit in the LeNet-5 neural network [1].

Moreover, we have assumed that 16-bits are used for Fully Connected (FC) layers, in accordance to [173]. With these assumptions, the number of MAC operations performed at each bit-width for the classification of one digit using LeNet-5 is reported in Table 3.5.

In the following, we compare three different DVAS-based MAC operators, each supporting 4-bit, 8-bit and 16-bit modes. A first *Classic* version, used as baseline for comparison, has been obtained without synthesis optimizations, i.e. applying classic DVAS (a posteriori) to a circuit implemented with a standard flow. Additionally, we have generated two versions of the MAC using the algorithm of Figure 3.26. For the first, all weights w_i have been set equal to 1 (*Uniform* version). When generating the second version, instead, weights have been set to a value proportional to the usage probability of each bit-width mode, i.e. $w_{16} = 0.01$, $w_8 = 1.6$, $w_4 = 0.3$ (*Weighted* version).

Power, area and energy comparison

Table 3.6 shows the supply voltages used in reduced bit-width modes by the three MAC versions, as well as the area overheads with respect to the Classic version. Figure 3.27 shows the post-P&R total power consumption (including leakage and dynamic contributions) obtained by the three designs at each considered bit-width.

In the Classic MAC, scaling the V_{DD} to $0.90V$ causes timing violations at both 8-bit and 4-bit precision, due to the wall-of-slack. In the Uniform version, conversely, thanks to the multi-scenario optimization, the synthesis tool enforced timing compliance at $0.90V$ for 4 and 8 bit modes. The V_{DD} was not reduced further because of the high power overheads that this would cause at 16-bit. However, this intermediate solution does not take the statistics of application data into account. In particular,

MAC Version	**8-bit V_{DD} [V]**	**4-bit V_{DD} [V]**	**Area Ovr. [%]**
Classic	0.95	0.95	-
Uniform	0.90	0.90	9
Weighted	0.75	0.70	16

Table 3.6 Supply voltage V_{DD} for reduced bit-width operation and area overheads of three precision-configurable implementations of a MAC. Area overheads are normalized to the area of a standard MAC (*Classic* column).

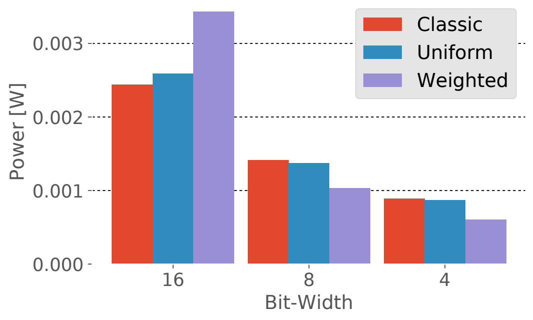


Fig. 3.27 Power consumption at different bit-widths for three precision-configurable implementations of a MAC.

it ignores that maximum-precision is only required in about 0.005% of the MAC operations.

Conversely, in the Weighted case, our algorithm selects a much more aggressively scaled V_{DD} for both 8 and 4-bit, since these two modes are much more relevant than 16-bit in LeNet inference. This causes an area overhead of 16% compared to the Classic version, but allows significant power savings at reduced bit-widths (27% at 8-bit, and 31% at 4-bit).

The advantage of the Weighted solution is demonstrated by Figure 3.28, which reports the total *energy* consumption due to MAC operations for classifying one digit in LeNet-5, if the three MAC versions are used. Results are normalized to the Classic implementation.

While the Uniform version only consumes $\approx 5\%$ less energy compared to a standard design, the Weighted solution reduces consumption by $\approx 27\%$. Importantly, this result also accounts for the additional leakage energy of the Weighted version, caused by the larger area.

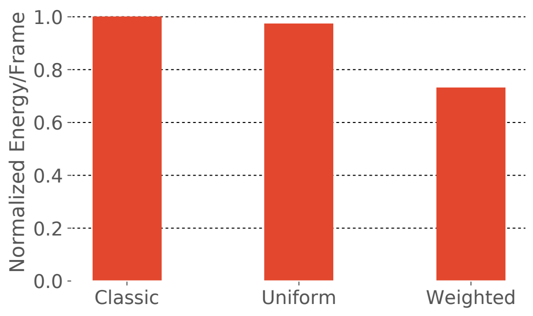


Fig. 3.28 Energy due to MAC operations for classifying one frame in LeNet-5 using three different precision-configurable implementations of a MAC.

Comparison with the fine-grain V_{th} approach

In Sections 3.3 and 3.4 we have described two different techniques to implement quality-configurable operators based on the principles of D(V)AS. Both solutions have been demonstrated superior to a straight-forward application of standard DVAS. As a final experiment, in this section we provide a comparison of these two methods, examining the advantages and limitations of each.

One example of such comparison is shown in Figure 3.29. To generate this plot, we have implemented the synthesis-based technique described in this part of the chapter on the same 16-bit Booth multiplier used in Section 3.3.8 to evaluate the method based on dynamic V_{th} tuning. For fairness, the set of supply voltages used for this experiment is the same of Section 3.3.8. The clock frequency has been set to $f_{clk} = 1GHz$, to ensure that also the circuit generated with incremental synthesis can operate at maximum precision without violations, despite not using FBB. The circuit based on fine-grain V_{th} tuning has been implemented using a regular grid of 2x2 domains. In the algorithm of Figure 3.26, we have targeted three precision configurations: 16, 8 and 6-bit. Moreover, in absence of information about the target application, we have set the power weights for each precision mode to 1, as for the Uniform MAC of the previous experiment. The plot of Figure 3.29 reports the power consumption of the two methods for different bit-widths, normalized to the power of the synthesis-based solution at 16-bit.

On average, the method based on fine-grain threshold voltage tuning (*Fine-grain* curve) obtains larger savings for this design. This is expected and it is motivated by the greater flexibility offered by dynamic V_{th} tuning, compared to the static optimization performed at synthesis time in the second method. In particular, the

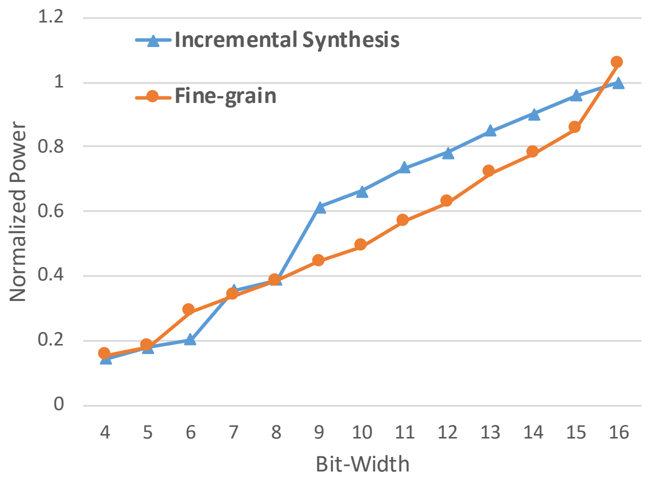


Fig. 3.29 Power consumption versus bit-width curves obtained on a 16-bit Booth multiplier using the two proposed techniques for designing quality-configurable hardware operators based on LSB gating.

fine-grain V_{th} approach does not optimize a specific set of precision configurations; instead, it provides a runtime knob that can be used to optimize power at *all* bit-widths. On the other hand, the incremental synthesis approach specifically optimizes a subset of precisions, by permitting a more aggressive voltage scaling than what would be possible in standard DVAS. By doing so, it achieves comparable power savings for its three targets, i.e. 16, 8 and 6-bit. However, in between these points, the only power reduction in the synthesis-based method is due to the reduction of switching activity caused by gated inputs. In contrast, the fine-grain method can optimize intermediate bit-widths by partially changing the back biasing assignment to some domains.

In terms of overheads, the incremental synthesis solution achieves the best results for this circuit. As shown on the plot, the power at maximum precision is smaller than that of the fine-grain approach. The same is true also for the area overhead compared to a standard Booth multiplier implementation, which is approximately 10.3%, compared to the 14.6% of the solution based on back biasing. This last difference is motivated by the strong impact of separation guardbands between BB domains in such a small circuit; the ranking of the two methods in terms of area overheads may change for larger operators.

In summary, none of the two solutions clearly outperforms the other. The method based on fine-grain V_{th} tuning offers the highest flexibility and largest power savings on average, at the cost of a not negligible area overhead. The incremental synthesis method is of interest when the target circuit will be operated in a limited set of quality-configurations. Moreover, it has the important advantage of not relying on technology-specific knobs, which makes it equally effective in all technologies. Finally, the incremental synthesis can also take into account a characterization of the target application when optimizing the design. While this makes the result less flexible, it may allow to obtain larger savings at a specific precision (e.g. 8-bit), if the latter is expected to be used most of the times.

3.5 Final Remarks

The three works described in this chapter propose different ways to design quality-configurable operators, each of which can be applied to a different set of design scenarios and technologies. Nonetheless, all three follow a similar automation-driven approach, and have as main goal the integration of EQ scalable design techniques within standard flows used in industrial-level EDA.

In all three cases, we have demonstrated that this kind of integration is possible. Most importantly, we have shown that the results obtained by our flows often differ significantly from those yielded by a naive application of the original ideas of RPR and DVAS (both improving and worsening them, depending on the scenario). This is, in our opinion, the main take-away of this chapter: in order to be really useful for building commercial products, any new design technique based on EQ scalability principles should be tested and integrated with industrial EDA flows, possibly through modifications of the latter. Additionally, the issues of generality and dynamic quality reconfigurability should be always taken into account.

To conclude the chapter, this section recaps some of the features of the proposed methods. A comparative analysis of the two flows for implementing quality-configurable hardware based on the principles of DVAS has been already presented in Section 3.4.4. Although both these solutions are significantly different from the one based on RPR, it is still possible to perform a high-level comparison among them. In the following, we subdivide this comparison in two parts. First, we discuss the different ways used by the two approaches to achieve EQ scalability. Second, we compare their overheads and application use cases. We refer to the work described in Section 3.2 as *RPR-based* and to the ones of Sections 3.3 and 3.4 as *DVAS-based*.

Type of EQ scaling

DVAS-based techniques achieve power savings by *systematically* reducing the precision of computations. Thus, reduced-quality modes almost *always* produce wrong results (except when the truncated portion of inputs is equal to 0). The magnitude of these errors is limited in the worst case by the amount of gated LSBs. For example, in a circuit whose maximum precision is N bits, the maximum relative error when working at $M < N$ bits using LSB-gating is equal to: $\frac{2^{N-M}-1}{2^{N-1}-1}$, assuming integer data represented in two's complement. This is a typical *fail small* approach, in which errors have a high rate, but are limited in magnitude (see Chapter 1).

The RPR-based solution achieves quality scaling by VOS. Assuming that the critical paths of a circuit are only seldom activated, a RPR architecture may produce outputs identical to those of a standard design most of the times. When large errors occur due to timing violations, they are mitigated by the EC block. Therefore, RPR is theoretically both a *fail small* and a *fail rare* approach, since errors occur sporadically, and when they do their magnitude is limited.

In practice, however, the occurrence of unmitigated errors in RPR is much more common then expected. This is again due to the wall-of-slack effect, which makes the majority of paths critical. We have introduced this phenomenon when discussing DVAS-based methods, showing how it prevents aggressive voltage scaling even when input bit-width is reduced. Although less dramatically, the same effect also negatively impacts RPR. Indeed, when working in VOS, the wall-of-slack causes timing violations to happen not only on MSBs, but also on some LSBs. Given the decision scheme of RPR, the latter could remain undetected, causing *small* but *frequent* errors on the final output [3]. However, these errors are still smaller than the systematic ones produced by DVAS-based circuits, on average. Therefore, from the point of view of pure output behavior, RPR is definitely the most effective technique. One simple justification comes from the consideration that the output function of the replica in RPR is basically the same of a DVAS-based circuit operating at reduced-precision. However, while the latter is used as is, the former is only used as a "last resort", to recover from a large MDSP timing error.

Another fundamental difference between the two approaches is that it is not possible to set a theoretically proven upper bound on the error committed by a

[3]These errors are taken into account by our simulation-based flow.

RPR circuit implemented with our flow. In fact, as explained in Section 3.2, the detection circuit parameters are determined based on a finite-length simulation. When different inputs are applied to the circuit, some timing errors in the MDSP may remain undetected. Clearly, if simulation inputs are representative of the final application, the average output quality will not deviate substantially from the expected.

Overheads and Use Cases

One aspect where DVAS-based approaches are definitely superior to RPR is cost, as clearly shown by the results presented in Sections 3.2.5, 3.3.8 and 3.4.4 (although obtained on different technologies). In fact, while DVAS-based circuits require minimal additional hardware compared to a standard design (e.g. BB generators or larger gates for contrasting the wall-of-slack), RPR almost invariably leads to *doubling* the size of a circuit. These area overheads also translate into reduced energy benefits, due to the contribution of the replica and decision block on the total consumption of the system.

Another advantage of DVAS-based approaches is the *flexibility* they offer for reconfiguring power consumption and quality at runtime. Indeed, these solutions offer a very large number of possible quality modes (e.g. a 16-bit datapath operator may theoretically work in 16 different modes). In contrast, RPR only allows two modes of operation, respectively at nominal and VOS supply voltage. Consequently, although RPR-based circuits still offer some reconfigurability to perform accurate computations, they are destined to application-specific hardware, for which the desired output quality can be determined at design time. DVAS-based operators, in contrast, could be embedded in more general purpose hardware for error resilient applications (e.g. an error resilient processor such as [95]).

In conclusion, despite their similarities, the methodologies proposed in this chapter have very different metrics and application use cases. Consequently, there is also not a straight-forward way to combine them into a single technique, so that the benefits of both approaches are combined.

The choice among one approach or the other boils down to the analysis performed in this section: the smaller overheads of DVAS-based methods come at the cost of a larger number of errors, and a generally lower output quality. However, these methods provide higher reconfiguration flexibility and a larger set of operating modes. Therefore, the techniques presented in this chapter are positioned in different corners of the cost versus benefits design space, as exemplified in the qualitative diagram of Figure 3.30.

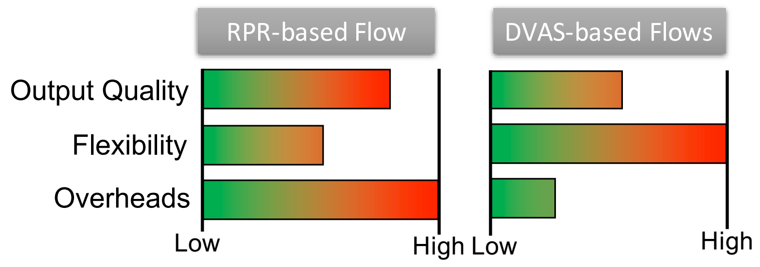


Fig. 3.30 Qualitative comparison of the different methods proposed for the design of quality-configurable processing hardware.

Chapter 4

Approximate Bus Encodings for Quality-Configurable Serial I/O

In this chapter, we describe two different encoding techniques for reducing energy consumption in serial I/O buses. A general background on serial bus encodings has been presented in Section 2.6.2. Both our proposed encodings are *lossy*; as such, they achieve large reductions in the number of level transitions on the bus (which translate into power savings) in exchange for some inaccuracies in the decoded data. A dynamic reconfiguration of the tradeoff between error and power is also possible by tuning the parameters of both encodings. The content presented in this chapter is an extended and revised version of our previous publications found in [51–54].

4.1 Motivation

As anticipated in Chapter 2, serial communication links are ubiquitous in modern digital systems. On-chip, they are employed within some modern NoC designs; off-chip, they are the de-facto standard for interconnecting processing elements with I/O peripherals.

Even a single off-chip serial bus can be a significant contributors to the total power consumption of a device. Indeed, the average energy consumption of a PCB trace is in the order of 1-2 pJ/bit/inch [132]. For a trace of a few centimeters, this corresponds to about 10 pJ of energy for the transmission of a single bit of information. In comparison, an ultra low-power processor for IoT sensor nodes may have an active current of about 50-100 μA/MHz, which translates to 10-20 pJ/instruction at a typical clock frequency [174, 175]. Therefore, the transmission of a single bit on a serial bus may consume roughly the same energy as the execution of one processing instruction. This clearly motivates the need for energy efficient serial I/O, especially when considering that a complex system may easily contain tens of peripherals, each connected serially to the processing logic.

One kind of interface where serial buses are commonly found is that between processing elements and sensors. In this scenario, the EQ scaling design paradigm can be adopted for reducing the power consumption of the link. In fact, the data produced by a sensor is typically error tolerant, being already affected by external noise and analog to digital conversion errors (see Chapter 1). Furthermore, sensor nodes typically have very limited power budgets, being battery operated and often self-powered, and targeting long autonomy cycles (from several months to several years). Another type of interface with similar characteristics is that connecting actuators such as displays or speakers, whose outputs are destined for human consumption.

Lossy bus encodings can be adopted in these interfaces, in order to achieve large power savings at the expense of some approximations on the decoded data. In order to be effective, a lossy encoding may consider the features of the data being transmitted. Indeed, both the data produced by sensors and those directed to "multimedia" peripherals tend to have very well-defined dynamics. Moreover, as for most EQ scalable systems, the amount of tolerable error in such a connection is not invariable. For instance, it may be influenced by environmental conditions (amount of external noise, range of variation of the measured quantity, etc.), as well as system status (battery state of charge, usage context, etc.), and application target (precise measurement versus rough estimate). Therefore, a lossy serial bus encoding should allow dynamic reconfiguration of the amount of error committed during transmission.

In this chapter, we propose two different lossy encodings for serial buses, called *Approximate Differential Encoding* (ADE) and *Serial-T0* (ST0), that leverage data dynamics and permit runtime reconfiguration of the error. The former targets energy reduction in generic interfaces from sensors and to actuators, exploiting the *temporal correlation* of the serial data traces produces by these devices. The latter specifically targets image data, and leverages the *bursty* data of the corresponding traces.

For each encoding, we also propose a set of variants for specific use cases (e.g. DC-balanced connections). We show that the encoding and decoding (CODEC) functions of these methods can be implemented with very simple hardware, incurring minimal energy overheads compared to the savings on the bus. Additionally, we discuss the integration of our encodings with standard serial protocols.

4.2 Serial Buses Power Consumption Model

An off-chip serial connection can be modeled, in first approximation, as a purely capacitive load, due to the large pitch of PCB lines [126, 129–132]. According to this model, all power dissipation occurs in correspondence of electrical level changes, i.e., the total consumption is approximately equal to the dynamic power. Therefore, the power consumption on the channel is:

$$P_{chan} = P_{dyn} = \alpha \cdot C_{tot} \cdot V_{swing}^2 \cdot f \tag{4.1}$$

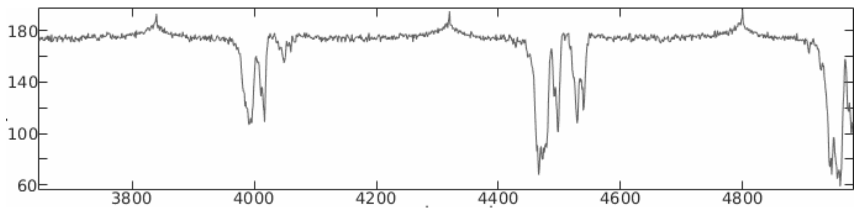
In this expression, C_{tot} represents the total channel capacitance, accounting for driver, pin and wire contributions, V_{swing} is the voltage swing between electrical levels and f is the signaling frequency. Finally, $\alpha \in [0,1]$ is the *switching activity factor*, i.e. the average probability of occurrence of a logic value transition in a clock cycle.

Serial bus encodings achieve power savings by reducing the number of transitions on the bus, that is by acting on α. In the following, we refer to the total number of transitions on a serial line during the transmission of a sequence of words as the *Transition Count* (TC) of that sequence.

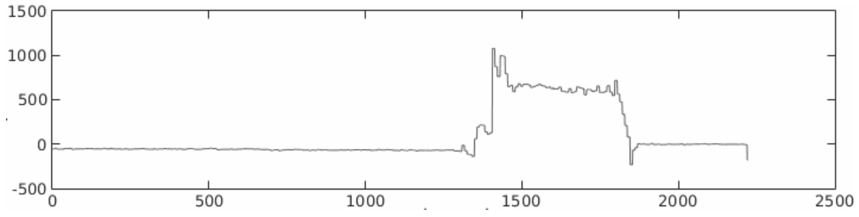
4.3 Common Features in Error-Tolerant Serial Bus Traces

Data streams coming from sensors or transmitted to audio and video peripherals may have different ranges of variation and rates. However, they often share one common characteristic, which is that of (intermittent) *temporal correlation*. In general, temporal correlation refers to the dependency among values transmitted subsequently in time on the bus. In information theory terminology, this corresponds to saying that the sequence of words transmitted on the bus is not a *Discrete Memoryless Source* (DMS) [176]. In practice, this correlation manifests as a *numerical similarity* among subsequently transmitted values, deriving from the physical nature of the considered quantities. In many cases correlation is intermittent, meaning that the data trace is composed of relatively long regions with strong similarity among subsequent words, alternated with sporadic regions of faster and larger variations. We will refer to these two types of behaviors with the informal terms of *correlated regions* and *bursty regions* respectively.

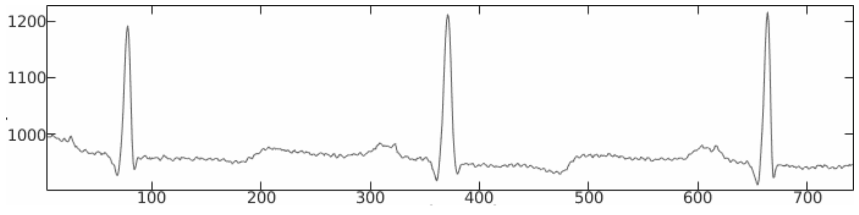
Three examples of serial data correlation, taken from the bus traces produced by real peripherals are shown in Figure 4.1.



(a) CCD-Camera bus trace [177].



(b) Accelerometer (x-channel) bus trace [178].



(c) ECG bus trace [179].

Fig. 4.1 Examples of serial bus traces produced by real sensors

The uppermost graph shows the trace generated by one of the three RGB color channels of a CCD camera. In this case, correlated regions represent sections of the image in which the color is almost constant, whereas bursty regions (e.g., around the 4000-th word) represent lines or boundaries between two different colors. While this sequence refers to the acquisition of an image, the transmission of serial data

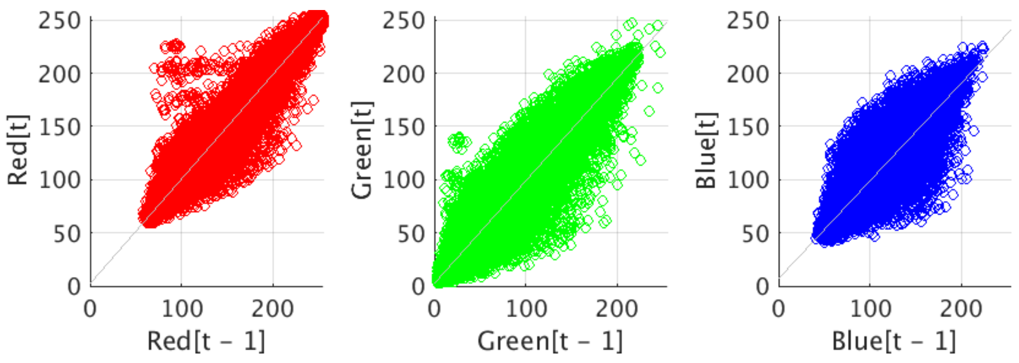


Fig. 4.2 Pixel-to-pixel magnitude correlation in serialized image data.

towards a display clearly follows a similar pattern. The second graph shows the data generated by one of the axes of an accelerometer. This sequence is strongly correlated when the device is idle or moving at a constant speed, while large variations occur correspondence of sudden movements (e.g., the interval between the 1300-th and 1800-th words). The lowermost graph shows a biomedical signal taken from an *Electrocardiogram* (ECG). Therein, correlated and bursty regions correspond to the intervals between heartbeats and to the pulses themselves, respectively.

A further example is provided in Figure 4.2, for the case of image data (camera sensor or display). In particular, the three scatter plots refer to the three RGB channels of the *Lena* image [177]. Each plot relates the color components of the pixels sent on the bus at time $t-1$ and at time t, assuming that the image is sliced by rows for serialization. The correlation coefficients among subsequent pixels are ≥ 0.91 for all three color channels.

Analyzing public databases for sensor and audiovisual data [179, 178, 177, 180], correlated regions tend to be much longer than bursty ones. Therefore, following the typical "common-case" approach of low-power design, an encoding for this type of data should reduce power especially in correlated regions. Additionally, these are also the regions for which a larger error (in relative terms) is acceptable. In fact, most of the information content is carried by bursty regions, and correlated ones can be approximated with less impact on the overall quality. An example is given once again by the traces in Figure 4.1: evidently, the information on lines/color changes in an image is more important then the exact RGB value of a pixel. Similarly, the occurrence of a sudden event is more important than the exact acceleration value in an accelerometer.

4.4 Approximate Differential Encoding for Correlated Sensor Data

In this section, we describe ADE and its variants. ADE is the EQ scalable version of the previously proposed lossless *Differential Encoding* (DE), which has been proven effective in reducing the number of serial bus transitions for correlated data [126]. On top of the principles of DE, we add a simple data approximation scheme that further improves the TC reduction, by leveraging the observation that small magnitude errors in highly-correlated data are often acceptable.

We also present a set of variants of ADE, to be used in specific scenarios, i.e. when preserving small data fluctuations is fundamental and when the physical connection requires a balance among electrical values (DC balancing). Finally we describe how the proposed encoding can be jointly optimized with the *Analog to Digital Converters* (ADC) included in sensors, to further reduce energy consumption.

Using realistic data traces generated by error resilient applications, in Section 4.6 we will demonstrate that ADE achieves competitive results both in terms of sheer TC reduction, as well as in terms of total energy savings, also considering the overheads due to the encoding and decoding circuitry.

4.4.1 Differential Encoding

DE is a simple yet very effective way to reduce the number of transitions, hence the total power consumption, produced by a serialized stream of correlated data. One effective use of this method is found within the SILENT algorithm described in [126], which has been been already summarized in Chapter 2. In this section, we present the functionality of DE in a more detailed way.

DE works by transmitting on the bus the *bitwise difference* between the current and previous words. Accordingly, the bits of a codeword are generated using the following equation:

$$B_i[t] = b_i[t] \oplus b_i[t-1],\ \forall i \in [1,n] \tag{4.2}$$

where $b_i[t]$ is the i-th bit of the data word to be transmitted at time t, and $B_i[t]$ is the corresponding codeword bit. The decoding expression for DE is as follows:

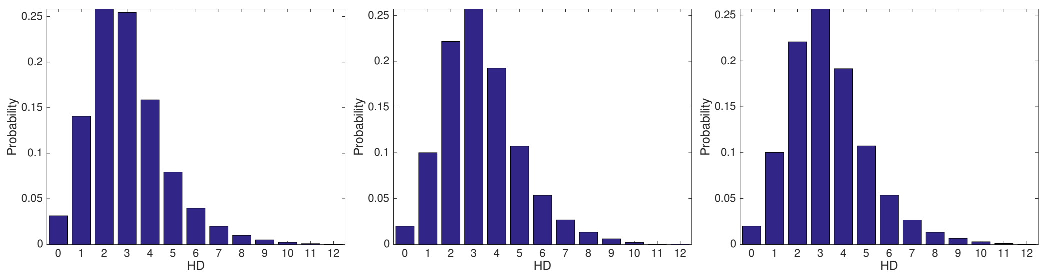
$$b_i[t] = B_i[t] \oplus b_i[t-1],\ \forall i \in [1,n] \tag{4.3}$$

DE is effective due to the fact that data transmitted by sensors or received by audio/video devices are normally represented in positional notation. Common positional formats include unsigned binary, *Magnitude and Sign* (M&S) and two's complement, with integer or fixed point semantics. These formats are used to represent physical quantities in sensors, as well as sound and color intensities.

For positional data, numerically close values such as those found in subsequently transmitted words of a correlated stream tend to have a very small Hamming distance. This is because the MSBs of two numerically similar positional values tend to be equal, which makes the HD among the two small (e.g. $\underline{001010}00$ and $\underline{001010}11$ in pure binary, corresponding to 40 and 43 in decimal). One exception occurs for two's complement data with similar magnitude and different sign, case in which MSBs tend to be *all* different (e.g. $\underline{00000}101$ and $\underline{11111}000$ corresponding to $+5$ and -8), and the corresponding HD is large. Finally, there are also conditions for which a small value difference does not translate into a small HD, for example when the two values are respectively smaller and larger than a power of two, or vice versa (e.g. 00111111 and 01000000).

The empirical *Probability Mass Functions* (PMFs) of the Hamming distance obtained for strongly correlated data represented in the three most common base-2 positional formats is shown in Figure 4.3. These PMFs have been obtained computing the HD among 1 million of pairs of 12-bit words. The first word of each pair has been drawn randomly for the entire range of values, while the second has been picked within a range of ± 25 (in decimal) from the first, corresponding to $\pm 0.6\%$ of the full scale. The plots show that, regardless of the chosen format, the first of the three conditions explained above (i.e. a small HD due to value similarity) is by far the most common. Indeed, approximately 85% of the word pairs have an HD ≤ 4.

When the HD between words is very small or very large, the codewords generated by DE are characterized by a long sequence of 0s or 1s respectively, in the MSB positions. Since this sequence does not have any value transition when serialized, the overall TC obtained transmitting it will be significantly smaller than that obtained by



(a) Unsigned binary (b) Sign and Magnitude (c) Two's complement

Fig. 4.3 Hamming distance PMF for 1 million pairs of random 12-bit words with absolute value difference $|w_1 - w_2| \leq 50$.

raw data. Given the results of Figure 4.3, such a TC reduction will be obtained for the large majority of the words, when encoding a strongly correlated trace.

4.4.2 Limitations of DE

Figure 4.4 shows the qualitative dependence between the average HD between subsequently transmitted words in a sequence and the average transition count of the corresponding DE codewords, as presented in [126]. The graph also reports the TC of unencoded input data, which is roughly independent from the HD. As mentioned in the previous section, DE reduces the TC for small or large HDs; however, it tends to *increase* the TC, hence incurring a power overhead, when the HD is about half of the word length (n).

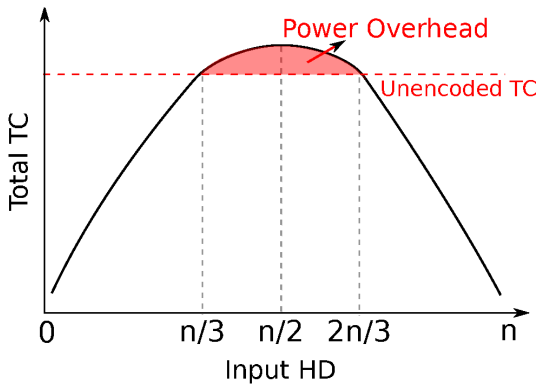


Fig. 4.4 Input Hamming distance versus transition count for lossless differential encoding.

This overhead is particularly relevant when DE is used to encode data transmitted from/to the error resilient peripherals mentioned above. In fact, most of these data are represented on small bit-widths. For example, 8-12 bits are commonly used to store samples from sensors, and image pixels are rarely represented with more than 8-bit per color component [122]. When considering such small bit-widths, even if the transmitted data is highly correlated, many pairs of words will have an HD falling exactly in the overhead range of DE (i.e. $\frac{n}{3} < HD < \frac{2n}{3}$).

Figure 4.5a reports one example of this phenomenon. The figure shows the HD PMF obtained from serialized pairs of pixel components in the RGB image *Lena* [177]. Despite the good correlation among subsequent pixels, demonstrated in Figure 4.2, about 59% of the transmitted pairs of words have HD$\in [3:5]$. For these words, differential encoding will not reduce the number of transitions on the bus.

The work of [127] and others have proposed ways to solve this issue while still providing a lossless encoding scheme. However, these solutions require redundant bits, which add large overheads to serial buses, either in terms of throughput or cost, as detailed in Section 2.6.2. Approximate DE (ADE), in contrast, provides a low-overhead solution by transforming DE into a lossy encoding, as explained in the next section.

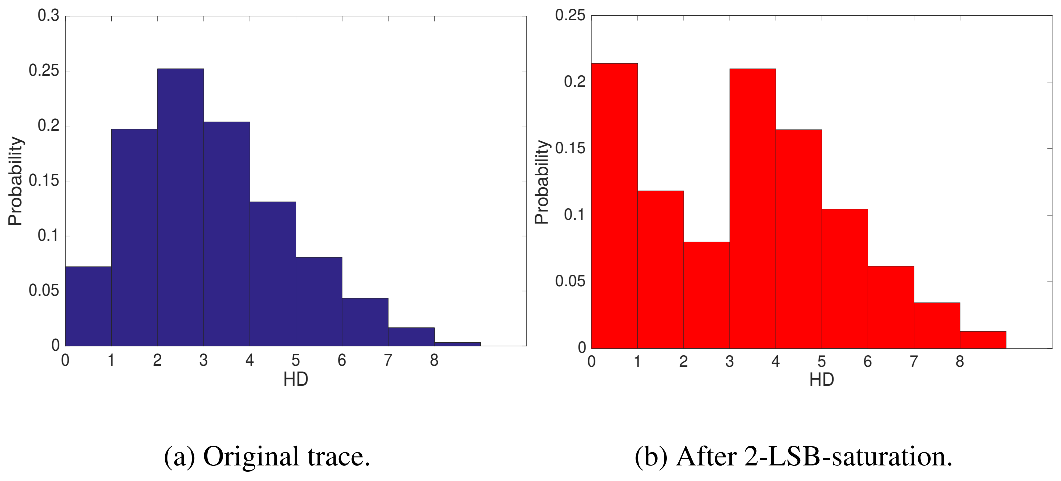


Fig. 4.5 PMF of the Hamming distance between consequent words in the bus trace produced by the *Lena* image, before and after LSB-saturation.

4.4.3 Approximate Differential Encoding

We propose to solve the limitations of DE described above by leveraging the error tolerance quality exhibited by many of the applications that involve serial links. The

```
1: procedure ADE ENCODING(w = data word of length n, l = n. of saturated bits)
2:     // Saturation Phase
3:     for $i \in [1,l]$ do
4:         $t_i[t] = w_{l+1}[t]$
5:     end for
6:     for $i \in [l+1,n]$ do
7:         $t_i[t] = w_i[t]$
8:     end for
9:     // Bitwise Difference Phase
10:    for $i \in [1,n]$ do
11:        $B_i[t] = t_i[t] \oplus t_i[t-1]$
12:    end for
13: end procedure
```

Fig. 4.6 Proposed ADE encoding algorithm.

basic principle of our encoding, ADE, is to *reshape the HD distribution* of the input data pairs. By doing so, HD values which are favorable for DE are made more frequent, and the overall TC reduction improves. To achieve this objective, the basic form of ADE simply saturates some LSBs of the input words to all 1s or all 0s, before encoding the word with accurate DE. Thanks to saturation, the least significant part of the ADE codeword will not introduce any transition, being composed of either all zeros or all ones.

This simple idea is very effective in reshaping the HD distribution. As an example, Figure 4.5b shows the HD PMF obtained after saturating 2 LSBs of each word in the data trace generated by the *Lena* image. Compared to the original PMF of Figure 4.5a, this new distribution is visibly more skewed towards HD= 0. The number of word pairs with an unfavorable HD is reduced from 59% to 45%.

The number of saturated LSBs, called l hereinafter, allows to trade-off power consumption for output quality in ADE. If n is the bit-width of the input data, the encoding procedure of ADE is implemented as follows:

$$\begin{aligned} B_i[t] &= b_{l+1}[t] \oplus b_{l+1}[t-1], \ \forall i \in [1,l] \\ B_i[t] &= b_i[t] \oplus b_i[t-1], \ \forall i \in [l+1,n] \end{aligned} \tag{4.4}$$

where $b_n[t]$ represents the MSB and $b_1[t]$ is the LSB. A pseudo-code implementation of ADE encoding is reported in Figure 4.6.

The decoding procedure of ADE is identical to the one of lossless differential encoding, i.e. it is implemented using (4.3). Notice that the decoder does not need to be informed of the value of l, hence no parameter handshaking is needed between transmitter and receiver when the accepted error changes.

The maximum absolute error introduced by ADE for integer data is $E_{MAX} = 2^l - 1$. In the example of Figure 4.5b, the saturation of 2 LSBs introduces a maximum error of: $2^2 - 1 = 3$, about 1.2% of the maximum value representable on 8-bit (255). Through the experiments shown later, we will demonstrate that this error produces a minimal reduction of visual quality on the decoded image. In exchange for this small error, ADE significantly reduces the TC on a serial link, both with respect to unencoded data and to lossless differential encoding. For the *Lena* example, transitions are reduced of about 61%.

Example of Operation

Input		DE		LSBS			ADE		
Word	TC	Word	TC	Word	TC	$\lvert E \rvert$	Word	TC	$\lvert E \rvert$
00001011	4	00001011	4	00001000	2	3	00001000	2	3
00001111	2	00000100	2	00001111	2	0	00000111	2	0
00001101	4	00000010	2	00001111	2	2	00000000	0	2
00001101	4	00000000	0	00001111	2	2	00000000	0	2
00010111	4	00011010	4	00010111	4	0	00011000	2	0
00100011	4	00110100	4	00100000	2	3	00110111	4	3
00000100	2	00100111	4	00000111	2	3	00100111	4	3
00001101	4	00001001	4	00001111	2	2	00001000	2	2
00001110	2	00000011	2	00001111	2	1	00000000	0	1
00001011	3	00000101	3	00001000	2	3	00000111	1	3
Totals	33		29		22	19		17	19

Table 4.1 Example of operation of the ADE serial bus encoding for a trace of 8-bit unsigned data.

An example of operation of ADE for 8-bit unsigned binary data is reported in Table 4.1. The table shows the input data trace in the leftmost column pair, whereas ADE codewords are found on the rightmost set of columns. The two remaining column sets are included for sake of comparison, and report the outputs of conventional DE and *LSB Saturation* (LSBS) for the same inputs. In this example, both LSBS and ADE saturate 3 LSBs of each word, which corresponds to allowing a maximum relative error of 2.7% for 8-bit inputs. Columns labeled TC report, for each encoding, the transition count generated when the corresponding words are transmitted on a serial bus, assuming a LSB-first protocol and including also inter-word transitions. Column |E| reports the absolute error introduced by approximate encodings (LSBS and ADE) on each word.

As shown in the example, ADE outperforms both LSBS and lossless DE, by combining the features of these two algorithm. The total transition count is reduced of $\frac{33-17}{33} \approx 48\%$, whereas the average absolute error per word is just $\frac{19}{10} = 1.9$, which corresponds to less than 0.75% of the full-scale value (255).

Comparison with Reduced Bit-width Transmission

In this section, we compare ADE to a reduced bit-width serial transmission based on lossless DE, that simply skips the transmission of the l LSBs. In particular, we show that the former is more flexible, despite introducing the exact same amount of error.

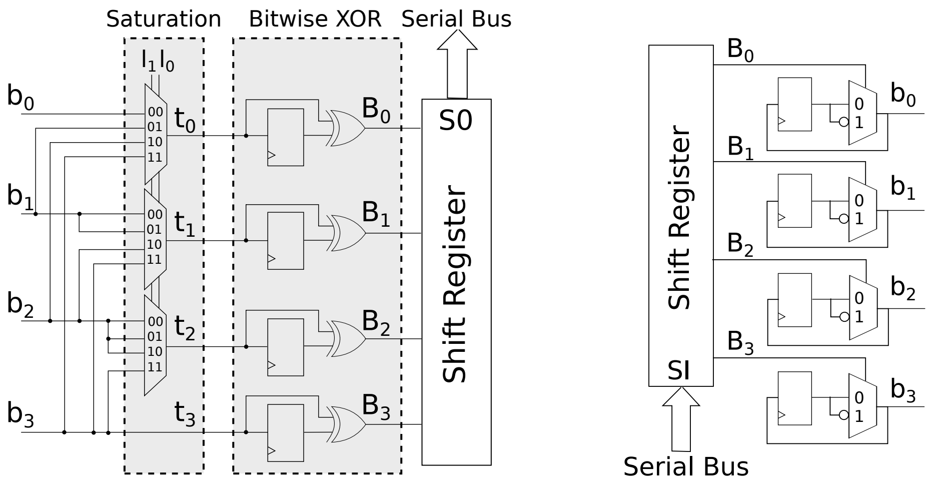
One obvious advantage of ADE is the possibility of changing the amount of error introduced in codewords at runtime. This can be done very easily simply by changing the parameter l in the encoding procedure, in response to changes in the error tolerance of the running application, or in the conditions of the system (e.g. battery status). Throughout these changes, the codeword bit-width remains constant and equal to n. Conversely, achieving the same result with reduced bit-width transmission requires a *variable word length* transmission.

Therefore, the ADE approach is easy to integrate with standard serial protocols (e.g. SPI, I2C, etc.), which rely on a fixed length scheme of transmission, without having to change the I/O interfaces of existing peripherals. Variable width transmission, on the other hand, results in significantly more complex transmitter and receiver circuitry, and is also impossible to integrate with legacy devices based on standard protocols. For example, SPI-based devices [123] often have *fixed-size* MOSI/MISO registers. Other protocols, such as I2C [124], support variable-width transmission only in multiples of one byte, a too coarse granularity to allow a fine tuning of quality and energy.

Another reason why ADE is more flexible than reducing the transmission bit-width is that error reconfiguration can happen independently from the receiver, as the latter does not need to know the number of saturated bits l. In contrast, for a variable length transmission, even if the underlying protocol supports it, transmitter and receiver should share the knowledge of the current word length. Therefore, a parameter handshaking phase is needed every time the amount of acceptable approximation changes.

4.4.4 Hardware Implementation

One of the advantages of ADE with respect to other lossy serial bus encodings (e.g. [132]) is that its encoding and decoding algorithms can be implemented very easily in hardware. One possible implementation is shown in the form of block diagrams in Figure 4.7. The encoder and decoder represented in this figure refer to a 4-bit implementation, for ease of visualization.



(a) ADE encoder. (b) ADE decoder.

Fig. 4.7 Hardware implementation of the proposed ADE serial bus encoding.

The encoder hardware is conceptually organized in two main sections, highlighted by gray dashed areas in the figure, corresponding to the two phases of the algorithm in Figure 4.6.

The *Saturation* section implements LSB rounding, and corresponds to lines 2-8 of the pseudocode. It is composed of an array of multiplexers, each of which determines if one of the bits in the input word (b_i) should be forwarded to the next section, or replaced with a bit of higher significance. The selection signals for these multiplexers (l_j) are the bits that compose the base-2 representation of l, the main parameter of ADE, introduced in (4.4). For instance, if $l = 1$, meaning that 1-bit is saturated, then $l_0 = 1, l_1 = 0$ and each multiplexer selects the second input form the top, resulting in $t_0 = b_1$, and $t_i = b_i, \forall i \geq 1$. For a generic n-bit word, this part of the encoder is composed of n instances of a n-way multiplexer, each having $\log_2(n)$ control inputs, in order to represent all possible values of l, from $l = 0$ to $l = n - 1$.

The second section of the encoder, labeled *Bitwise XOR*, implements word-level differential encoding, taking as input the intermediate saturated word. This section corresponds to lines 9-12 of Figure 4.6. The saturated data word transmitted in the previous clock cycle is stored in an array of flip-flops, and used to perform a bitwise XOR operation with the current datum. The result of the XOR is loaded into a shift register which acts as serializer.

The decoder hardware is even simpler, as shown in Figure 4.7. The previously decoded word is stored in a bank of flip-flops, initialized with zeros at the beginning of a transmission. When a new word is received and deserialized, an array of multiplexers selects between the memorized value and its negation, depending on the corresponding bit read from the parallel output of the shift register. As mentioned in Section 4.4.3, this hardware is independent of the parameter l.

Both circuits in Figure 4.7 have a latency of one clock cycle [1]. Moreover, they have a throughput of one encoded/decoder word per cycle. This means that (assuming that the encoding and decoding circuits operate synchronously with the transmitter and receiver) implementing ADE does not introduce any performance overhead in the serial communication. Moreover, area and power consumption of these hardware implementations are very limited, since both blocks are composed of very simple logic and memory elements.

4.4.5 Selective ADE for Small Variation Preservation

ADE achieves power savings by eliminating small variations on the input data through LSB saturation. However, there are contexts in which small variations in

[1]The latency for serialization and deserialization is not considered since it is present in any serial transmission.

a signal are important, especially when the amplitude of the signal itself is small. Indeed, while small errors on the LSBs may be tolerable for signals that exhibit large variations (close to the full dynamic range), when the amplitude is small the LSBs become the carriers of the essential information.

An example of such situation is the transmission of digital audio data containing the recording of a person speaking. When the sound amplitude is large (high volume), the noise introduced by LSB approximations can be tolerated, as it does not impact the intelligibility of words. On the other hand, when the amplitude is small (for example because the speaker moved away from the microphone), the same amount of noise may prevent a correct understanding.

In situations like the one above, a modified version of ADE can be used in order to better preserve small variations in the signal, at the expense of smaller power savings, yet still greater or at worst equal to those achieved by accurate DE. This variant, called *Selective ADE* (SADE), only accepts data approximations through LSB saturation when the input signal varies significantly, and reverts to accurate DE when the variation is small. Formally, SADE implements the following equation:

$$B[t] = \begin{cases} DE(b[t], b[t-1]) \text{ if } \|b[t] - b[t-1]\| \leq T_h \\ ADE(b[t], b[t-1], l) \text{ otherwise} \end{cases} \tag{4.5}$$

where T_h is a user-defined threshold and $DE()/ADE()$ correspond to equations (4.2) and (4.4) respectively.

As for standard ADE, also SADE can be implemented efficiently in hardware. A high-level block diagram of the encoder for this variant is shown in Figure 4.8, where the additional components with respect to the circuit of Figure 4.7 are highlighted in gray. The decoding hardware for SADE is exactly equal to that of DE and ADE.

The threshold T_h in SADE must be set according to the application requirements, and is in principle independent from the number l of saturated bits. However, in many cases, it might make sense to tune these two parameters jointly. In particular, setting T_h as follows:

$$T_h = K \cdot (2^l - 1) \tag{4.6}$$

corresponds to implicitly accepting a maximum error equal to $K \cdot 100\%$ of the current variation of the signal. For example, if $K = 1$, the encoding saturates l bits (i.e., generates a maximum error of $2^l - 1$), only when the variation of the signal is larger

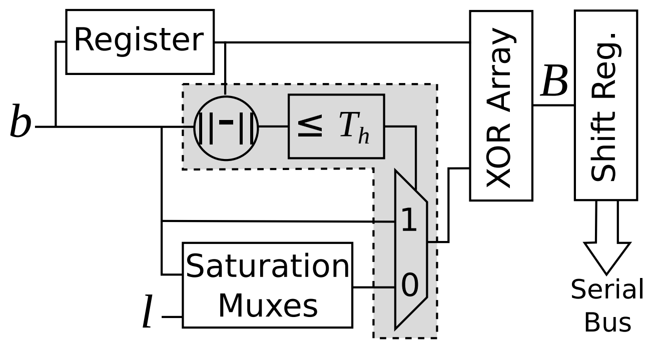


Fig. 4.8 Encoder hardware for the proposed Selective ADE algorithm.

than $2^l - 1$. In other words, the error is accepted when it alters the variation of the signal of less than 100%.

The functionality of the original ADE algorithm can also be obtained using SADE, if the threshold is set to $T_h = 0$. Therefore, a single software or hardware implementation can account for both alternatives, and choose among the two on-the-fly setting the parameters appropriately.

It is easy to see that the savings obtained by SADE are always intermediate between those of ADE and DE, for any input sequence. In particular, the worst case scenario for SADE is when the input signal is *only* composed of strongly correlated data, with similar magnitude among subsequently transmitted words. In this case, the second expression of (4.5) is never executed, and SADE is unable to introduce approximations, reducing to accurate DE. Conversely, in order for SADE to be effective, the signal should include sufficiently long *bursty* phases.

The concept of SADE can be further extended, generalizing the relation between the amount of approximation and the local variation of the signal. A scheme of this generalized encoder is shown in Figure 4.9. In this block diagram, the saturation parameter l used in ADE encoding is obtained as a generic function $f()$ of the current data variation and, possibly, of an external parameter l_{ext}. For example, rather than a “binary” decision between transmitting accurately or approximately, one could think of using *multiple thresholds*, each enabling a different degree of approximation (i.e., a different value of l).

A specific variant of this scheme, called n-SADE, generalizes (4.6) to n thresholds, where n is the length of a data word. In practice, instead of setting a single

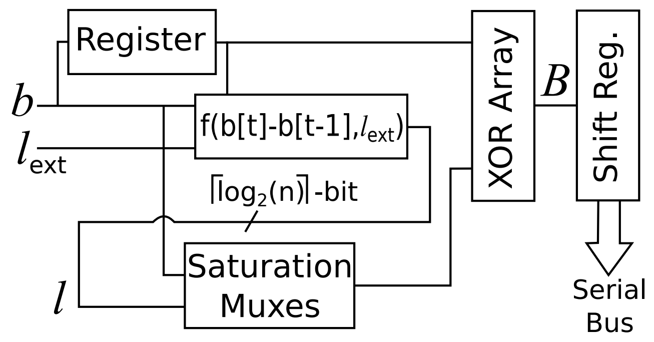


Fig. 4.9 Encoder hardware for the proposed generalized SADE and n-SADE algorithms.

value of l and a threshold T_h, as in basic SADE, quality is controlled by imposing a maximum ratio K between the variation of data and the allowed error:

$$E_{MAX} = \frac{||b[t] - b[t-1]||}{K} \tag{4.7}$$

Then, for every new word to be encoded, the appropriate value of l is obtained from the local data variation as follows:

$$l = \lfloor \log_2 E_{MAX} + 1 \rfloor = \lfloor \log_2 \frac{||b[t] - b[t-1]||}{K} + 1 \rfloor \tag{4.8}$$

The scheme of (4.8) can be implemented in practice using n thresholds, corresponding to increasingly larger values of data variation, and to an increasing number of saturated bits l.

Generalizations of SADE tend to increase the complexity in the encoding circuitry. For the particular case of n-SADE, described above, an implementation requires n comparators, one for each threshold. Simpler implementations are possible, but only for specific values of the parameters (e.g., if K is a power of two). The experiments that we performed on n-SADE did not show a consistent improvement in terms of power saving at iso-quality with respect to basic SADE, especially considering the increased overheads due to the encoder and decoder. However, the possible relations encapsulated by $f()$ in Figure 4.9 are virtually infinite. A more in-depth analysis of SADE generalizations will the subject of future work.

4.4.6 ADE for DC Balanced Connections

In the context of bus encodings, DC balancing consists in ensuring that the sequence of data transmitted on the bus contains approximately the same number of logic-0s and logic-1s, and in limiting the length of sequences of bits with the same logic value. This is important for AC-coupled connections, which act as high-pass filters, cutting off the DC component of the signal. Indeed, a strong prevalence of 0s or 1s in the data sent on such connections biases the DC component of the signal, and may cause decoding errors.

Most standard serial bus protocols do not specify any constraint relative to DC balancing on the adopted encoding of data. Moreover, PCB connections are mostly DC-coupled; hence, the problem of DC balancing is not relevant for the most common use cases of our encoding. Nonetheless, we show that ADE and its variants can be easily adapted to support DC balancing if needed.

By construction, ADE produces codewords that contain more 0s than 1s for correlated data, since the output of the bitwise XOR phase will mostly contains 0s in its MSBs. One example is shown in the two leftmost bars of Figure 4.10, which report the ratio between logic-0s and logic-1s in the serial stream produced by the pixels of the *Lena* image, when the latter is transmitted unencoded (left), and encoded with standard ADE with $l = 2$ (center). While the logic values in the raw

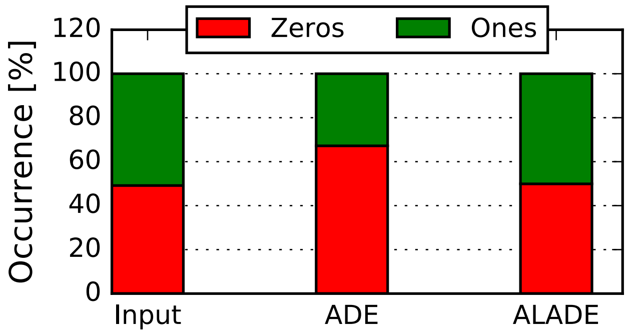


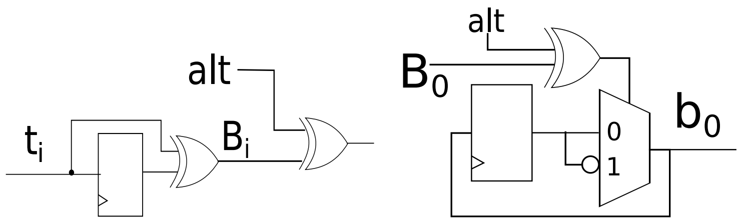
Fig. 4.10 DC balancing in the proposed ADE serial bus encoding.

image are almost perfectly balanced (49.1% of zeros), codewords exhibit a clear prevalence of logic-0s (67.2%).

To restore balancing, we start from the observation that ADE could be equivalently implemented replacing the bitwise difference operation with the *bitwise* *equality* (i.e. using XNOR gates instead of XORs in hardware). In fact, bitwise equality between consecutive correlated words would produce long strings of 1s instead of 0s in the MSB-part, but the transition count reduction would be exactly the same.

Therefore, we propose *Alternate ADE* (ALADE) as a DC balanced variant of ADE, in which words are encoded alternating difference and equality operators (XORs and XNORs) on a word-by-word basis.

The hardware must be slightly modified to support ALADE; one possible implementation of 1-bit encoder and decoder circuits for ALADE is reported in Figure. 4.11. In the encoder, only the *bitwise XOR* section of Figure 4.7 is affected.



(a) Encoder. (b) Decoder.

Fig. 4.11 Hardware implementation of the proposed ALADE algorithm.

The output of each XOR gate is fed to a conditional inverter, implemented with another XOR and controlled by the inversion signal *alt*. The latter is provided by a 1-bit counter (T flip-flop), whose value toggles every time a new word is transmitted/received. As for the ALADE decoder, the modifications are even simpler. An alternating signal, generated by another T flip-flop, is used to invert the selection of the multiplexers for every other word, as shown in Figure 4.11b.

This simple solution allows to achieve almost perfect DC balance, as shown in the rightmost bar of Figure 4.10: when *Lena* pixels are encoded with ALADE, logic-0s are 49.9% of the total transmitted values. This result is paid with a reduction in the achieved power savings. The most extreme case occurs when the input trace is perfectly constant: with ADE, after transmitting the first word, the rest of the encoded trace consists only of 0s. Thus, both the TC per word and the ratio between number of 1s and number of 0s tend to 0. With ALADE, instead, the encoded trace consists of the alternation between 00..0 and 11...1, hence the ratio between 1s and 0s tends to 1 (perfect DC balance), but the TC per word also remains at 1. In practice, however, the TC difference between ADE and ALADE is much less marked for real data traces. In the example of Figure 4.10, for instance, ALADE reduces the TC of 60.7% with respect to transmitting raw pixels, just a 0.5% less than the 61.2% reduction achieved by standard ADE.

4.4.7 Fine-grain Adaptation of Power/Quality Tradeoff

The main tunable parameter of ADE and its variants is the number of saturated bits l, which influences the maximum accepted error. As explained in Chapter 1, the notion of output quality is essentially application dependent; hence, the selection of l should be driven by application-level considerations. In particular, the problem of finding the optimal l is essentially an error-constrained power minimization. The designer must set the *maximum* l that allows to meet application-level constraints on quality, since it will produce the minimum power consumption on the bus.

For many application domains, such as radio and multimedia, quality (error) constraints are not specified in terms of maximum error, but rather through statistical aggregate metrics, such as the *MED* or the *MSE*, both introduced in Chapter 1. In this section, we show that ADE can be used also when targeting quality constraints expressed with these metrics. This is thanks to the fact that l can be easily reconfigured at runtime. For instance, in the hardware implementation of the encoder for ADE, the number of saturated LSBs can be stored in a flip-flop-based register, so that its value can be changed in a single clock cycle.

Given that l only determines an error upper bound, the ideal way to meet a statistically-specified constraint (e.g. in terms of *MED*) would be to keep track of the considered metric within the encoder, computing the *actual* error introduced by saturation on each transmitted word. This value would then be compared with the application-level constraint, in order to determine l for the next transmitted word. While this scheme would be feasible in software, it would introduce a significant overhead to a hardware implementation of the encoder.

The simpler yet effective solution described in this section consists in periodically alternating between different values of l according to a fixed *Duty Cycle* (DC), computed offline. This solution works under the reasonable assumption that LSBs of input words are uniformly distributed [181]. If this assumption holds, it is easy to derive that the saturation error is also a uniformly distributed random variable e, taking values in the range between 0 and the l-dependent upper bound $E_{MAX}[l] =$

$2^l - 1$. Consequently the MED and the MSE can be computed analytically as:

$$\begin{aligned} MED[l] &= \mathbb{E}(e) = \frac{E_{MAX}[l]}{2} \\ MSE[l] &= \mathbb{E}(e^2) = \frac{E_{MAX}[l](2E_{MAX}[l]+1)}{6} \end{aligned} \tag{4.9}$$

Both equations only depend on E_{MAX}, which is in turn function of the encoder parameter l. Given a constraint on MED or MSE, these equations can be inverted to extract the *ideal* value of the maximum error $\hat{E}_{MAX}$ that would generate the desired value of the aggregate metric.

Unfortunately, the achievable values of E_{MAX} in ADE are *quantized*, due to the fact that the number of saturated bits l cannot be fractional. This means that except for some particular cases, the desired maximum error $\hat{E}_{MAX}$ cannot be achieved with a fixed setting of l, which motivates the need for periodic duty cycling.

The ideal (fractional) l will be in general included between two integer values:

$$\begin{aligned} l_L &= \lfloor \log_2(\hat{E}_{MAX}+1) \rfloor \\ l_H &= \lceil \log_2(\hat{E}_{MAX}+1) \rceil \end{aligned} \tag{4.10}$$

Alternating between these two values with the proper duty cycle allows to almost exactly match the imposed constraint. For example, supposing that the metric of interest is MED, and that the target error constraint is $\hat{E}_{MED}$, the problem of matching the target constraint reduces to finding a duty cycle $DC < 1$ so that:

$$DC \cdot MED[l_H] + (1 - DC) \cdot MED[l_L] = \hat{E}_{MED} \tag{4.11}$$

Solving (4.11) for DC and using (4.9) yields:

$$DC = \frac{\hat{E}_{MED} - MED[l_L]}{MED[l_H] - MED[l_L]} = \frac{2\hat{E}_{MED} - E_{MAX}[l_L]}{E_{MAX}[l_H] - E_{MAX}[l_L]} \tag{4.12}$$

The denominator of this equation is always an integer, and represents the minimum period (in number of clock ticks) of the duty cycle. The numerator represents the number of clock ticks in each period for which the ADE encoder must use $l = l_H$, before switching to $l = l_L$ for the rest of the period. A similar relation can be computed for the MSE case.

Theoretically, using the duty cycle computed with (4.12) would yield exactly the desired statistical error, under the previously mentioned assumptions on the uniform distribution of LSBs. In practice, it is not always possible to exactly achieve the desired DC, and consequently the error target can only be matched approximately. This is due to two main reasons. First, the numerator in (4.12) might be not integer and second, the sequence of transmitted words might not be a multiple of the DC period. Both conditions cause a slight bias in the error. The latter, however can be made conservative (i.e. tending towards smaller *MED*/*MSE*) using two precautions: (i) the numerator of (4.12) should be truncated towards zero rather than rounded to the nearest integer, (ii) every new duty cycle should start using $l = l_L$, so that a lower error is produced whenever the cycle cannot be completed.

In this approach, the optimal duty cycle for a given error target can be computed offline. Consequently, the hardware cost of this solution is simply that of a counter and a comparator, used to increment and decrement l according to the numerator and denominator of (4.12). As mentioned above, switching l has practically zero cost in the hardware implementation of ADE, as it simply consists in changing the value of a register.

4.4.8 Compatibility with Standard Bus Protocols

Most standard serial bus protocols do not specify a particular semantic of data bits, which means that words can be encoded arbitrarily without violating the standard. However, the details of ADE integration depend on the selected protocol.

Serial Peripheral Interface (SPI) [123], *Inter-Integrated Circuit* (I2C) [124], *Integrated Interchip Sound* (I2S) [182] and others specify an interface composed of three main types of wires: (1) one or more data lines, (2) a separate clock line, (3) one or more control lines (e.g., slave select in SPI). For these protocols, ADE is clearly only applied to data lines, whereas clock, which provides synchronization and control lines, which have very low switching activities are not encoded. Moreover, since ADE does not alter the word length, it can be fully integrated with preexisting interfaces.

In SPI, the entire control of the transmission takes place through the *Slave Clock* (SCLK) and *Slave Select* (SS) lines. Thus, the *Master-Out-Slave-In* (MOSI) and *Master-In-Slave-Out* (MISO) lines are only used for data bits, and can be fully encoded with ADE. The same applies also to the I2S protocol, in which the *Serial Data* (SD) line only transmit audio samples.

In other cases, such as I2C, the data line is used, in different instants, to transmit both data and control information (e.g., slave addresses). In this scenario, ADE encoding and decoding must be applied selectively to data words, which are the only ones that can tolerate errors, skipping control words. However, although with a reduced impact on total power savings, ADE is fully applicable to these cases.

Other protocols, such as RS-232 [183] and *Controller Area Network* (CAN) [125] are asynchronous, i.e., timing information is conveyed through data lines and no separate clock lines are present. RS-232 transmits synchronization and control information through start/stop bits (and optional parity bits, etc.) on the data line. CAN is a much more complex standard, which embeds up to 64 bytes of data in *frames* that contain synchronization, priority, acknowledge and CRC information. All these elements must remain untouched in order for the transmission to complete correctly. Nevertheless, ADE can still be used on data bits.

The only case when ADE cannot be used straight-forwardly is when a standard specifies a precise data encoding. This is the case for some display protocols such as MIPI's DSI [184], which uses TMDS. In this case, although ADE can still be applied in cascade, it is likely to have limited effectiveness.

4.4.9 ADC/Encoder Co-Optimization

Sensor devices implementing ADE can obtain further power savings beyond those yielded by the encoding itself from the optimization of their embedded ADCs. This co-optimization is specifically possible for ADCs based on the *Successive Approximation Register* (SAR) principle [185].

SAR ADCs are relatively old, but they have recently regained popularity for embedded applications because of their low implementation costs. Indeed, this family of converters uses a single chain of *Sample-and-Hold* (S/H) and threshold comparator [185]. Despite this low complexity, the energy consumption of a SAR ADC can be comparable to that of a long off-chip bus. For instance, the commercial converter described in [186] consumes up to 17.3 pJ/bit when working at its maximum operating frequency of 48 MHz, which corresponds to 207.6 pJ for a 12-bit conversion.

A high-level block diagram of a SAR ADC is shown in the gray area of Figure 4.12. These ADCs work by comparing the input analog voltage sampled by the S/H ($V_{S/H}$) to increasingly accurate approximations (V_{APP}) generated by an internal *Digital to Analog Converter* (DAC), using a dichotomous search approach. Next, we will briefly recap the functionality of a SAR ADC in order to then discuss its integration with ADE. Without loss of generality, we will assume that the digital output produced by the converter is in unsigned binary format, and that the quantization of analog voltages is uniform.

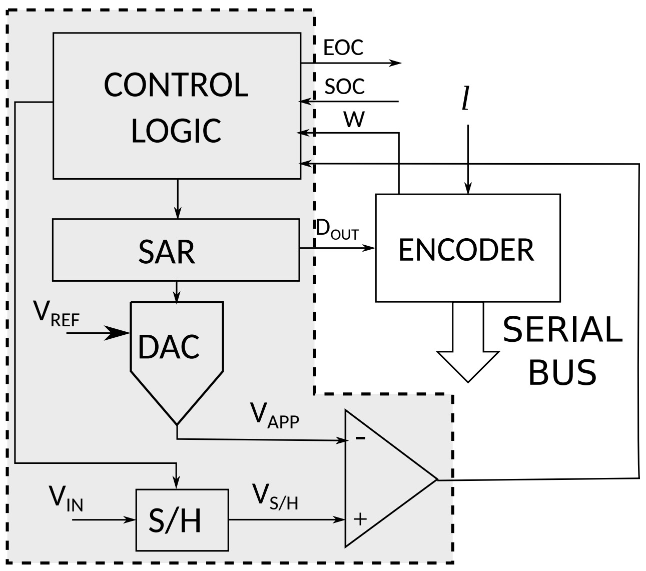


Fig. 4.12 Schematic of a SAR ADC and of its connection with the proposed ADE serial bus encoding.

A new conversion is triggered activating the *Start Of Conversion* (SOC) control signal. In the first conversion cycle, the successive approximation register is loaded with the digital value 10..0, which after D/A conversion, corresponds to $V_{APP} \approx \frac{V_{REF}}{2}$, where V_{REF} is the full-scale voltage. This value is then compared with the sampled analog signal $V_{S/H}$ by the threshold comparator. Depending on the outcome of this comparison the MSB of the SAR is kept at 1 or reset to 0. Specifically, these two operations are performed when $V_{APP} \geq V_{S/H}$ and when $V_{APP} < V_{S/H}$ respectively, since all values greater than V_{APP} have a 1 at the MSB in their output digital representation, and vice versa.

In the next conversion step, the second MSB of the SAR is set to 1, so that the register content is either 010..0 or 110..0 depending on the outcome of the previous cycle. Once converted to analog, these values correspond to either $\frac{V_{REF}}{4}$ or $\frac{3V_{REF}}{4}$.

This new approximation is compared again with the sampled analog input and the result is used to determine the final value of the second MSB of the SAR. This process is repeated a number of times equal to the desired output bit-width.

In summary, each cycle of a SAR conversion determines the value of exactly one bit of the digital output, starting from the MSB. At the end, the ADC activates the *End Of Conversion* (EOC) signal to notify that the process has completed.

The latency of a SAR ADC is linear in the number of output bits, while its power consumption is approximately constant throughout the conversion, since each cycle involves the same set of components (S/H in hold mode, SAR, DAC, comparator and control logic). Therefore, the *energy* consumed by the converter is in first approximation linearly dependent on the output bit-width.

This feature can be used in combination with the ADE family of encodings to simultaneously reduce both conversion and transmission energy, with the scheme shown in Figure 4.12. Several variants are possible, depending on the integration between encoder and ADC.

Let us denote with n the maximum precision of the ADC. At some point during the lifetime of the system a full-precision conversion output will be surely transmitted on the bus, therefore n should also be the length of the serial bus codewords. The contrary would mean that a too precise ADC has been selected, and a better choice would be to replace the component.

A first possible optimization consists of forcing a reduction in the A/D conversion bit-width using the saturation parameter l of ADE. In fact, it is an evident waste of energy to perform A/D conversion on the entire bit-width n if some LSBs will be ignored by the encoder because of saturation. As explained above, SAR converters process the bits of each sample starting from the MSB. Therefore, halting the conversion after $n-l$ clock cycles will induce an energy saving (and latency reduction) proportional to l. Many commercial SAR ADCs already support variable bit-width operation (input W in Figure 4.12). So, this basic form of early stopping does not require internal modifications of the converter [186]. It is enough to simply set the conversion precision parameter of the ADC to the value $W = n - l$.

Further savings are possible if the ADC and the serial bus encoder can be co-designed from scratch. In fact, the SAR can be easily modified to internally perform LSB saturation. This would allow to transform the encoder block of Figure 4.12

into a simple DE encoder (i.e. an array of XOR gates), eliminating the saturation multiplexers and reducing the complexity and energy consumption of the hardware.

Finally, an even tighter integration with SAR ADCs is desirable when using SADE. Indeed, this variant of our encoding performs saturation only when the difference between two consecutive words is larger than a threshold T_h. Therefore in the case of SADE, the ADC must always start with the maximum bit-width n, since at the beginning of a conversion it is not yet possible to know weather the transmitted word will be approximated or not. However, since SAR ADCs produce their output MSB-first, it is possible to determine if the difference between two samples is greater than T_h *while performing the conversion.* Then, if the ADC and the encoder are co-designed, the conversion can possibly be stopped after $l-n$ bits, or carried out completely with maximum accuracy.

To better explain this integration, we call $b_{n-l}[t]$ the sample currently being digitized by a SAR ADC after $n-l$ clock cycles. According to the functionality of these converters, this sample includes $n-l$ correct MSBs, while the remaining l LSBs are equal to 0. Let $\hat{b}_{n-l}[t]$ be the word with the same MSBs as $b_{n-l}[t]$ but with l 1s at the LSBs. The final SAR output at maximum precision is guaranteed to be in the range between these two words. Calling the last fully converted word $b[t-1]$, a conversion can be stopped early if either one of these two conditions verifies:

$$\begin{aligned} b[t-1] &\geq \hat{b}_{l-n}[t] + T_h \text{ or} \\ b[t-1] &\leq b_{l-n}[t] - T_h \end{aligned} \tag{4.13}$$

The implementation of these comparisons obviously requires additional hardware with respect to the SADE encoder shown in Figure 4.8. However, this scheme allows to effectively optimize the precision (and energy consumption) of a SAR ADC combined with SADE on a sample-by-sample basis.

4.5 Serial-T0 Encoding for "Bursty" Sensor Data

The second lossy bus encoding presented in this chapter is called *Serial-T0* (ST0). While ADE and its variant are effective for encoding data coming from a variety of error tolerant peripherals, ST0 is especially adequate for *image* data. Therefore, the ideal targets for ST0 encoding are the serial buses connecting image sensors (e.g.

CCD or CMOS cameras) and displays (e.g. LCDs or OLEDs). These connections are often relevant contributors to the energy consumption of a system, due to the high signaling frequencies involved, in the order of hundreds of MHz, and the large quantities of data transmitted. Typical standard protocols for these devices include SPI and I2C, as well as the more specific Camera Serial Interface (CSI-2).

Serial-T0 is inspired by the T0 technique for parallel address buses in microprocessors [187], from which it inherits the idea of leveraging the correlation among subsequently transmitted data to totally eliminate value transitions from the bus. However, this result is achieved in a significantly different way compared to the original T0, considering the serial nature of the target physical links and the error tolerance of the transmitted data.

In the rest of this section, we describe the encoding algorithm of Serial-T0, as well as the corresponding hardware implementations of encoding and decoding circuits. We then discuss the integration of ST0 with standard protocols used in commercial image peripherals and the required modifications to support DC balancing.

In the experimental results of Section 4.6 we show that, for image data, ST0 is superior to both lossless and lossy state-of-the-art encodings, including the ADE algorithm introduced in Section 4.4. We also evaluate the impact of a ST0-based serial transmission on the outputs of two machine learning applications involving images (OCR and face recognition), showing that significant power savings on the bus can be obtained at the cost of a negligible drop in output quality.

4.5.1 Parallel T0 Encoding for Memory Addresses

ST0 is inspired by a popular (lossless) encoding for parallel buses called T0, first proposed in [187]. T0 was designed to encode addresses sent by a processor towards the *instruction memory*. Its functionality can be expressed by the following equation:

$$(B[t], INC[t]) = \begin{cases} (B[t], 1) & \text{if } b[t] = b[t-1] + S \\ (b[t], 0) & \text{otherwise} \end{cases} \tag{4.14}$$

In this formula $b[t]$ and $B[t]$ are the input data word and the corresponding codeword transmitted at time t, $INC[t]$ is a redundant bit, and S is a constant power of 2 called *stride*, corresponding to the parallelism of the memory (e.g., $S = 4$ for 32-bit addresses).

In practice, when the memory addresses requested by the processor form an ascending sequence with stride S, T0 avoids transmitting the new address completely and instead sets the *INC* bit to 1, leaving the data lines of the parallel bus untouched. The actual requested address is computed automatically by the decoder (within the memory). The rationale of this algorithm is that the instruction memory is mostly accessed in sequences with fixed stride, occasionally interrupted by jumps, in correspondence to loops, function calls or interrupts in the processor.

The name T0 refers to the fact that, for an ideal infinite ascending sequence, this encoding reduces the number of transitions on the bus to 0.

4.5.2 Serial-T0

Serial-T0 inherits from T0 the idea of transmitting new data only when some condition is verified and letting the receiver autonomously “infer” the implicitly transmitted value in the remaining cases. However, the two encodings are extremely different in many aspects:

- First and foremost, T0 is designed for *parallel* buses, while ST0 deals with *serial* data connections.
- T0 is a *lossless* encoding whereas ST0 is *lossy* and trades off power reductions for data approximations.
- T0 exploits data locality in *addresses*, while ST0 leverages the bursty nature of image *data*.

The similarity between the two solutions lies in the analogy between the statistics of requested memory addresses and that of serialized image pixels. Specifically, the correlation among consecutive memory addresses in a sequence resembles the one between subsequently transmitted pixels. Likewise, occasional breaks in the sequentiality of addresses caused by jumps and loops are similar to *bursty* regions in image data traces (see Section 4.3).

Serial-T0 exploits the fact that most of the relevant information in a serialized stream of image pixels is contained in bursty phases. Accordingly, it reduces the number of value transitions on the bus to 0 during high correlation phases and transmits accurately during bursty phases. This is achieved selectively replacing the word to be transmitted with a special pattern that does not induce any transition on the bus (called 0-TC pattern hereinafter). Formally, the basic form of ST0 encoding implements the following equation:

$$B[t] = \begin{cases} \text{0-TC pattern} & \text{if } \|b[t] - b[t']\| \leq T_h \\ b[t] & \text{otherwise} \end{cases} \tag{4.15}$$

where T_h is a tunable maximum error threshold, and time t' represents the last time in which a data word was directly sent on the bus.

The similarity between (4.14) and (4.15) is clear. However, one important difference is the use of a special 0-TC pattern in ST0. This difference is due to the fact that, in a serial bus, repeating the last word does not eliminate transitions (in general) as in the parallel case.

Serial-T0 decoding is implemented as follows:

$$b[t] = \begin{cases} b[t'] & \text{if } B[t] = \text{0-TC pattern} \\ B[t] & \text{otherwise} \end{cases} \tag{4.16}$$

In practice, the codeword is simply taken as is, except when the 0-TC pattern is received. In that case, the decoder assumes as output the value of the last valid (non 0-TC) word received.

Data approximations in ST0 occur when the decoder output is different from the transmitted word, that is when:

$$0 < \|b[t] - b[t']\| \leq T_h \tag{4.17}$$

Increasing T_h produces more approximations, but allows to transmit more words as 0-TC patterns. Hence, tuning this parameter allows to explore the tradeoff between energy consumption and output quality.

Selection of 0-TC pattern and Implementation Details

Considering a data word of length n, there are only two n-bit patterns that do not produce value transitions when serialized: the all-zeros pattern (00..0) and the all-ones pattern (11..1). In order to completely eliminate all transitions during correlated regions, one of the two has to be used as 0-TC pattern in ST0. However, in most data formats, these binary patterns are already used to represent some other value. For example, in 24-bit RGB format, one of the most common representations of image pixels, 00..0 and 11..1 are interpreted as minimum/maximum hue of the corresponding primary color.

One way to solve this issue would be to add some form of redundancy to distinguish between the special 0-TC pattern and a real binary pattern with all-zeros/all-ones. Alternatively, one of the two patterns must be dedicated to be used *exclusively* as 0-TC pattern. Considering the critical impact of redundancy in serial buses, discussed in Section 2.6.2, our encoding uses the latter option.

This, however, creates a further issue: whenever a real datum equal to the 0-TC pattern is transmitted on the bus, the decoder erroneously interprets it as a repetition of the last valid word. Therefore, an additional data replacement is needed. A reasonable choice is to replace a real all-zeros/all-ones pattern with the binary word that is “most similar” in the chosen data representation, so to minimize the introduced error. In standard positional binary notations, as well as in 24-bit RGB, this corresponds to replacing 11..11 with 11..10, or 00..00 with 00..01. This substitution causes an additional error on the decoded data. For a standard positional representation, the relative magnitude of this error is in both cases equal to $1/FS$, where FS is the full-scale range of the data (e.g. 255 in 24-bit RGB).

With the goal of minimizing the occurrence of this data replacement, it is preferable to use as 0-TC pattern a binary word that occurs rarely in the input trace. In the case of image pixels, both 00..0 and 11..1 are uncommon and in general equally probable. However, in view of the possibility of using ST0 also for other types of error tolerant data, we choose 11..1 as the default 0-TC pattern.

Input	Total TC	Codeword	Total TC	Decoded	Error
00001011	3	00001011	3	00001011	0
00001111	5	11111111	3	00001011	4
00001100	8	11111111	3	00001011	1
00001101	11	11111111	3	00001011	2
00010111	15	00010111	7	00010011	0
00100011	19	00100011	11	00100011	0
00000100	22	00000100	14	00000100	0
00001101	25	00001101	17	00001101	0
00001110	28	11111111	17	00001101	1
00001011	31	11111111	17	00001101	2

Table 4.2 Example of operation of the proposed ST0 bus encoding for a trace of 8-bit unsigned data, under a maximum relative error constraint of $3.125\%(8/256)$.

Example of Operation

One example of the functionality of ST0, similar to the one described for ADE in Section 4.4.3 is reported in Table 4.2. The maximum error threshold of the ST0 encoder has been set to $T_h = 8$. This value corresponds to $\approx 3\%$ of the full-scale range of inputs and causes a total TC reduction of $\frac{31-17}{31} = 45.2\%$, considering both intra-word and inter-word transitions and assuming MSB-first serialization. This remarkable reduction is obtained at the expense of just $\frac{1}{255} = 0.4\%$ average error on the decoded data (in general, the average error is smaller or at worst equal to T_h).

For a better visualization of the effects of ST0 encoding, Figure 4.13 reports the trace of input data words and decoded codewords over time for the same example of Table 4.2, assuming a decimal interpretation of data. Notice how ST0 “flattens” the low variation phases of the data sequence, while preserving accurately the bursty phases. In this example, high and low correlation phases are approximately equal in length. However, the former are normally much longer in real image sensor traces. Thus, in real applications, Serial-T0 can obtain even larger TC reductions.

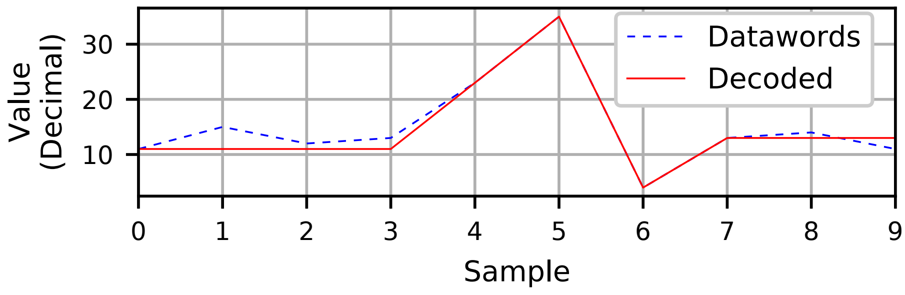


Fig. 4.13 Example of operation of the proposed Serial-T0 bus encoding for a trace of 8-bit unsigned data, under a maximum relative error constraint of $3.125\%(8/256)$.

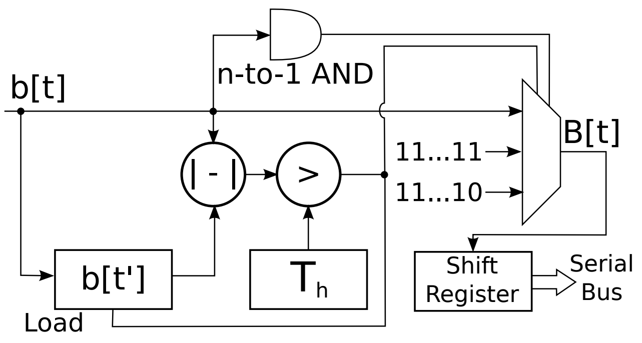
Comparison with Reduced Bit-width Transmission

As for ADE and any other lossy encoding, one first important assessment to be made on the usefulness of ST0 is whether our proposed encoding is superior to a simple reduced bit-width transmission.

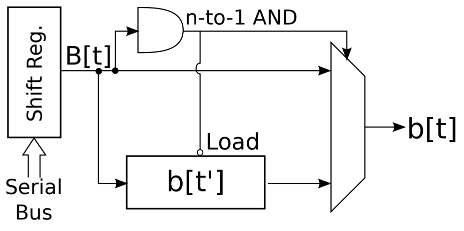
Similarly to the case of ADE, the amount of data approximation in ST0 can be easily set at runtime. In practice, quality reconfiguration can be implemented simply changing the value of T_h in (4.15). The codeword length is not affected by this reconfiguration, making it transparent from the point of view of the underlying transmission protocol Moreover, T_h can be tuned with very fine granularity (± 1 LSB), allowing precise control of the maximum allowed error. In contrast, ADE only allowed changing the maximum error in steps of powers of two.

Also in this case, the tuning of the acceptable error can be performed independently from the receiver, without the need for an initial handshake. In fact, the decoder functionality described in (4.16) does not depend on T_h.

All these features of our encoding provide the same advantages described in the case of ADE with respect to the integration with standard protocols. In particular, we will discuss in more detail the integration of ST0 with CSI-2, one of the most popular standard for image sensors, in Section 4.5.5. This standard only allows packing data in multiples of one byte, therefore variable bit-width transmission could only be performed at a very coarse granularity. In contrast, ST0 still allows very fine dynamic quality configuration when implemented for links that use legacy peripherals, without requiring any hardware modification.



(a) ST0 encoder.



(b) ST0 decoder.

Fig. 4.14 Hardware implementation of the proposed ST0 serial bus encoding.

4.5.3 Hardware Implementation

A further similarity between ADE and ST0 is the ease of implementation of the encoding and decoding procedures in hardware. Two block diagrams of possible implementations are shown in Figure 4.14. Both circuits are designed to use the all-ones pattern as 0-TC pattern.

The hardware of the encoder is composed of six main elements. A n-bit register (where n is the codeword length) stores the last word transmitted without approximations $b[t']$, and another is used to store T_h, so that the maximum error constraint can be changed at runtime. A subtractor and a comparator are used to determine whether the absolute value difference between $b[t']$ and the current word $b[t]$ is larger than T_h. Moreover, a n-to-1 AND gate is used to detect the occurrence of a real 11..11 pattern. Based on the outputs of the comparator and of the AND gate, a multiplexer forwards to the serializer either the input data word, the 0-TC pattern, or the replacement pattern 11...10. Notice that, as for ADE, storing T_h in a register allows quality reconfiguration in a single clock cycle.

The decoder circuitry is even simpler. Again, a n-bit register stores the last valid received word. Another n-to-1 AND gate detects the occurrence of a 0-TC pattern, and a multiplexer produces the final output selecting between the current codeword and the last valid word.

4.5.4 Serial-T0 for DC Balanced Connections

Coherently with our previous presentation of ADE, this section discusses possible modifications to the ST0 algorithm to support DC balancing.

Although the balancing of logic values is not an issue for most protocols considered in this work, it is still possible to devise a simple variant of ST0 to achieve it, at the cost of a small increase in the encoder and decoder complexity and of a slightly worse TC reduction.

The prevalence of 0s or 1s in ST0 codewords strongly depends on the selection of the 0-TC pattern, as discussed in Section 4.5.2. Using all-zeros as special 0-TC pattern will result in codewords with a large number of 0s, and vice versa. The most straight forward solution to avoid this strong prevalence is to alternate between the two possible 0-TC patterns for each new transmitted word. It is easy to see that doing so will not worsen the DC balance of unencoded input data. Conversely, it will tend to improve it. Moreover, considering that ST0 mainly targets image data, there are no strong drawbacks in choosing this solution, since both 00..00 and 11..11 are equally uncommon patterns in image pixels, as discussed in Section 4.5.2.

In order to implement DC balanced Serial-T0, a toggle flip-flop must be used in order to change the used 0-TC pattern in every clock cycle. When a “real” 00..00 or 11..11 pattern has to be transmitted, it will be approximated with 00..01 or 11..10, respectively. The corresponding modifications to the encoding and decoding circuits of Figure 4.14 are trivial and similar to those presented in Section 4.4.6.

Figure 4.15 shows an example of the effectiveness of this solution, analogous to the one described for ALADE. The three bars report the occurrence of logic-0s and 1s in the serialized bus trace generated by the *Lena* image. The leftmost bar refers to unencoded data words, while the one in the center has been obtained after ST0 encoding with $T_h = 8$. Our encoding yields a TC reduction of $\approx 63\%$ for these data, but also produces a strong prevalence of logic-1s, due to the use of the all-ones 0-TC pattern. Finally, the rightmost column shows the outcome of the DC-balanced

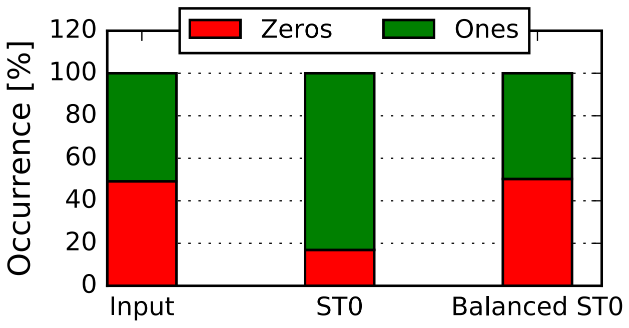


Fig. 4.15 Occurrence of 0s and 1s in the serial bus trace produced by the *Lena* Image before and after encoding with two Serial-T0 variants. Both variants use $T_h = 8$.

version of ST0. In this case, the TC reduction is less dramatic ($\approx 52\%$), but the balancing between logic values is almost perfect (49.9% of 0s).

In general, similarly to ALADE, DC balanced ST0 will always yield inferior TC reductions compared to the original algorithm. The worst case is that of a highly correlated bus trace, in which the difference between all consecutive words is included in the interval $[-T_h : T_h]$. In such scenario, standard ST0 would totally eliminate logic value transitions on data lanes: after correctly transmitting the first datum, all subsequent words would be replaced by the chosen 0-TC pattern. For the same trace, the DC balanced version of ST0 would instead generate one transition per word, due to the alternation among 11..11 and 00..00.

4.5.5 Compatibility with Standard Bus Protocols

The most widely adopted serial protocols for connecting image sensors (i.e. cameras) to processing elements are those developed by the MIPI alliance [184]. In particular, the *Camera Serial Interface* (CSI-2) is currently the de facto standard for camera interfacing, due to its good compromise between high speed and low implementation costs [188].

CSI-2 can be combined with multiple physical level protocols. The most common choice is to use the D-PHY standard [189], which is based on LVDS connection pairs. In particular, D-PHY specifies up to four physical LVDS pairs (called *lanes* in the standard) for transmitting data, while a separate lane is used for transmitting a *Double Data Rate* (DDR) source synchronous clock. In the latest versions of CSI-2 and D-PHY, the peak transmission rate of each lane is 2.5 Gbps.

CSI-2 uses D-PHY only for the transmission of pixels, which can be represented in multiple formats, including RGB, YUV, RAW and others. Control messages and status information on the peripheral, instead, are exchanged on a separate 2-wire physical interface that implements the I2C protocol [124].

A high-level schematic of an interconnection between a camera and a processing element using CSI-2 and D-PHY is shown in Figure 4.16.

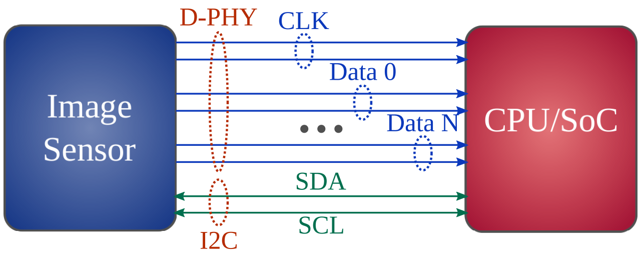


Fig. 4.16 Example of camera to processor connection using CSI-2 and D-PHY standards.

As discussed in the case of ADE, in order to integrate ST0 with CSI-2, care must be taken to only encode raw pixel data. In fact, to preserve full compliance with the standard and allow correct communication, control bytes (sync-sequences, packet headers and footers, etc.) cannot be approximated. To this end, pixel data must be encoded with ST0 *before* encapsulating them into a CSI-2 packet.

Differently from other more general purpose protocols, CSI-2 explicitly specifies the semantics of the packet payloads sent on D-PHY lanes, which must contain image pixels. However, although ST0 alters the value of the transmitted words, it does not alter their semantics (i.e. a pixel component remains a pixel component after being encoded with ST0). Therefore, ST0 can be applied on CSI-2 payloads without having to modify the standard in any way.

In terms of effectiveness, since LVDS is a level-based encoding, every transition eliminated by ST0 on data directly translates to power savings on the bus. Therefore, the final impact of our encoding is proportional to the amount of data bits over the total bits transmitted including control.

Special care must be taken when ST0 is used to transmit color images. In fact, different color components are in general not cross-correlated. Therefore, in principle, the encoding and decoding hardware of Figure 4.14 should be replicated for each color channel. In practice, however, an optimized implementation can reduce costs by sharing hardware (namely the mux. and the n-to-1 AND gate) if the physical data lanes are less than 3.

In terms of pure transition count reduction, the optimal interconnect configuration from the point of view of ST0 is one that uses a *separate physical wire* for each color component (e.g., three lanes in CSI-2). In fact, this allows to further reduce the TC, eliminating also some inter-word transitions when multiple 0-TC patterns are transmitted in sequence. In general, however, this advantage has to be balanced with the cost of the encoder, which thanks to sharing slightly reduces for a smaller number of lanes.

While this section has focused on CSI-2 due to its wide adoption for image sensors, similar considerations also apply to other protocols (SPI, I2C, CAN, etc.). Clearly, as discussed for ADE, the net impact of ST0 will be more significant in simpler protocols (such as SPI) with respect to more complex ones (like CAN), due to the larger amount of error intolerant control-related information exchanged in the latter.

4.6 Experimental Results

Setup

In a first experiment, we assess the effectiveness of the two proposed encodings on realistic serial traces of error tolerant data. In particular, we consider three groups of inputs.

A first set contains the traces generated by three common sensors found in mobile devices, namely accelerometer, magnetometer and gyroscope, obtained from the *Pervasive Systems Research Data sets* [178]. Data in these sets are collected when the user of a mobile device is performing a specific activity. In our experiment, we consider the *walking* and *biking* activities. The original decimal data have been converted to binary format, using the datasheets of commercial products to determine appropriate conversion parameters (bit-width, full scale ranges, etc.) [190–192]. All three datasheets refer to devices equipped with serial interfaces, using either I2C or SPI.

Input Set	Format	Bit-Width	Ref.
Accelerometer	Two's complement	12-bit	[190]
Magnetometer	Two's complement	16-bit	[191]
Gyroscope	Two's complement	16-bit	[192]
ECG	Unsigned	11-bit	[179]
Audio	Two's complement	16-bit	[180]
Images	RGB (Unsigned)	3·8-bit	[177]

Table 4.3 Input data sets used for comparative experiments on EQ scalable serial bus encodings.

A second group of data contains ECG samples taken from the *Physionet* online database [179], as representatives of biomedical applications. To simulate serial transmission, ECG samples have been converted to unsigned binary on 11-bit, according to the specifications of Physionet.

The third and last set refers to multimedia peripherals, namely voice recordings and images. Audio data have been obtained from the *Open Speech Repository* [180] and consist of 16-bit two's complement *Pulse Code Modulation* (PCM) samples. Images have been taken from the *Kodak Database* [177] and are represented in 24-bit RGB. Table 4.3 summarizes the inputs used in these first experiments and the corresponding data representations.

As shown in the following, ADE achieves superior savings for the same error constraint compared to ST0 for most of these inputs. A notable exception are image pixels, which are exactly the main target of the latter encoding. To further evaluate the effectiveness of ST0 on image data, we then evaluate its performance in the context of real error tolerant applications. Specifically, we test the effectiveness of ST0 for two machine-learning classification tasks dealing with images, i.e. *Optical Character Recognition* (OCR) and face detection/recognition. For these applications, we assess the impact of transmitting a camera image with ST0 on classification accuracy. The selected faces and text data sets are reported in Table 4.4. OCR is performed using the open-source software *Tesseract*, currently maintained by Google [193]. Face detection uses *Haar-like* features cascade [194], whereas face recognition is obtained through a *Local Binary Pattern Histogram* (LBPH) recognizer [195]. Both

Application	**Data Set**	**Content**	**Size [pixels]**	**Ref.**
Face Recognition	Faces94	Faces	180x200	[197]
OCR	ICDAR 2003	Text	Variable	[198]

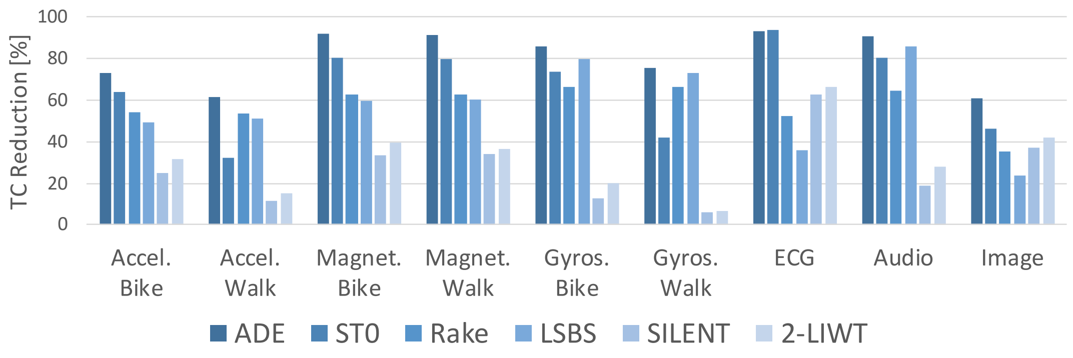
Table 4.4 Input data sets used for application-specific experiments on image data for the proposed Serial-T0 encoding.

algorithms are already included in the OpenCV [196] library and have been used with default parameters.

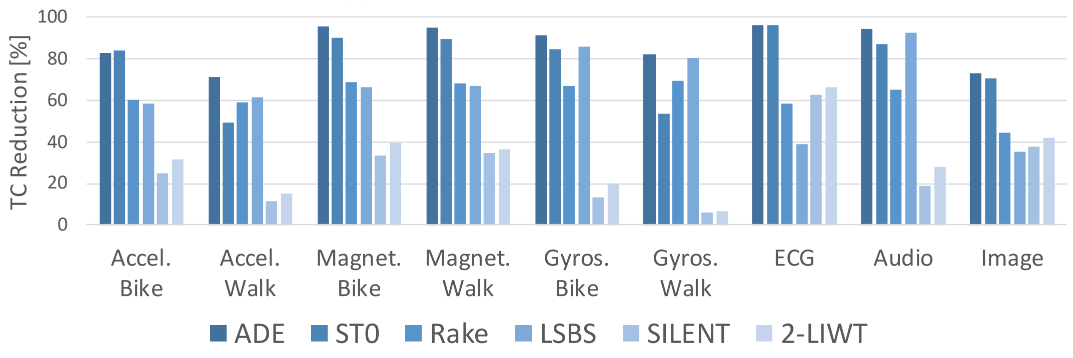
In our experiments, we compare ADE and ST0 with two state-of-the-art lossy serial bus encodings, namely LSBS [131] and Rake [132]. While Rake is a general-purpose encoding, LSBS is specifically designed for image data. Moreover, we also present results for two representative lossless approaches. With the same rationale, we consider SILENT [126], which internally makes use of accurate DE and is a general-purpose solution for correlated data, and the image-specific k-LIWT [130]. In particular, we select the best variant of this algorithm in terms of performance versus overhead tradeoff according to the authors of [130], i.e. 2-LIWT. In order for our comparisons to be fair, all selected works from the literature mentioned above *do not use redundancy*. This is in accordance with the analysis of the large overheads of redundancy in serial buses in terms of cost or throughput, presented in Section 2.6.2. To evaluate TC reductions, both our encodings and the comparisons from the literature have been tested within a simulation framework implemented in Python 3.5.

Without loss of generality, we assume that the underlying protocol on top of which our encodings are implemented does not transmit control bits on the same physical line as data. This is similar to the assumption done in the encodings we compare against, in order to simplify the computation of value transitions. Real protocols such as SPI and I2S follow exactly this architecture. As discussed in Sections 4.4.8 and 4.5.5, for more complex standards (e.g., CSI-2) the TC reduction is approximately equal to the one reported in these experiments, but weighted by the ratio between data and control bits.

To accurately evaluate the overheads associated with the implementation of our two proposed encodings, we have also synthesized the encoding and decoding hardware of Figures 4.7 and 4.14. The circuits have been specified at RTL using



(a) Maximum Relative Error = 2%



(b) Maximum Relative Error = 4%

Fig. 4.17 Transition count reduction of state-of-the-art serial bus encodings for different input data sets and for two different maximum relative error constraints.

VHDL, and synthesized using Synopsys Design Compiler K-2015.06 on a 45nm standard cell library from ST Microelectronics. We have verified the correctness of the post synthesis netlists by means of a timing simulation in Mentor QuestaSim v10.4, with random input stimuli. During these simulations we have also annotated the internal switching activity of the encoder and decoder netlists, which we have then loaded in Synopsys PrimeTime J-2014.12 to accurately estimate power.

4.6.1 Transition Count Reduction for Error Tolerant Data

TC Reduction Versus Maximum Error

Figure 4.17 reports the TC reduction (expressed in percentage) achieved by all considered encodings for the input data sets of Table 4.3. The two charts refer to two different constraints in terms of allowed error. Specifically, we impose an upper bound on the *maximum relative error* E_R, as a percentage of the full-scale range of variation of each data set. The parameters of the different encodings are set so as not to violate these constraints. In particular, the l parameter in ADE is computed as:

$$l = \left\lfloor \log_2\left(\frac{E_R}{100} \cdot FS + 1\right) \right\rfloor \tag{4.18}$$

while the threshold T_h in ST0 is set to:

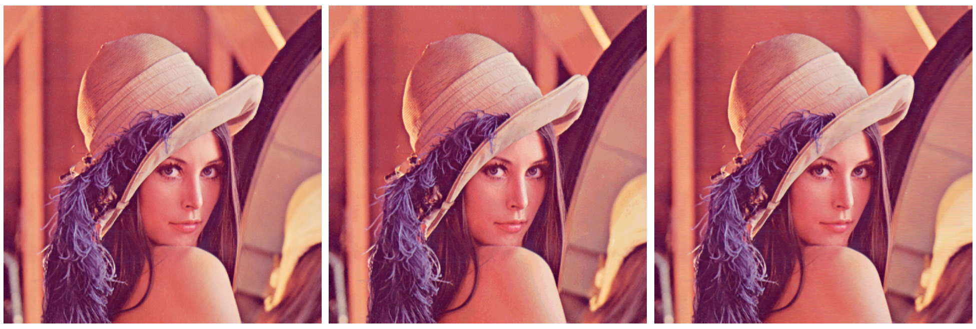
$$T_h = \lfloor E_R \cdot FS/100 \rfloor \tag{4.19}$$

In both cases, the floor operator ensures that the constraint is never violated. The parameters of competitor lossy encodings have been in a similar way, to have a fair comparison. SILENT and 2-LIWT, being lossless, do not accept a quality/error parameter. Consequently they have the same TC reduction in both graphs.

As shown by the plots, ADE obtains the largest TC reductions of all considered encodings for 7/9 or 8/9 input sets, depending of the maximum allowed error. ADE is slightly outperformed by ST0 for ECG data and Accelerometer data relative to the biking activity, because of the strong *bursty* nature of these traces. ST0 also achieves the second best reduction for 3 to 5 inputs, depending on the allowed error.

Both ADE and ST0 *always* obtain better TC reductions compared to the two considered lossless encodings, i.e. SILENT and 2-LIWT. This is an important result, that justifies the data approximations performed by our algorithms. In contrast, both Rake and LSBS are sometimes outperformed by one or more lossless encoding.

Figure 4.18 gives a sense of the impact of the errors committed by ADE and ST0 for one of the previous inputs. The leftmost picture is the original 24-bit RGB *Lena* from [177]. The other two are the images received by an ADE decoder and a ST0 decoder respectively, after being encoded with the corresponding parameters set to obtain a maximum E_R of 4%. In accordance to the graph of Figure 4.17, using this configuration yields a TC reduction of $\approx 72\%$ for ADE and $\approx 70\%$ for ST0. Nonetheless, there is hardly any noticeable difference between the two received images and the original input. In quantitative terms, the average normalized error distance (*NED*) on pixel colors is only $\approx 1.3\%$ for ADE and $\approx 1.1\%$ for ST0. A better metric for evaluating images differences is the *MSSIM* [31], which has been demonstrated to correlate well with the way humans perceive similarity. The ADE and ST0 outputs in the above example achieve *MSSIM* values of 0.9930 and 0.9933 respectively, where 1 would correspond to perfectly identical images.



(a) Input image. (b) ADE output. (c) ST0 output.

Fig. 4.18 Example of the effect of data approximations in the transmission of an image using the proposed ADE and ST0 encodings, with a maximum relative error constraint of $E_R = 4\%$.

Notice that ST0 tends to produce slightly better output quality for the same error constraint for image data; this trend will be confirmed in the next section. In summary, this experiment shows that a constraint such as $E_R = 4\%$ may be reasonable for a realistic application and that the relevant TC reductions obtained by our two proposed encodings do not translate in relevant quality degradations.

TC Reduction versus Actual Mean Error

In this second experiment, we consider the tradeoff between the TC reduction achieved by each encoding and the actual *NED* on decoded data. As a matter of fact, the parameters of all the lossy encodings considered in this section only constrain the *maximum* accepted error. The actual deviation introduced on each datum is in general smaller than this bound; consequently, also the final *NED* is in general smaller than the imposed constraint.

Figure 4.19 reports the tradeoff between *NED* and TC reduction for all encodings and for the same inputs of Figure 4.17. The points on the Pareto-curves relative to ADE and have been obtained varying the l parameter from $l = 1$ to $l = 5$. For each of these values, in order to have a fair comparison, the ST0 threshold has been set to $T_h = 2^l - 1$, i.e. exactly the maximum error induced by the saturation of l bits. As for the previous experiment, the parameters of competitor encodings have been set in a similar way to ensure fairness. The two horizontal lines in each graph represent

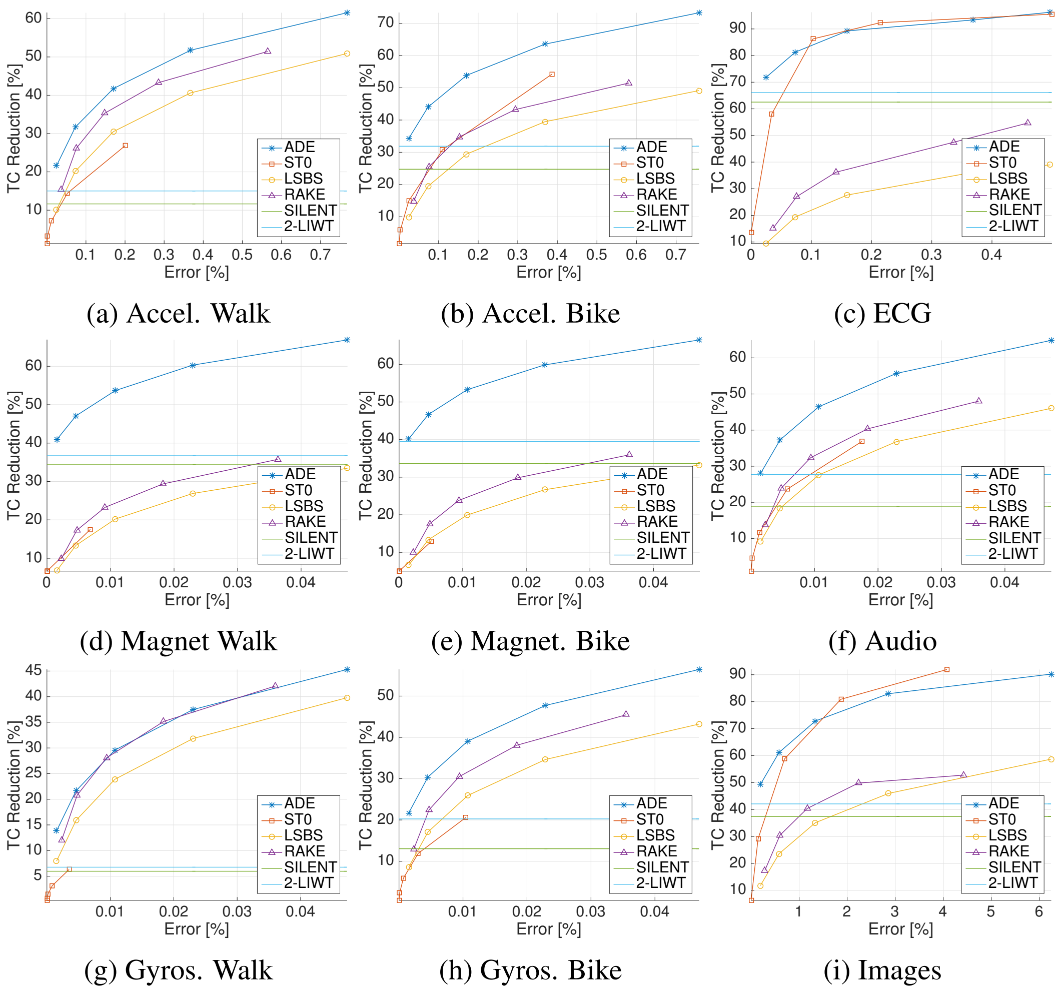


Fig. 4.19 Tradeoff between transition count reduction and *NED* in state-of-the-art serial bus encodings for different input data sets.

the TC reductions of SILENT and 2-LIWT: since these two encodings are lossless, their TC reduction is constant with respect to the error axis.

The curves show that, even when considering the actual error introduced by lossy encodings, ADE outperforms all alternatives for the majority of the inputs. In fact, the top-left corner of each graph represents optimality, i.e., an ideal encoding that reduces transitions to zero while not introducing errors. In most cases ADE dominates other encodings in the Pareto sense, by reducing the transitions more for the same output error.

Another conclusion drawn from this experiment is that ST0 has a much more limited range of applicability with respect to ADE. In fact, while the former is the best performing algorithm for two inputs (ECG and images) and for large-enough allowed errors, it achieves the worst tradeoff for many other data sets. This is due to the marked bursty nature of ECG and pixel data, which favors the functionality of ST0. The remaining inputs are clearly still temporally correlated (otherwise SILENT would not achieve positive savings); nonetheless, they tend to have a less marked alternation between idle and bursty phases, making ADE much more effective than ST0.

Notice that the relations between the TC reductions of different encodings in Figure 4.19 are generally different from those of Figure 4.17. On the one hand, this is because the considered quality metric is different (NED versus maximum error). On the other hand, differences are also due to the error ranges considered. In fact, by varying l in a fixed interval $[1:5]$, we are considering different ranges of NED, depending on the bit-width of the data (see the x-axes of each graph).

4.6.2 Transition Count Reduction for Random Data

ADE and ST0 are built specifically to deal with error resilient data from real devices. Nonetheless, analyzing their effectiveness for random data allows to draw interesting conclusions. Figure 4.20 shows the TC reduction obtained by all encodings considered in previous experiments when applied to 12-bit unsigned inputs generated randomly. Each point in the graph refers to a different set of inputs. All sets are drawn from a Gaussian distribution with the same mean μ, equal to half of the full scale range (2^{11}), whereas the standard deviation σ is varied in increasing powers of 2, from 2^0 to 2^{12}. Each input set consists of 10,000 words. In this case, the maximum allowed error for lossy algorithms has been set to 1% of the full scale range.

Figure 4.20 shows that ST0 achieves the best TC reduction for small σ. This is expected, since a small standard deviation enforces a strong correlation among subsequently transmitted words. For values of σ larger than 2^4 and for a quite large range of standard deviations, ADE is once again the best performing algorithm. The effectiveness of ST0, in contrast, decreases rapidly for more widely distributed data, which confirms the conclusion drawn is the previous section that this encoding is best used *only* for very strongly correlated data.

At the rightmost extreme of the graph, i.e., for highly-uncorrelated data traces, ADE yields approximately the same TC reduction as LSBS. This is because dif-

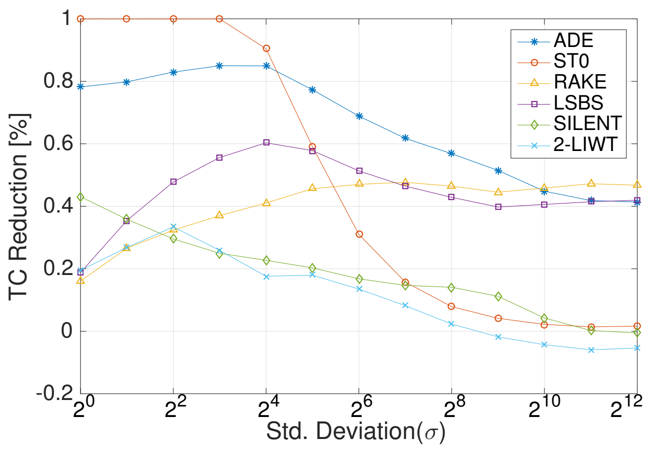


Fig. 4.20 Transition count reduction of state-of-the-art serial bus encodings for random Gaussian data traces with different standard deviations (σ) and same mean (μ). Maximum relative error for lossy encodings $E_R = 1\%$.

ferential encoding has no effect for this kind of data and actually incurs a slight *increase* in the number of transitions. Such behavior is expected, since a Gaussian distribution with large standard deviation tends to approach a uniform distribution, which is the worst-case condition for differential encoding. In fact, it can be proven that, for uniform data, the HD between consecutive words has a binomial distribution centered in $n/2$, with n being the word length. Thanks to LSB saturation, ADE is still able to reduce the TC of about 40% in this case.

4.6.3 Transition Count Reduction for ADE Variants

This section is dedicated to the assessment of the effectiveness of SADE and ALADE. Figure 4.21 reports the TC reduction obtained for ADE and its variants, as well as for accurate DE, for some of the inputs listed in Table 4.3. In this experiment, the l parameter of ADE and its variants has been set to $l = 4$, which corresponds to a maximum allowed error of 15. The bar chart reports three variants of SADE, differing in the values of the threshold: $T_h = 16$ (SADE-16), $T_h = 32$ (SADE-32) and $T_h = 64$ (SADE-64) [2]. As explained in Section 4.4.5, using these thresholds with $l = 4$ corresponds to accepting a maximum error equal to approximately 100%, 50% and 25% of the signal variation.

[2]Although we use the same symbol to refer to both of them, the reader should not confuse the SADE threshold with the analogous parameter of ST0.

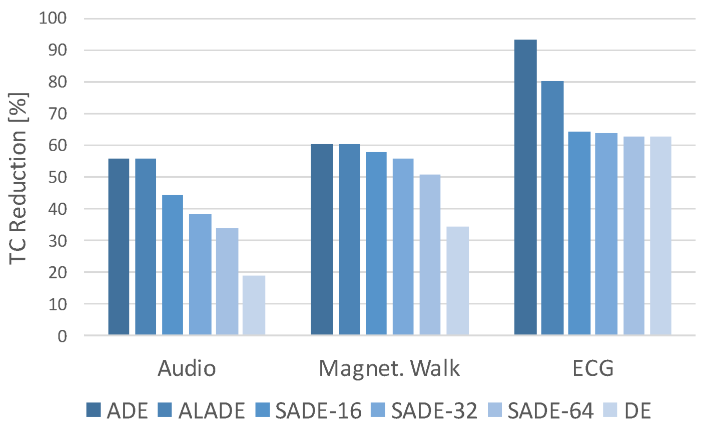


Fig. 4.21 Transition count reduction of the proposed ADE serial bus encoding and its variants for a selection of input sets. Saturation parameter set to $l = 4$.

A first observation is that, for audio and magnetometer inputs, ALADE achieves practically the same TC reduction as ADE (55.7% versus 55.6% and 60.3% versus 60.0% respectively). This indicates a small impact of inter-word transitions, which are the only source of difference in TC between the balanced and unbalanced versions of our proposed encoding. For ECG data the difference in TC is more marked, due to the fact that these inputs have long almost constant (highly correlated) sequences of words in the intervals between two heart beats. The latter resemble the worst-case condition for ALADE, discussed in Section 4.4.8. Similar results can be obtained also for the DC balanced version of Serial-T0.

As expected, all SADE versions achieve smaller TC reductions compared to basic ADE. Indeed, SADE skips some approximations in order to preserve small variations in the signal. A larger T_h generally causes smaller toggle reductions, yet always larger than those of accurate DE. The audio and magnetometer traces are composed by the alternation of small variation and large variation phases, while in ECG data, small variations are predominant. Thus, as for ALADE, SADE variants obtain better results for the first two data sets.

A better way to evaluate the effectiveness of SADE is to use an error metric that takes signal variation into account. To this end, we consider a metric that normalizes the error on each word with respect to the local variation of the data. In principle, this could be achieved simply dividing the error by the absolute value difference between two consecutive words. However, when two consecutive words are equal, such metric would produce a division by zero. Therefore, we use a *windowed error* metric E_w, in which the error on each word is normalized to the maximum variation of the signal in a sliding window of size $w \geq 1$, where w is computed as the minimum that avoids divisions by zero. Normalized errors on different words are then averaged to obtain the final aggregate metric. Mathematically, E_w is computed as:

$$E_w = \frac{1}{N} \sum_{t=1}^{N-w} \frac{\|b_d[t] - b[t]\|}{\max\limits_{t \leq s \leq t+w}(b[s]) - \min\limits_{t \leq s \leq t+w}(b[s])} \tag{4.20}$$

where $b[t]$ and $b_d[t]$ are the input and decoded words at time t respectively, and N is the length of the input trace. As desired, this metric gives more “weight” to errors happening in correspondence of small signal variations.

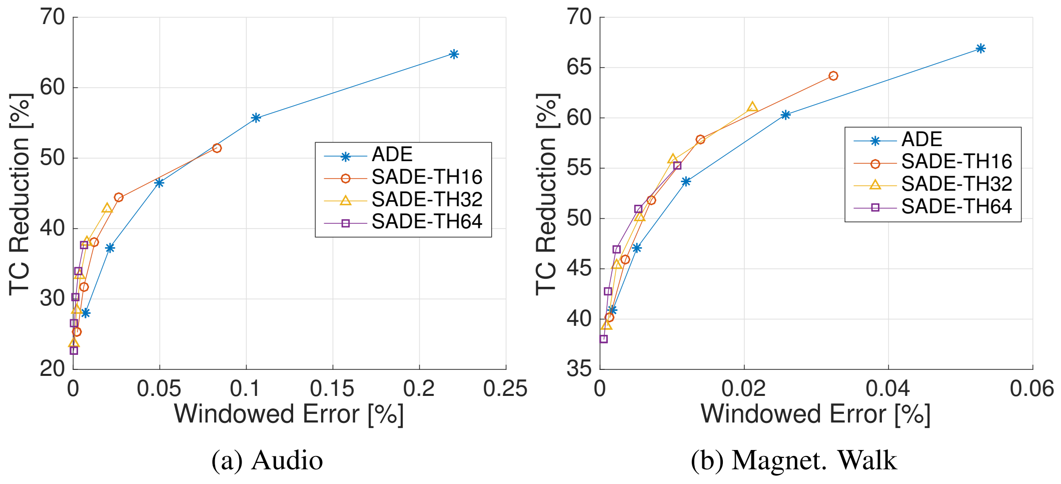


Fig. 4.22 Transition count reduction versus “windowed error” tradeoff obtained by the proposed ADE serial bus encoding and its variants for two input data sets.

Figure 4.22 shows the error versus TC reduction tradeoff for ADE and SADE considering E_w, for audio and magnetometer inputs. Different points in the graph have been obtained varying the general parameter l of all encodings from $l = 1$ to $l = 5$. The window length for error computation is $w = 4$ for audio data, and $w = 5$ for magnetometer. When using E_w SADE outperforms ADE for both inputs. Notice that SADE variants with a larger T_h always produce very small windowed errors, regardless of the value of l, because they have less opportunities for approximations. When very high quality is required, SADE-64 configures as the best solution for both input sets, while if a slightly lower quality is acceptable, an intermediate solution such as SADE-16 might be preferable. In general, the choice of the appropriate T_h must be evaluated depending on the specific inputs considered.

Finally, we present a motivating example of the use of duty cycling in ADE to fine-tune the error and TC reduction. To this end, we use a realistic application scenario. As mentioned in Chapter 1 image quality is sometimes measured by the *PSNR* metric. Acceptable *PSNR* values, e.g., in the context of lossy compression algorithms, are in the range of 20 to 40 dB [199]. Consider an application that uses ADE to transmit image pixels, e.g., digitized by a camera. Assume that the *PSNR* of decoded data with respect to the input is required to be $\geq 38\ dB$. For instance, this could be imposed in order to make the effect of our lossy encoding negligible with respect to a following compression phase, performed by the receiving processor. Using a typical 8-bit representation for each color component of a pixel, the *PSNR* constraint can be converted to $MSE \leq 10.306$.

Table 4.5 shows the result of a simulated ADE transmission with $l = 2$ and $l = 3$ of the usual *Lena* image. With $l = 2$, the *MSE* is significantly smaller than the requirement, meaning that ADE is not fully exploiting the available error tolerance, and is loosing opportunities to reduce the TC. On the contrary, with $l = 3$, the requirement in terms of *MSE*/*PSNR* is violated. Since l must be integer, no intermediate values are possible. Consequently, in this example, *no single value* of the parameter allows to closely match the *PSNR* constraint.

l	MSE	PSNR [dB]	TC Reduction [%]
2	3.48	42.71	61
3	17.01	35.82	73
Duty Cycling	9.929	38.162	66

Table 4.5 Example of fine-grain quality tuning in the proposed ADE serial bus encoding by means of duty cycling. The example refers to image data.

This limitation can be solved resorting to duty cycling, as discussed in Section 4.4.7. Using equations similar to (4.11-4.12), but for the *MSE* metric, we can obtain the minimum duty cycle length and the corresponding parameter assignment to closely match the desired *PSNR*. In this example, the results of such analysis yield a period of 84 cycles, in which $l = 2$ is used for the first 44 and $l = 3$ for the remaining 40. The result of encoding *Lena* with this scheme is reported in the last line of Table 4.5. As expected, this solution matches the desired quality constraint almost perfectly. By doing so, it allows to improve the TC reduction of an extra 5% with respect to the fixed solution with $l = 2$. Finally, notice that the *MSE* of the duty cycling solution is slightly *smaller* than the requirement, due to the conservative management of errors described in Section 4.4.7.

4.6.4 ST0-based Transmission Within Real Applications

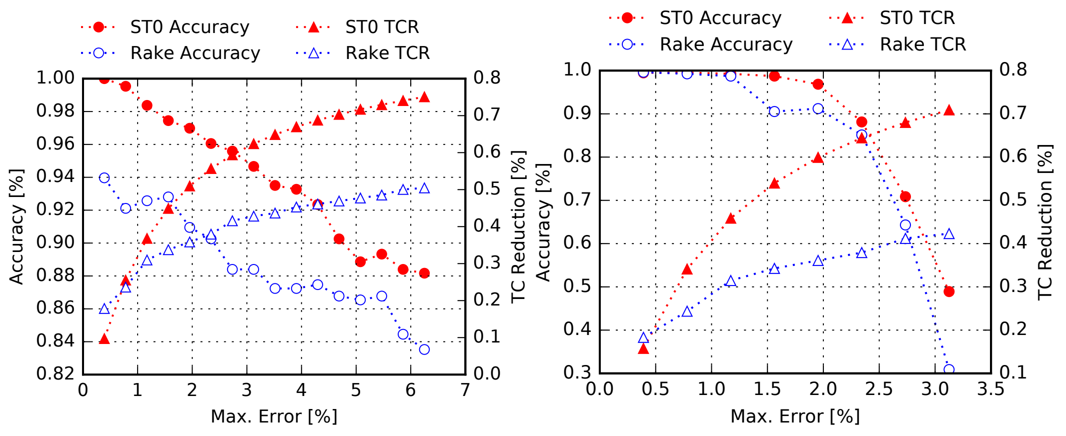
The experiment of Figure 4.19 has demonstrated that, for a relaxed-enough error constraint, ST0 achieves the best tradeoff in terms of TC reduction versus *NED* for image data. Therefore, in this section, we analyze the effectiveness of this encoding for transmitting images within the context of the two realistic applications introduced in Table 4.4. We have used the Rake encoding as state-of-the-art comparison for this experiment.

We have initially pruned the data sets of both applications, in order to only include those images for which classification is correct without approximations. Then, for all images in the pruned data sets, we have evaluated the results of applying the same algorithm to the output of a ST0-based or Rake-based serial transmission rather than to the original image. We have repeated this experiment for different error constraints; both Rake and ST0 accept such constraints in the form of a maximum allowed error (with a resolution of one LSB), which makes the comparison fair.

We have used application-level metrics for evaluating the quality of outputs for both tasks. In particular, for face recognition we have considered the *Hit Ratio* (HR), i.e. the ratio of correctly classified images over the total size of the pruned data set, while for OCR we have adopted the *Levenshtein distance* [200].

The results of this experiment are reported in Figure 4.23. In these graphs, the horizontal axes report the maximum allowed relative error (E_R), corresponding to the constraint imposed on the two encodings. The vertical axis shows the corresponding TC reduction and the output accuracy according to the selected metric, on a scale from 0 to 1.

These results show that, for the same error constraint, ST0 not only achieves larger TC reductions, but it also allows to obtain a better classification accuracy than Rake. This is motivated by the fact that ST0 only makes approximations in regions of the bus data trace with strong temporal correlation, which in turn correspond to regions of slowly varying color in the image (e.g. the background around the subject's face for the case of face recognition). These areas do not provide relevant



(a) Optical Character Recognition (OCR) (b) Face Recognition

Fig. 4.23 Transition count reduction and application-level output quality as functions of the maximum error parameter of the proposed ST0 serial bus encoding and of one competitor encoding, for two realistic classification applications.

information from the point of view of the classification task, which therefore is able to still output the correct result.

Assuming as a goal that of maintaining the accuracy of classification above 95% of the golden reference, ST0 allows to reduce the transitions on the bus of about 60% for both applications. For face recognition, Rake only generates a TC reduction of $\approx$ 30% for the same target quality. Furthermore, the Rake accuracy for the OCR application drops below 95% even when the maximum error is set to 1 LSB (leftmost point on the graph).

4.6.5 Hardware Overheads and Break-even Length Analysis

In previous experiments, we have compared different serial bus encodings in terms of pure TC reduction. A more accurate assessment of the real benefits of such encodings must take into account the actual energy consumption on the bus, including the overheads due to encoding and decoding circuitry.

To this end, we have synthesized the circuits of Figure 4.7 and 4.14 with the setup described at the beginning of the experimental results section of this chapter. We have implemented both 8-bit and 16-bit versions of the CODECs for ADE and ST0, targeting a $200MHz$ clock frequency in both cases. Table 4.6 reports the results of this synthesis. Moreover, the table also provides a simplified power versus frequency

Bit-width	**Circuit**	**Area** $[\mu m^2]$	**Power @ 200MHz** $[W]$	**Power vs Freq.** $[W]$
8-bit	**ADE Encoder**	85.02	$2.33 \cdot 10^{-5}$	$1.14 \cdot 10^{-13} f + 5.94 \cdot 10^{-7}$
	ADE Decoder	68.80	$3.32 \cdot 10^{-5}$	$1.64 \cdot 10^{-13} f + 5.24 \cdot 10^{-7}$
	ST0 Encoder	304.82	$4.59 \cdot 10^{-5}$	$2.20 \cdot 10^{-13} f + 1.95 \cdot 10^{-6}$
	ST0 Decoder	73.38	$1.66 \cdot 10^{-5}$	$8.08 \cdot 10^{-14} f + 4.84 \cdot 10^{-7}$
16-bit	**ADE Encoder**	173.57	$4.83 \cdot 10^{-5}$	$2.35 \cdot 10^{-13} f + 1.21 \cdot 10^{-6}$
	ADE Decoder	136.53	$6.70 \cdot 10^{-5}$	$3.30 \cdot 10^{-13} f + 1.05 \cdot 10^{-6}$
	ST0 Encoder	720.06	$1.11 \cdot 10^{-4}$	$5.22 \cdot 10^{-13} f + 6.29 \cdot 10^{-6}$
	ST0 Decoder	146.41	$3.28 \cdot 10^{-5}$	$1.59 \cdot 10^{-13} f + 9.01 \cdot 10^{-7}$

Table 4.6 Synthesis results for 8-bit and 16-bit encoders and decoders for the proposed ADE and ST0 serial bus encoding algorithms, implemented on 45nm CMOS.

model, where we assume that leakage power is constant, and dynamic power linearly depends on frequency.

Although both algorithms result in extremely low hardware overheads, the total power consumption of ADE is approximately 80% of that of Serial-T0, for the 16-bit implementation at 200MHz, and 90% for the 8-bit version, denoting also a better scalability. This is expected, given the respective complexity of the block diagrams of Figure 4.7 and Figure 4.14. In particular, the presence of a subtractor, an absolute value computation block and a comparator accounts for most of the area and power in the ST0 encoder, which is the block responsible for most of the additional overhead with respect to ADE. In the remaining three circuits, area occupation and power consumption are dominated by the contribution of the flip-flop registers that store the latest transmitted/received word (in ADE) or the latest "valid" word (in ST0).

Having the hardware overheads due to encoding and decoding available allows the evaluation of the total energy consumption of a serial transmission that uses ADE or ST0. To do so, we have modeled the bus data line as a microstrip on a FR-4 PCB [201]. We have assumed a line with $25\mu m$ of width and $10\mu m$ of thickness, over a substrate of $200\mu m$ height with permittivity $\varepsilon_r = 4.5$. These parameters yield a capacitance per unit-length equal to $C_{LEN} = 42.3\ pF/m$. This value can be used with the model of (4.1) to evaluate the power and energy per unit length consumed by the line.

The remaining parameters of (4.1), depend on the application considered and are summarized in Table 4.7. In particular, the switching activity (α) and the bit-width of input words (n) considered in this experiment refer to the transmission of the *Audio* and *Images* inputs from Table 4.3. We have selected two reasonable transmission frequencies for these two multimedia domains and we have assumed that encoder, decoder and serial line all use the same clock signal. For both inputs, we have compared the energy cost for transmitting unencoded data, with the one required for transmitting ADE and ST0 codewords. The maximum error constraint for our two proposed encodings has been set to 10 units for both data sets.

The results of this comparison are reported in Table 4.7. To obtain them, we have first computed the energy required by ADE and ST0 CODECs for encoding and decoding one word of data (E_{hw}). To this end, we have used the power consumption model from Table 4.6. Next, we have computed the average energy per unit-length consumed by the microstrip to transmit one word (E_l). The latter depends on the encoding used, as both ADE and ST0 reduce the number of transitions on the line, and consequently affect α.

For combining these two energy values, we have considered that the hardware implementations of our two CODECs are able to encode and decode one word in a single clock cycle. In contrast, the transmission of a word on the microstrip takes n clock cycles. Consequently, we have assumed that, when not used, the CODEC hardware blocks are power managed, and we have approximated their power consumption with 0 during this time.

The last column of Table 4.7 reports the *break-even length* l_{BE} of the bus required to amortize the cost of the CODEC. An encoding saves energy only if the bus exceeds l_{BE}, because the savings on the line due to the reduced TC become larger than the overheads due to the CODEC.

For both encodings, the break-even length obtained for image data is in the order of millimeters. In contrast, ADE outperforms ST0 for audio data, still obtaining

Data	**n**	**f [MHz]**	**V_{swing} [V]**	**Encoding**	α	**E_{hw} [pJ]**	**E_l [pJ/m]**	**l_{BE} [mm]**
Audio	16	4.0	1.8	**None**	0.34	-	744	-
				ADE	0.18	1.13	398	3.26
				ST0	0.28	1.46	620	11.8
Images	8	184.3	1.6	**None**	0.52	-	454	-
				ADE	0.14	0.57	123	1.74
				ST0	0.15	0.61	133	1.89

Table 4.7 Analysis of energy consumption and break-even line length for the proposed ADE and ST0 serial bus encodings, when they are used to transmit audio and image data.

a value of l_{BE} of few millimeters, whereas ST0 requires at least 1 cm of line to be effective. The superior results of ADE are produced by two combined effects: on the one hand, this encoding achieves better TC reductions, especially for audio data. On the other hand, it also incurs a smaller encoding and decoding overhead. However, notice that these results do not consider the actual impact of the encoding on the transmission quality. For example, although both encodings are constrained to commit an error smaller than 10 LSBs, the average error after decoding committed by ST0 is 14% smaller in the case of image data, and 74% smaller for audio data, compared to ADE. Therefore, a complete evaluation should account for the effect of each encoding on the quality of outputs, considered from the point of view of the target application. Nonetheless, all the break-even lengths reported in Table 4.7 are very short compared to the typical size of a PCB line, hence confirming the effectiveness of our proposed encodings.

4.7 Final Remarks

In this chapter, we have described two different lossy encodings for serial buses. Both these techniques leverage the inherent error tolerance of data that are commonly transmitted serially, such as samples from sensors, audio and images. By introducing controlled errors in the transmitted data, they trade off energy consumption for transmission quality.

Differently from the case of the techniques dedicated to processing elements presented in Chapter 3, the ADE and ST0 encodings detailed in this chapter have very similar application targets. Therefore, an extensive comparison among the two has already been presented in Section 4.6. Specifically, we have shown that ADE is superior to ST0 for most of the considered data sets. This is due both to its lower complexity in terms of hardware implementation and to its superior results in terms of reduction of the number of transitions on the bus. Nonetheless, ST0 is still useful for data that are characterized by an alternation of short bursty phases and long correlated phases. The most notable domain that produces this type of serial data is image transmission, which is particularly relevant due to the high frequencies and large amounts of data involved (other signals such as ECG also produce similar streams). In summary, the most appropriate encoding for a given application should be chosen through a preliminary analysis of the statistics of the serial streams involved, as well as considerations on the impact of each algorithm on the application-level output quality.

At the end of this chapter, it is worth mentioning that a follow-up work inspired by our proposed Serial-T0 has been recently published in [133]. The authors of this work have proposed a new encoding algorithm called AXSERBUS, which includes three modes of operation. Two of these modes correspond exactly to the ones of ST0, i.e. the transmission of a special 0-TC word to indicate a repetition of the previous word, and the standard transmission of the unencoded binary value of a word. However, the authors propose to also use *1-TC patterns* to encode intermediate differences among subsequently transmitted values, which are small but not totally negligible. In particular, the $2(n-1)$ 1-TC patterns available for n-bit data are used to encode increasingly large positive and negative differences, in a non uniform way. Each pattern is used to represent a range of possible differences, hence introducing a further data approximation. Two thresholds D_0 and D_m control the ranges of data differences transmitted as 0-TC and 1-TC patterns respectively.

With experiments on image data, the authors of AXSERBUS show that their encoding achieves superior TC reductions and quality compared to both ST0 and ADE for image data. However, the additional operating mode complicates the corresponding encoding and decoding hardware, which appears to be almost one order of magnitude more power hungry than the circuits for ADE and ST0 (also AXSERBUS has been synthesized on 45nm technology, albeit from a different manufacturer). Therefore, a complete analysis that considers CODEC overheads, similar to the one of Section 4.6.5, should be performed to accurately compare this encoding with our proposals. Such analysis will be subject of future work.

Chapter 5

Low-Overhead Image Transformations for Energy-Quality Optimization in OLED Displays

In this chapter, we describe two techniques to achieve energy efficiency in *Organic Light-Emitting Diode* (OLED) displays. The required background on OLED energy optimization has been presented in Section 2.6.3. Both our proposed techniques exploit the content-dependent power consumption model of OLEDs to obtain savings through image transformations. These transformations are *adaptive*, meaning that although the functions they implement remain constant, their parameters change automatically depending on the considered input image. This adaptiveness tailors the "strength" of the power optimization to the current input, so that visual quality is not affected too drastically. Despite this common approach, the two transformations have significantly different goals. The first technique, presented in Section 5.3, is based on simple *brightness scaling* and leverages an adaptive method to identify the optimal scaling factor for each image. The second transformation, presented in Section 5.4, combines display power reduction with *image enhancement* to achieve better visual quality at the cost of a more complex implementation. The content presented in this chapter is an extended and revised version of our previously published papers found in [38, 55, 58].

5.1 Motivation

OLED displays are increasingly being used as an alternative to classic *Thin Film Transistor* (TFT) *Liquid Crystal Displays* (LCD) in mobile devices [202]. This is due to the numerous advantages of OLED technology compared to LCD, including higher brightness and better viewing angles, as well as the possibility of building thinner and flexible screens [203].

As anticipated in Section 2.6.3, the most peculiar feature of OLED displays is that they are composed of emissive devices and therefore do not require an external light source. This has the important consequence of making the power consumption of these displays strongly *image-dependent*. In particular, while OLEDs are more efficient than traditional LCDs on average, they tend to consume significantly more power for very bright images [202, 203]. Therefore, considering the strong contribution of the display subsystem on the total power consumption of many devices (see Figure 1.6b), a proper management of OLED energy consumption becomes fundamental [204, 140].

The main techniques for reducing energy consumption in OLEDs have been analyzed in Chapter 2. In particular, we have mentioned that the most flexible approaches are those that rely solely on *image transformations*. In fact, these methods do not require any hardware modification of the panel, and can be applied also to off-the-shelf displays. Two main types of transformations have been proposed in literature. Some are based on *brightness* or *luminance scaling*, i.e. a simple reduction of all (or a range of) pixel intensities in the image. Others leverage more complex nonlinear functions, that try to concurrently reduce power and enhance the image quality.

In formal terms, an energy efficient image transformation for OLEDs turns an input image I_i into an output image $I_o = T(I_i)$, so that the power consumption of the latter is reduced, while the perceived visual quality is preserved or enhanced, depending on the type of approach. The parameters of the transformation function T are generally image-dependent, so that the amount of power reduction and quality can be tailored to the displayed image. However, this implies that the parameters must be recomputed in *real time*, every time a new image is stored in the frame buffer of the display. Typical display refresh rates are in the order of 50-60 Hz, thus the computation of parameters and the application of T must be repeated every

15-20 ms. Moreover, OLEDs also allow faster response times, possibly making timing constraints even tighter in future scenarios [202].

Most OLED image transformations in literature pay very little attention to the analysis of time and energy overheads.

For what concerns transformations for quality enhancement, most proposed methods involve computationally intensive operations, such as the solution of nonlinear optimization problems [148, 149], or complex histogram processing [151, 150]. Implementing these operations under the real-time constraints described above is not trivial. A software solution could consume a significant percentage of the CPU time, not leaving space for other tasks. Alternatively, dedicated hardware could be used. In both cases, the implementation of the transformation may consume a significant amount of energy, thus reducing or even nullifying the benefits due to power reduction on the display.

Even literature focusing on *adaptive* brightness scaling, i.e. those approaches in which scaling is performed in an image-dependent way, has proposed methodologies whose execution times are unacceptable for a real-time application [147, 146]. The only solutions that tackle overheads reduction do it relying on a custom camera application [153, 154] and cannot be applied to images obtained from different sources.

The works described in this chapter focus specifically on implementing low-overhead image transformations for energy reduction in OLEDs. In particular, we propose one method for each group of transformations: the solution for brightness-scaling is simply called *Low-overhead Adaptive Brightness Scaling* (LABS), whereas the transformation for concurrent power saving and image enhancement is called *Low-overhead Adaptive Power Saving and Contrast Enhancement* (LAPSE). Both solutions are designed to favor a simple hardware acceleration. This allows to implement them in real time with very low overheads in terms of both power and time. Experimental results show that our proposed methods are able to obtain similar savings and image quality compared to the state of the art, despite this great reduction in complexity.

5.2 OLED Power and Image Similarity Models

OLED Power Consumption

Each pixel of an OLED panel is formed by three emissive devices, corresponding to the three components of the RGB color space [140]. The intensity of a component depends on the current flowing through the corresponding device. Therefore, the total power consumption of the panel is strongly dependent on the *brightness* of the displayed image, and secondarily on the balance between colors. Measurements on real panels in [140] allowed to build an empirical model for the power consumption of an OLED, which can be expressed by the following equation:

$$P_{tot} = \sum_{i=0}^{W} \sum_{j=0}^{H} (w_0 + w_r \cdot R_{i,j}^{\gamma} + w_g \cdot G_{i,j}^{\gamma} + w_b \cdot B_{i,j}^{\gamma}) \tag{5.1}$$

In this formula, W and H are the width and height of the panel, $(R_{i,j}, G_{i,j}, B_{i,j})$ are the sRGB components of the pixel at position (i, j), and w_x and γ are panel-dependent coefficients, obtained via characterization. For most displays, $\gamma \in [2:3]$. Notice that w_0 represents a baseline power contribution, present even when a pixel is black (i.e. when all three sRGB components are equal to 0). This component is not influenced by image transformations, which operate only on pixel intensities.

The Mean Structural Similarity Index

There is no universally accepted metric to express the "quality" of a generic image in numerical terms. Although there is a large body of literature on the so called *no-reference image quality assessment* [205], no single model has yet being accepted by the community, and the most reliable way to assess the output quality of an image transformation is to perform a *subjective test*, i.e. a comparative evaluation survey with a pool of human subjects [1].

In contrast, when the goal is to measure *similarity* among two images, the *Structural Similarity Index* (*SSIM*) is by for the most commonly used metrics [31].

[1]This is true only for generic images. There are domain-specific types of images (e.g. some medical images) for which quality can be measured by their usability from the point of view of a following processing task (e.g. a disease classification).

This index has been shown to correlate very well with the way humans perceive similarity. Mathematically, the standard expression of the *SSIM* for a gray scale image is the following:

$$SSIM(\bar{x}, \bar{y}) = \frac{(2\mu_{\bar{x}}\mu_{\bar{y}} + c_1)(2\sigma_{\bar{x}\bar{y}} + c_2)}{(\mu_{\bar{x}}^2 + \mu_{\bar{y}}^2 + c_1)(\sigma_{\bar{x}}^2 + \sigma_{\bar{y}}^2 + c_2)} \tag{5.2}$$

In (5.2), $\bar{x}$ and $\bar{y}$ are subsets of the image pixels, obtained via a sliding window. Moreover, $\mu_{\bar{x}}$, $\mu_{\bar{y}}$ and $\sigma_{\bar{x}}^2$, $\sigma_{\bar{y}}^2$ are the mean and variance of pixels gray levels in $\bar{x}$ and $\bar{y}$, and $\sigma_{\bar{x}\bar{y}}$ is their covariance. The coefficients c_1 and c_2 are constant, and are added for stabilization purposes, i.e. to avoid $0/0$ divisions (e.g. when both images are totally black, $\mu_{\bar{x}} = \mu_{\bar{y}} = 0$).

For computing the *SSIM* on color images, gray levels are replaced with the luminance components of each pixel. The latter can be extracted by converting pixels to an appropriate color space, such as YCbCr, where the Y component corresponds to luminance. The conversion from RGB to YCbCr and vice versa is obtained through linear transformations:

$$\begin{bmatrix} Y_{i,j} \\ Cb_{i,j} \\ Cr_{i,j} \end{bmatrix} = \left(\mathbf{C} \cdot \begin{bmatrix} R_{i,j} \\ G_{i,j} \\ B_{i,j} \end{bmatrix} \right) + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \tag{5.3a}$$

$$\begin{bmatrix} R_{i,j} \\ G_{i,j} \\ B_{i,j} \end{bmatrix} = \mathbf{K} \cdot \left(\begin{bmatrix} Y_{i,j} \\ Cb_{i,j} \\ Cr_{i,j} \end{bmatrix} - \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \right) \tag{5.3b}$$

where $\mathbf{C}, \mathbf{K} \in \mathbb{R}^{3x3}$ are matrices of constant coefficients [206]. Using the JPEG standard for YCbCr conversion [207], both RGB and YCbCr pixels are represented on 24-bit, with each component spanning the range $[0, 255]$.

SSIM values for all positions of the sliding window are then averaged to compute the global *Mean SSIM* (*MSSIM*). This allows to have a single real number in the range $[0 : 1]$ which is well correlated with the similarity among two images. In particular, an $MSSIM = 1$ corresponds to identical images.

5.3 Low-Overhead Adaptive Brightness Scaling

The proposed LABS technique modifies the “amount of brightness scaling” applied to each image, so to optimize the balance between power saving and image alteration with respect to the input. In practice, brightness scaling is applied more aggressively when it can result in larger power savings, and vice versa.

This result is achieved maximizing an image-dependent metric that combines power saving and image similarity in a single function. We show that the expression for such a metric can be derived analytically in closed form and has a single optimum for a given image. We then demonstrate that this optimum point can be put in relation with simple features of the input image (namely its total luminance) by means of regression. This allows to easily extract the optimum transformation parameter at runtime in real time, with small software or hardware overheads.

Our solution is significantly different from all previous brightness scaling techniques. In fact, standard brightness scaling is *static*, i.e. it is based on a fixed transformation that is irrespective of the considered image. Although some adaptive solutions exist [147, 146], they normally try to optimize power under a similarity constraint, or the opposite. Moreover, they result in very large performance overheads, that prevent a real-time application. Our method, in contrast, *co-optimizes* power and similarity, and can be applied in real time with small overheads.

5.3.1 Brightness Scaling and its Limitations

Brightness or luminance scaling [2] is one of the simplest possible image transformations. More specifically, it consists of a *pixel-by-pixel* mapping. For gray-scale images, it transforms every pixel according to the following function:

$$x' = k \cdot x \tag{5.4}$$

where x is the input pixel value and k is a real number called *scaling factor*. In general, to obtain power savings $0 \leq k \leq 1$. For color images, the same effect is obtained scaling the luminance component of a pixel, e.g. in YCbCr.

[2] *Brightness scaling* is the most commonly found term to define this transformation in the literature. A more correct term is the less common *luminance scaling*, as luminance is a measurable property of light, while brightness is subjective. In the following, we will use both terms interchangeably.

Despite its simplicity, brightness scaling is very effective in reducing OLED power consumption. Indeed, (5.1) and (5.3b) imply that $P \propto Y^{\gamma}$, hence reducing the luminance of all pixels has a strong effect on power [148].

However, the traditional implementations of brightness scaling are *static*, i.e. they use a single scaling factor for all images. The value of k can only be changed by the user of a device, who acts on a setting knob (e.g. on a smartphone). Alternatively, the firmware or operating system of the device can also control k in reaction to some external event, such as a change in the ambient illumination or a decrease in the battery state-of-charge. In both these scenarios, the time granularity at which the scaling factor is adapted is very coarse (in the order of seconds to hours). This is clearly sub-optimal, as both the achievable power savings with a given k and the corresponding impact on visual quality will differ among different images.

There exist also few *dynamic* brightness scaling solutions in literature, in which the scaling factor is automatically adapted to the display content with fine time granularity. For example, the authors of [147] propose an algorithm to obtain the value of k that minimizes power, under a constraints on the minimum allowed *MSSIM* (or equivalently on the maximum allowed image alteration). However, these adaptive brightness scaling algorithms involve very complex calculations, which make them virtually impossible to implement in real time. For instance, the extraction of k in [147] requires about 2 seconds of execution on a high-end processor, for a 1920x1080 pixels image.

Our method takes a different approach, as explained in the introduction to this section. Rather than minimizing power under a similarity constraints, it concurrently optimizes both power reduction and similarity for the considered image. This allows to obtain the optimal scaling factor for each image, in a fully automatic way.

Within a mobile device (e.g. a smartphone), our solution could be leveraged to replace the traditional brightness control knob with a single *on/off button*, that activates adaptive brightness scaling when desired. This would allow to automatically optimize the visibility and power consumption of each image, and relieve the user from the task of manually setting the desired brightness level.

5.3.2 MSSIM and Brightness Scaling

The starting point of our approach is the analysis of the relation between brightness scaling and image similarity. To measure the latter, we use the *SSIM* and *MSSIM* defined in Section 5.2.

The authors of [147] have noted that (5.2) can be simplified in the case in which the pixels in $\bar{y}$ are brightness scaled versions of those in $\bar{x}$, i.e. $\bar{y} = k\bar{x}$. In particular, the following relations hold:

$$\mu_{\bar{y}} = k\mu_{\bar{x}} \tag{5.5a}$$

$$\sigma_{\bar{y}} = k\sigma_{\bar{x}} \tag{5.5b}$$

$$\sigma_{\bar{x}\bar{y}} = k\sigma_{\bar{x}}^2 \tag{5.5c}$$

Consequently (5.2) can be rewritten as:

$$SSIM(\bar{x}, k\bar{x}) = \frac{(2k\mu_{\bar{x}}^2 + c_1)(2k\sigma_{\bar{x}}^2 + c_2)}{[(1+k^2)\mu_{\bar{x}}^2 + c_1][(1+k^2)\sigma_{\bar{x}}^2 + c_2]} \tag{5.6}$$

What the authors of [147] did not consider, however, is that this equation can be further simplified by introducing a small approximation. Indeed, as explained in Section 5.2, c_1 and c_2 are only introduced in the equation for stabilization purposes. Consequently, their value is chosen to be *very small* compared to the other terms, so not to impact on the value of the metric.

As a matter of fact, the values used as defaults for these constants are: $c_1 = (0.01L)^2$ and $c_2 = (0.03L)^2$, where L is the maximum pixel intensity, e.g. $L = 255$ in JPEG YCbCr [207]. Excluding the special case of extremely dark images (which are not relevant for energy efficiency in OLEDs anyway), it is easy to see that c_1 and c_2 can be removed from (5.6) with a negligible impact on the numerical value of the *SSIM*/*MSSIM*. In fact, even for a small scaling factor k, the terms $2k\mu_{\bar{x}}^2$ and $2k\sigma_{\bar{x}}^2$ are at least one if not two orders of magnitude larger than c_1 and c_2. With this additional simplification, the *SSIM* equation reduces to the following:

$$SSIM(\bar{x}, k\bar{x}) \approx \frac{(2k\mu_{\bar{x}}^2)(2k\sigma_{\bar{x}}^2)}{[(1+k^2)\mu_{\bar{x}}^2][(1+k^2)\sigma_{\bar{x}}^2]} = \frac{4k^2}{(1+k^2)^2} \tag{5.7}$$

where the rightmost expression comes from trivial simplifications of identical terms.

The fact that this expression only depends on the scaling factor k means that (for the great majority of images), the *SSIM* and consequently also the *MSSIM* are approximately *independent from image pixel values*.

An experimental confirmation of this conclusion is shown by the box plots of Figure 5.1. These graphs show the distributions of the *MSSIM* values obtained comparing all images from three different data sets [208, 177, 209] with their otherwise identical brightness-scaled versions. The scaling factor for this experiment has been fixed to $k = 0.8$. As shown in the plots, for the majority of images, the *MSSIM* varies by less than 0.5%.

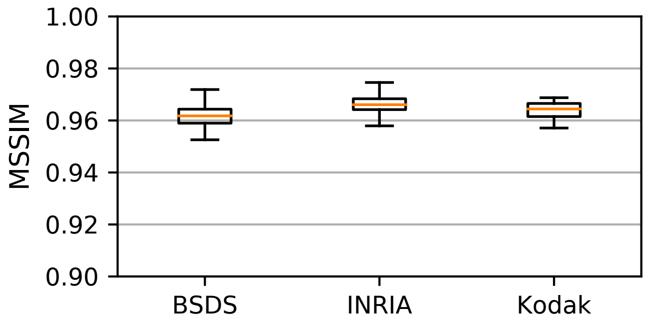


Fig. 5.1 Distribution of the *MSSIM* between original and brightness scaled images with a scaling factor $k = 0.8$, for three different image data sets.

This apparently counter-intuitive finding makes sense when considering that brightness scaling with a constant scaling factor applied to all pixels does not introduce any content-dependent image distortion. Therefore, any brightness scaled image receives the “same amount of alteration” with respect to the corresponding unscaled input, regardless of its content. Given that the *MSSIM* is a metric of image *similarity* or *alteration* (and not of image quality, albeit being often treated as such), it is reasonable that it takes the same value for any such pair of images.

5.3.3 The Power Reduction-Similarity-Product Metric

In the previous section, we have shown that the *MSSIM* of a brightness scaled image with respect to the input is roughly independent from pixel values. The same is not true for the power consumption of an OLED panel, as evident from (5.1).

Indeed, as mentioned in Section 5.3.1, the total power is proportional to the γ-th power of the total image luminance. Since brightness scaling reduces the luminance of each pixel of a factor k, the relation among the power P of an image and that of its brightness scaled version P_{SCAL} is:

$$P_{SCAL} \propto k^{\gamma} P \tag{5.8}$$

According to this equation, if a "stronger" brightness scaling (smaller k) is applied to an image with a high average luminance, the resulting power benefits are more significant, because of the larger initial power P.

At the same time, more luminous images are also those for which brightness scaling is more tolerable. In fact, although the numerical similarity ($MSSIM$) is invariant, dimming bright and dark images does not have the same effect on the perceived image *quality*. In the case of a bright image, scaling the luminance makes it more dull and generally less beautiful, but does not prevent the intelligibility of its content. In contrast, scaling an already dark image might bring to the partial or total loss of its information content.

The dependency of the achievable power savings (and information preservation) on the image content shows that an optimal brightness scaling approach should adapt the scaling factor k to the features of the input image.

To achieve this goal, we use a metric called *Power reduction Similarity Product* (PSP). The mathematical expression of the PSP is the following:

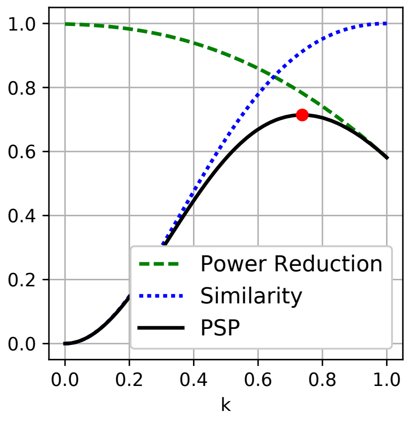
$$PSP(k,I) = \left(1 - \frac{P(kI)}{P_{MAX}}\right) MSSIM(I,kI) \tag{5.9}$$

where $P()$ is the total power consumption of the image computed with (5.1) and $MSSIM()$ is the approximate similarity metric for the case of brightness scaling, expressed by (5.7). Finally, P_{MAX} is the maximum power that can be dissipated by the OLED panel, i.e. that of a totally white image.

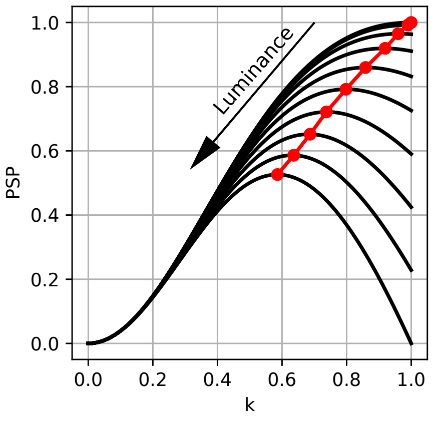
As the name says, the PSP is the product of two components; the first accounts for the achievable power reduction when brightness scaling is applied on the considered image, while the second measures the impact that this scaling has on image similarity. Specifically, the former is computed as the power reduction obtained by displaying the scaled image on the OLED panel, normalized to P_{MAX}. The latter is simply the $MSSIM$ between the original and scaled images. The normalization performed on power ensures that both components vary in the range $[0:1]$ regardless of the considered image.

Both components are functions of the scaling factor k. Moreover, both should be ideally maximized, since we would like to achieve large power savings and also maintain a good similarity with the input. However, this ideal goal is impossible, as the two components have opposite dependencies on k. For example, $k=1$ would yield perfect similarity (the output image would be identical to the input) but no savings, while $k=0$ would yield maximum savings (black image) but a very low similarity. Therefore, there exists a clear tradeoff among similarity and savings. By computing the product of these two terms, we build a combined metric that can be used to co-optimize two opposing goals.

The shape of the PSP and those of its two components as a function of k are shown in Figure 5.2a for one example image. The blue dotted curve is the approximate *SSIM*/*MSSIM* equation obtained in (5.7), while the green dashed curve represents the image-dependent power reduction component, which decreases proportionally to $-k^{\gamma}$. Due to normalization, the latter does not reach zero for $k=1$. This is because even when no scaling is performed, the considered image consumes only a portion of P_{MAX}, which depends on the intensity of its pixels.



(a) Example for one image.



(b) Maximum point variation as a function of luminance.

Fig. 5.2 Analysis of the proposed PSP metric for adaptive brightness scaling.

As a consequence of the opposite trends of power and similarity with respect to k, the PSP always has a single maximum, highlighted with a red dot in the example of Figure 5.2a. The value of k corresponding to this point is the scaling factor that achieves the best balance between power reduction and similarity for that image: $k_{opt} = arg\max(PSP(k,I))$.

Figure 5.2b shows how the maximum point of the PSP changes for images with different total luminance. The curves have been generated computing our metric on monochrome images, with average luminance values ranging from 0 (black) to 255 (white). As luminance increases, the power reduction curve becomes more steep. At the extreme, a totally white image produces a power consumption $P(I) = P_{MAX}$. Therefore, the dashed green curve of Figure 5.2a would reach 0 for $k = 1$ in this case. Because of this increasing steepness, k_{opt} moves towards lower values as luminance increases, expressing the intuition that that *the optimal scaling factor is smaller for brighter images*.

For a fully black image obviously $k_{opt} = 1$; indeed, power consumption is already minimal and scaling has no effect. At the other extreme, i.e. for a white image, the optimal scaling factor converges to $k_{opt} \approx 0.59$.

5.3.4 Low-Overhead Adaptive Brightness Scaling

In a single sentence, LABS computes the optimal scaling factor k_{opt} according to the previously described PSP metric for each new image and then applies a standard luminance scaling with that factor. Doing so at runtime, however, involves some additional technical measures.

In fact, computing k_{opt} directly from (5.9) would require computing the derivative of that equation with respect to k, and finding its single stationary point, i.e. the value of k that verifies:

$$\delta PSP(k,I)/\delta k = 0 \tag{5.10}$$

Even using the approximate *MSSIM* expression of (5.7), the PSP derivative still includes divisions, as well as γ-th and $(\gamma - 1)$-th roots, computed *for every pixel* of the image.

To obtain runtime adaptive scaling, k_{opt} should be recomputed for each new image with minimal energy overheads. Thus, it must be either computed in software, in very short time, or accelerated with simple hardware.

Since in realistic OLED models γ is not integer, the power operations described above would have to be implemented with numerical methods [32]. Implementing those operations, as well as divisions, at runtime, under the constraints determined by the display refresh rate, would either consume most of the main system CPU time just for transforming images, or require an additional dedicated processor, with huge energy overheads.

Therefore, LABS uses an alternative approach to obtain k_{opt} at runtime. This method is based on regression, and only provides an estimate of the optimal scaling factor. Nonetheless, as shown in the following experimental results, the regression error is low on average, while allowing to obtain k_{opt} with much lower overheads compared to the exact method.

Specifically, k_{opt} is put in relation to the *total luminance* Y_{tot} of the considered image, in accordance to the observation of Section 5.3.3 that power, and hence k_{opt}, are strongly affected by luminance. Computing Y_{tot} at runtime is straightforward, and just requires summing the Y component of each pixel in YCbCr space. The conversion to YCbCr, e.g. from RGB, is not an additional cost, as it is already part of most brightness scaling approaches.

Building such model model requires an initial off-line training phase in which Y_{tot} and the corresponding exact value of k_{opt} are computed using (5.9) for a set of representative images. These pairs of values are then used to determine the parameters of the regression equation.

With the goal of minimizing overheads we use *linear regression* to model this dependency, due to its extremely low evaluation complexity. Therefore, the computation of k_{opt} at runtime in LABS reduces to the evaluation of the following equation:

$$k_{opt} = mY_{tot} + q \tag{5.11}$$

at the cost of one multiplication and one addition *per image*, with constant coefficients. These operations have low energy and time overheads, both in software and in hardware, allowing to adapt the scaling factor at runtime in an efficient way.

Regardless of the selected model form, this method clearly produces an approximation of the real k_{opt}. In fact, it ignores that pixel chroma components contribute to power (and hence affect k_{opt}), although with a smaller impact compared to luminance [140]. Moreover, small image-dependent *MSSIM* variations are also neglected, due to the use of the simplified equation reported in (5.7). However, our experiments show that despite these approximations and the simple model selected, the estimation error produced with this method is low.

Hardware Implementation

A block diagram that summarizes the operations performed by LABS is shown in Figure 5.3. While purposely kept general, this figure can be easily mapped to a hardware implementation of our proposed technique. In particular, after RGB to YCbCr conversion, a simple accumulator is leveraged to compute Y_{tot} by summing the luminance of all pixels. Then, the linear regression model is evaluated using one multiplier and one adder, to extract k_{opt}. Finally, the luminance component of each pixel is scaled by means of another multiplication, which can be implemented by the same hardware block of the previous, since they are performed in disjoint times.

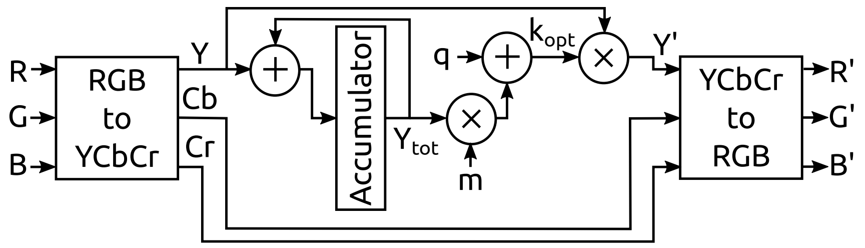


Fig. 5.3 Block diagram of the operations involved in the proposed LABS technique for OLED power and image similarity optimization.

Overall, the complexity of this additional hardware is minimal, even compared to that required to implement simple color space conversions.

5.3.5 Experimental Results

Setup

We have assessed the performance of the proposed LABS algorithm both on still images and on videos. For experiments on images, we have considered three publicly available data sets: the classic *Kodak* data set (24 images) [177], the *INRIA holiday data set*, (800+ images) [209] and the *Berkeley Segmentation Data Set* (BSDS), (500 images) [208]. All three contain images with a wide variety of subjects, luminance and contrast. Notice that, although we have experimented on natural images due to the availability of many standard data sets, the proposed method is equally applicable also to GUIs, as any other form of brightness scaling [147]. For experiments on videos, we have used sequences from the *Open Video Project* [210] and from the *Derf's Test Media Collection* [211].

To estimate the power consumption of an OLED panel we have used the model of (5.1), with the same numerical coefficients used in [148], i.e. $(w_0, w_r, w_g, w_b, \gamma) = (0, 70, 115, 154, 2.2)$.

The training phase of LABS and the computation of regression coefficients have been implemented using Python 3.5. The runtime adaptive scaling part has been implemented both in software and in hardware. The software version has been written in C and compiler with LLVM 3.9.1. Its execution time has been measured on a Intel Core i7 processor running at 2.2GHz, with 16GB of RAM. The hardware version has been specified in VHDL and synthesized on a commercial 45nm standard cell library from ST Microelectronics. Synthesis has been performed using Synopsys Design Compiler L-2016.03, setting the clock frequency of all blocks included in Figure 5.3 to 1 GHz. In this case, the execution time has been evaluated through gate-level simulations of the synthesis output performed in Mentor QuestaSim 10.6, whereas power consumption has been estimated in Synopsys PrimeTime L-2016.06.

We compare LABS against standard brightness scaling, with a single image-agnostic factor k, to show the advantages deriving from online adaptiveness. Moreover, we also compare it with a state-of-the-art adaptive scaling technique for OLEDs [147] in terms of software implementation performance.

Average Power Saving and Adaptiveness

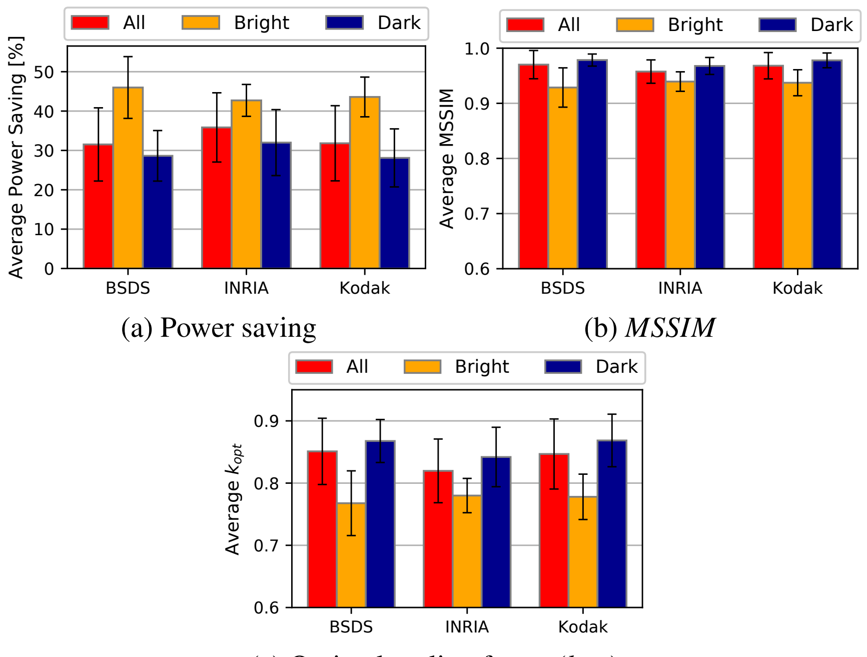
In this first experiment, we have computed the average power saving obtained by LABS on the three still images data sets. We have also analyzed the corresponding average *MSSIM* and optimal scaling factor k_{opt}. Power saving has been computed with respect to the corresponding image without brightness scaling, in percentage:

$$P_{SAV} = \left(1 - \frac{P_{SCAL}}{P_{ORIG}}\right) \cdot 100 \tag{5.12}$$

Furthermore, to show the benefits deriving from image-adaptiveness in LABS, we have repeated the same evaluation after splitting each data set into two subsets.

Specifically, defining L as in Section 5.3.2, the first subset includes images whose average luminance is $< 0.5L$, denoted as *Dark*, while the second includes all remaining images, denoted as *Bright*.

The results of this experiment are shown in Figure 5.4, where the heights of the bars represent average values over the corresponding set of data, and black segments indicate the intervals defined by the standard deviation.



(a) Power saving (b) *MSSIM*

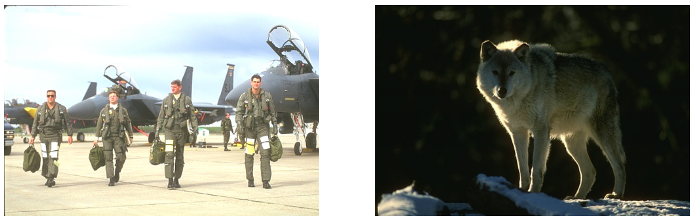
(c) Optimal scaling factor (k_{opt})

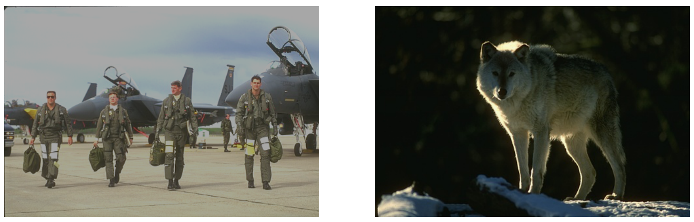
Fig. 5.4 Average power saving, *MSSIM* and scaling factor obtained by the proposed LABS algorithm for adaptive brightness scaling in OLEDs, before and after splitting each data set into dark and bright subsets.

The average power reduction obtained by LABS is superior to 30% for all three data sets. Moreover, the variability for different images and different data sets is relatively low. When considering bright and dark images separately, as expected, the largest savings are achieved for the former, due to the smaller values of k_{opt} selected by our method.

The *MSSIM* between input images and their brightness scaled counterparts is > 0.97 on average, when considering all three data sets in their entirety. While luminous images are transformed more aggressively, the similarity remains larger than 0.92 also for the *Bright* subsets.

A visual example of the benefits of our adaptive approach is shown in Figure 5.5. In this experiment, we have compared the visual quality obtained processing two images of different luminance with adaptive scaling factors, with the results obtained on the same images using a *single k* for both. The latter is the output that would be produced by a standard, image-agnostic brightness scaling approach. To make this comparison fair, we have computed a scaling factor that, when applied to both images, produces the same total power saving of LABS.


(a) Original


(b) Adaptive scaling factor

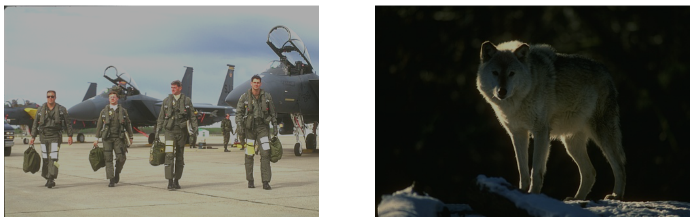

(c) Uniform scaling factor

Fig. 5.5 Comparison between the proposed LABS algorithm for adaptive brightness scaling in OLED displays and a standard uniform scaling.

According to the model of (5.1), the total power consumption for displaying the two images of Figure 5.5a on an OLED panel is mostly determined by the leftmost, brighter one ($\approx 95\%$ of the total). The rightmost image only contributes for $\approx 5\%$.

When LABS is applied to these two images, the scaling factor that is obtained for the leftmost image is $k_{opt,l} \approx 0.68$. Due to its lower luminance, the rightmost image is processed with a much less aggressive scaling, and specifically with $k_{opt,r} \approx 0.96$. The LABS outputs obtained using these two factors are shown in Figure 5.5b. As expected, the leftmost image has become less bright, thus obtaining a large power reduction. However, its content is still perfectly understandable. In contrast, the rightmost image is practically unchanged. This is because the saving that could be obtained by scaling it more aggressively is not proportionate to the corresponding similarity loss. The total power reduction obtained for the two images is 55.1%.

The results of image-agnostic brightness scaling are shown in Figure 5.5c. The common scaling factor has been computed with a bisection method, in order to achieve the same total savings as LABS (55.1%). The result of this computation is $\tilde{k} \approx 0.70$. As expected, this value is similar to $k_{opt,l}$, since the leftmost image accounts for most of the total power. However, applying the same scaling factor to both images is clearly inferior to our approach; while there are no visible benefits on the brighter image, the darker one is significantly duller, and many details are lost.

Application to Video Sequences

The advantages of LABS are even more evident for videos. We have processed a number of video sequences from [210] and [211] with LABS and put them side by side with the corresponding unprocessed videos for comparison. The results of this experiment have been uploaded to the URL found in [212].

Figure 5.6 shows the variation of the scaling factor identified by LABS over time, for one of these sequences. Namely, the graph refers to the *Nasa Anniversary 01* clip from [210]. In order to better understand the variations of k_{opt} over time, some key frames of the video have been superimposed to the graph.

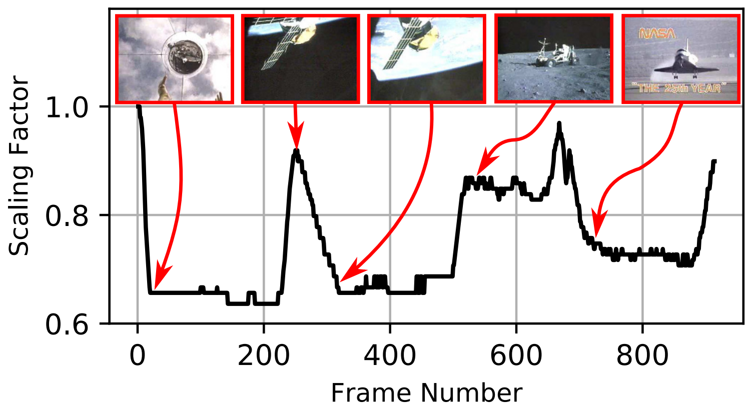


Fig. 5.6 Sequence of scaling factors obtained by the proposed LABS algorithm for a video clip with varying luminance.

Notice how our algorithm adapts the scaling factor to the content of the video, applying a more aggressive scaling when brightness increases, and vice versa. The sequences uploaded to [212] also show that, although the scaling factor is recomputed for every new frame, this does not generate any flickering in the videos. In fact, abrupt changes in k_{opt} only occur in correspondence of scene changes, when the brightness of the frame changes significantly. Within a single scene, k_{opt} remains approximately constant, and LABS does not generate visible artifacts.

Regression Analysis

LABS uses a regression model to obtain a simplified relation between the total image luminance Y_{tot} and the optimal scaling factor k_{opt}, which enables low-overhead runtime adaptiveness.

In order to verify the effectiveness of this approach, we have validated it using BSDS images [208], which are already conveniently split into training and test subsets. We have computed the total luminance Y_{tot} and the correct value of k_{opt}, obtained directly from (5.9), for all images in the data set. Then, we have used the results obtained on the training subset to determine the parameters of the linear regression model of (5.11), i.e. the slope m and the bias q, using ordinary least squares. Finally, we have validated this model against the results obtained on the test subset.

The model output and test data are shown in Figure 5.7, in which the total luminance has been normalized to a $[0 : 1]$ range. The caption of the figure also reports the numerical value of the model parameters, as well as the regression scores and errors.

The R^2 value close to 1 confirms the goodness of the model. More importantly, the small *RMSE* and *Maximum Absolute Error* (MAXE), show that the difference between exact and predicted k_{opt} values is negligible. Specifically, the error is never larger than 0.05, and in most cases it is smaller than 0.01, a difference that is completely indistinguishable for the human eye. These errors are due both to the simple model form and to the approximations involved in it (i.e. the fact that the effects of chroma on power are neglected, and the simplified *MSSIM* equation used). However, their limited magnitudes show that the proposed model can achieve a good accuracy despite these simplifications.

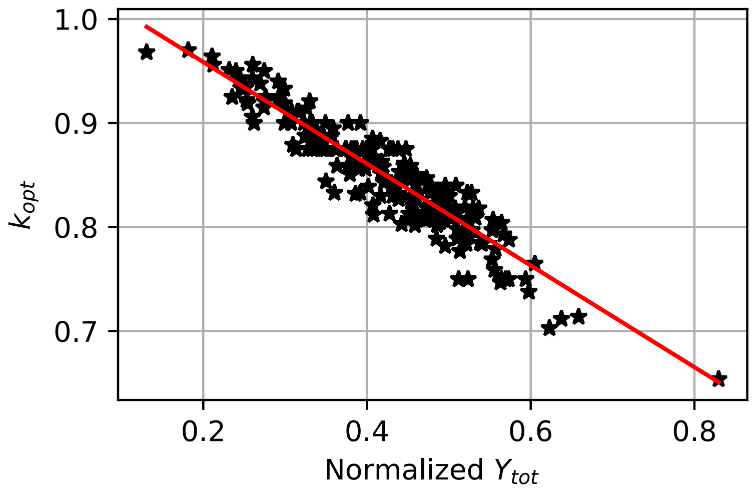


Fig. 5.7 Linear regression model validation for the proposed LABS algorithm. Model slope: $m = -0.34$, bias: $q = 1.05$. Scores: $R^2 = 0.89$, $RMSE = 0.01$, $MAXE = 0.05$.

Software and Hardware Implementation Results

The results of hardware and software implementations of LABS are reported in Table 5.1. These refer to the design and evaluation setup described at the beginning of the experimental results section. Time and energy results are reported for two different image sizes. Given that all operations that compose the runtime part of LABS (shown in Figure 5.3) have data independent complexity, these are valid for any image of the same size. The software execution time reported is the average over 1000 runs.

	SW	**HW**				
Image Size	**Time [ms]**	**Net Count**	**Area [mm^2]**	**Power [mW]**	**Time [ms]**	**Energy/ Frame [μJ]**
512x512	0.49	2557	0.0058	2.89	0.52	1.50
1920x1080	4.59				4.14	11.97

Table 5.1 Software and hardware implementation results of the proposed LABS algorithm for adaptive brightness scaling in OLEDs.

Software execution time does not include the contribution of color-space conversion phases (RGB to YCbCr and vice versa), in order to fairly compare our approach with other image transformations. The authors of [147] report a processing time of

1.78 μs per pixel for their technique. For a 512x512 pixels image, this corresponds to a total execution time of 466.6 ms. For the same image size, our method requires less than 0.5 ms, corresponding to a speedup of almost 1000x. Although the objectives of the two methods are slightly different ([147] minimizes power under a quality constraint) this speedup is remarkable. In particular, differently from [147] our method can easily meet the runtime constraints imposed by the display refresh rate.

The results of the hardware version, instead, refer to the full transformation, including color-space conversions. The hardware that we have implemented uses fixed-point to represent all intermediate quantities (m, q, k_{opt}, etc.). The bit-width of each variable is selected to ensure an average error on the output pixels smaller than 0.5% with respect to software, which uses double precision floating point. The very small energy and area costs and the short execution times obtained for this version confirm the benefits of the proposed technique.

For sake of comparison, the datasheet of a small 240x320 pixels AMOLED panel reports a power consumption of 260 mW in typical conditions [213]. Assuming a refresh rate of 15 ms, this corresponds to an energy consumption per frame of $0.26 \cdot 0.015 = 3.9 \cdot 10^{-3} J$. Despite the fact that the panel is even *smaller* than the minimum image considered in Table 5.1, the energy consumption of the display is three orders of magnitude higher compared to that of the additional hardware implementing the transformation. This proves that the theoretical savings computed in previous experiments will be preserved almost exactly in a real system.

Notice also that hardware results in Table 5.1 refer to pixel-serial implementations, that process one pixel per clock cycle. However, parts of the architecture of Figure 5.3 (e.g. the final scaling of all pixels luminance) would easily lend themselves to parallelization, allowing to further explore the execution time versus cost design space.

5.4 Low-Overhead Adaptive Power Saving and Contrast Enhancement

The technique presented in this section belongs to the category of image transformations that try to concurrently achieve power savings and image enhancement. Our method, called *Low-overhead Adaptive Power Saving and Contrast Enhancement*

(LAPSE), is specifically designed to favor a hardware acceleration of the involved image transformation, with very small energy costs. This is achieved thanks to two key factors:

- The use of a "hardware-friendly" shape for the transformation function T.
- The offloading of part of the computational burden to a training phase, performed offline.

A high-level scheme of the functionality of LAPSE is shown in Figure 5.8. The computationally intensive *Offline Phase* produces a fitting model that puts in relation simple features of an image (μ and σ^2, detailed in the following) with the corresponding optimal parameters (**a**) of the transformation function T. This model is then used to reduce the complexity of the *Online Phase*, which becomes *linear* in the size of the panel. During the latter, features μ and σ^2 are computed for the target image, and the parameters of T are derived according to the fitting model. Finally, the actual low-complexity transformation is applied to produce a low-energy and high-quality output image.

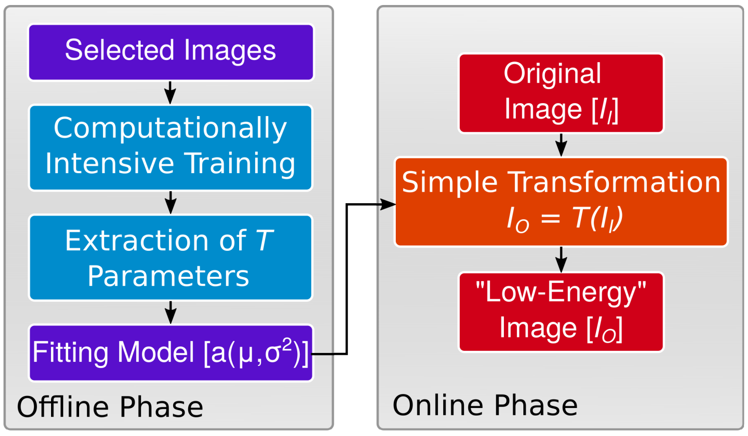

Selected Images
Computationally Intensive Training
Extraction of T Parameters
Fitting Model [a(μ,σ^2)]
Offline Phase
Original Image [I_I]
Simple Transformation $I_O = T(I_I)$
"Low-Energy" Image [I_O]
Online Phase

Fig. 5.8 Flow of the proposed LAPSE method for power saving and image enhancement in OLEDs.

This high-level scheme is similar to the one proposed for LABS. However, the two methods differ in the method used for training, in the selected set of image features, and clearly also in the shape of the transformation function T, which for LAPSE is more complex than a simpler brightness scaling.

In the experimental results of Section 5.4.5 we will show that LAPSE yields power savings and visual quality results comparable to those of previous solutions [148, 151, 149, 150], yet significantly reducing the computational complexity of the transformation.

5.4.1 Theoretical Foundation

This section presents the basic theoretical foundation of LAPSE, and motivates the main choices done in the following, most importantly the selection of the transformation function T. In Sections 5.4.2 and Sections 5.4.3 we will then detail the offline and online phases of the flow of Figure 5.8, respectively.

Transforming Luminance

Regardless of the involved optimizations, most of the previous image transformations for OLED power reduction [148, 151, 149, 150] eventually produce a *pixel-by-pixel* intensity mapping. The latter normally involves only the luminance component of each pixel, while chroma components are left untouched. This choice is motivated by the fact that, especially for photos and videos, color alteration dramatically affect the perceived visual quality [148–150]. In summary, the image transformation function reduces to:

$$Y_{out,i,j} = T(Y_{i,j}) \quad \forall\, 0 \leq i < W, 0 \leq j < H \tag{5.13}$$

where $Y_{i,j}$ and $Y_{out,i,j}$ are respectively the input and output luminance components of the pixel in position (i, j). LAPSE also follows this approach. The main difference with previous literature is the selection of the shape of T.

Since most systems internally store images in RGB, pixels must be converted to YCbCr before applying T. Therefore, the complete image transformation is composed of three steps: (i) color space conversion from RGB to YCbCr, (ii) application of an intensity mapping to the first component of each YCbCr pixel (iii) conversion of the output image back to RGB. The operations involved in color space conversions have been introduced in (5.3a) and (5.3b).

Choice of the Generic Transformation Function

Power reduction in OLED displays can be trivially achieved by decreasing pixels luminance, which is the approach followed by brightness scaling techniques (see Section 5.3). However, pure luminance reduction degrades significantly the perceived quality. Therefore, as mentioned in Section 2.6.3, most techniques that aim at concurrent power saving and image quality improvement combine a reduction of the total luminance with *contrast enhancement* [26, 148, 151, 149, 150].

The shapes of the pixel intensity mappings produced by these solutions share some common characteristics. In order to enhance contrast, they are generally non-linear. Additionally, in most cases, they have non-monotonic concavity, in order to alter the contrast of dark and bright areas of the image differently [148, 151].

Since the primary goal of our approach is overhead containment, the chosen form of T must be computable with simple operations. One obvious choice is to use polynomials, which only require additions and multiplications, i.e. some of the least expensive operations to implement both in software and in hardware. In general, the lower the order of a polynomial, the less operations are required to evaluate it. Therefore, in an attempt to minimize overheads, the ideal choice would be to use a third order polynomial, which is the lowest order transformation that can take shapes similar to those observed in previous solutions, since it can have varying concavity.

Figure 5.9 shows two motivating examples for the choice of this pixel transformation. Red solid curves represent luminance mappings obtained by the PCCE algorithm of [148] for the famous *Lena* and *Monarch* images. The main parameter β of PCCE has been fixed to the default value of 1 for this experiment. Black dashed lines are the best third order polynomial fittings of the PCCE output. As shown, cubic curves can approximate PCCE output transformations very closely for both images.

For a more quantitative estimate of this similarity, we have determined the best cubic fitting of the PCCE output when the latter is applied to each of the images in the two datasets that will be used for our experiments of Section 5.4.5. Table 5.2 reports the average R^2 score and $RMSE$ obtained for each dataset, and the corresponding standard deviations. The small error values and high scores confirm that, despite its simplicity, a cubic polynomial is effective in producing a very similar mapping to previous power reduction and contrast enhancement algorithms.

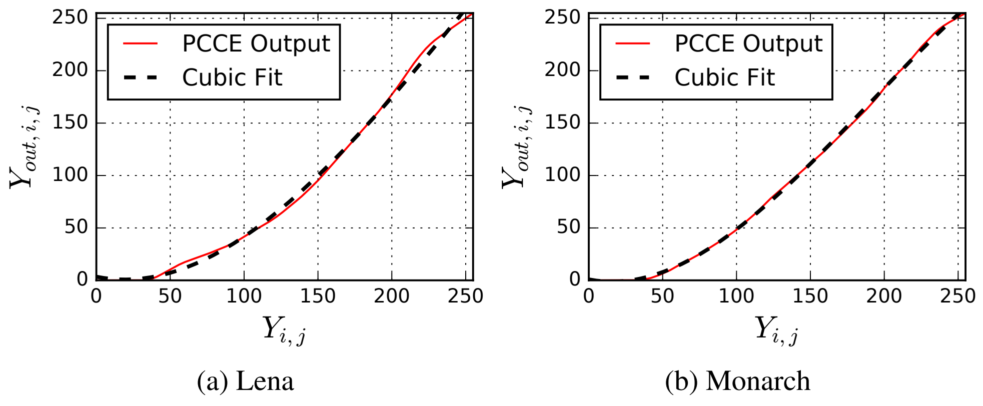


Fig. 5.9 Output of the state-of-the-art PCCE algorithm for OLED power reduction and image enhancement for two example images and corresponding cubic fittings. $RMSE = 1.6\%$ and 0.6% respectively.

Data	**R^2**	**RMSE [%]**
Live	0.999 ± 0.001	0.858 ± 0.376
BSDS	0.997 ± 0.003	1.464 ± 0.907

Table 5.2 Cubic fitting of the pixel transformations generated by the state-of-the-art PCCE algorithm for OLED power reduction and image enhancement, aggregate results for two image data sets.

Given this analysis, in LAPSE we transform the luminance of each pixel according to the following equation:

$$Y_{out,i,j} = T(Y_{i,j}) = a_3 Y_{i,j}^3 + a_2 Y_{i,j}^2 + a_1 Y_{i,j} + a_0 \tag{5.14}$$

The coefficients a_x in (5.14) are set to a numeric value that minimizes power while enhancing contrast, and are adapted to the target image, as described in Section 5.4.2.

Even in its most general form, (5.14) only requires 6 multiplications and 4 additions per pixel. However, the degrees of freedom for the general expression of T can be further reduced imposing some additional constraints, similar to those of [148]. Specifically, we constrain the transformation T to:

1) **Have an output that spans the entire range of luminance intensities**, i.e. $[0 : 255]$ in JPEG YCbCr. Since one of the goals of the transformation is to enhance contrast, it is desirable to have it span the full range of luminance values, so that the

image dynamic range is preserved. This is achieved imposing:

$$T(Y_{min}) = Y_{min} \qquad\qquad T(Y_{max}) = Y_{max} \tag{5.15}$$

where Y_{min} and Y_{max} are the minimum and maximum luminance values, i.e. $Y_{min} = 0$ and $Y_{max} = 255$ in YCbCr. Substituting (5.15) into (5.14) yields:

$$a_0 = 0 \qquad\qquad a_3 = \frac{1 - a_1 - 255a_2}{255^2} \tag{5.16}$$

and consequently:

$$Y_{out,i,j} = \frac{1 - a_1 - 255a_2}{255^2} Y_{i,j}^3 + a_2 Y_{i,j}^2 + a_1 Y_{i,j} \tag{5.17}$$

2) **Be monotonically increasing in $[\mathbf{0}:\mathbf{255}]$.** This constraint is to avoid the creation of artifacts due to inversions of luminance relations between the input and output images [148]. It allows to set a relation between the two remaining coefficients a_1 and a_2 and defines the search space for the optimal (image dependent) transformation parameters. Monotonicity is imposed forcing the derivative of T to be non negative for the entire luminance range:

$$T'(Y_{i,j}) = 3\frac{1 - a_1 - 255a_2}{255^2} Y_{i,j}^2 + 2a_2 Y_{i,j} + a_1 \geq 0,\ \forall Y_{i,j} \in [0, 255] \tag{5.18}$$

Geometrically, $T'(Y_{i,j})$ is represented by a parabola. The *forbidden region*, i.e. the region of plane that the parabola must not intersect in order to meet the constraint expressed by (5.18), is shown in Figure 5.10. The figure also reports one example for each of the four families of parabolas that meet the constraint, together with the corresponding mathematical relations on a_1 and a_2. The four families differ in the orientation of the curves, determined by the sign of the coefficient that multiplies $Y_{i,j}^2$, and in the value of the two solutions of $T'(Y_{i,j}) = 0$, computed as:

$$Y_1, Y_2 = \frac{-a_2 \pm \sqrt{a_2^2 - \frac{3}{255^2}(1 - a_1 - 255a_2)a_1}}{\frac{3}{255^2}(1 - a_1 - 255a_2)} \tag{5.19}$$

Parabolas with upward orientation (i.e. tending to $+\infty$ for $Y \to \pm\infty$) assume negative values when Y is between Y_1 and Y_2. In particular, parabolas with upward orientation and no real solutions are always positive, hence they meet the constraint.

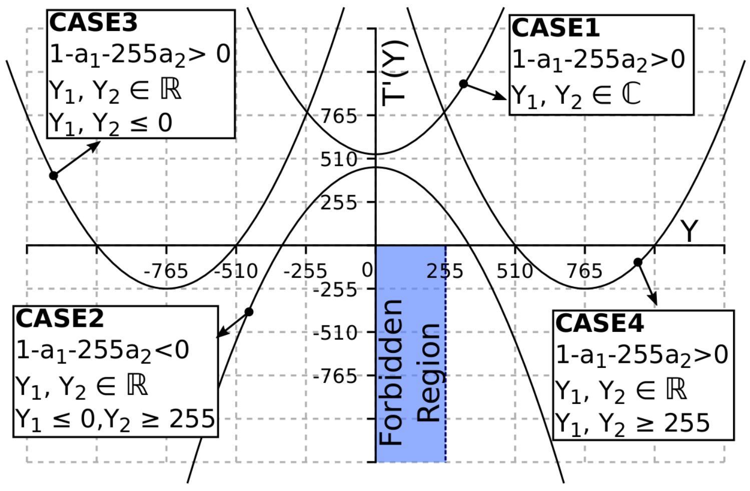


Fig. 5.10 Examples of first-order derivatives that ensure a monotonically increasing transformation in the proposed LAPSE method for OLED power reduction and image enhancement.

This is represented by the example labeled CASE1 in Figure 5.10. Similarly, if there are real solutions, but they are both smaller than 0 or greater than 255, the negative part of the curve will not intersect the Forbidden Region. Examples of these two categories are shown as CASE3 and CASE4. These groups also include the limit cases of two coincident solutions.

In the case of downward orientation, conversely, the curve is *positive* for $Y \in [Y_1, Y_2]$. Therefore, to meet (5.18), we must have $Y_1 \leq 0$ and $Y_2 \geq 255$, as in the example labeled CASE2.

These four subsets can be further reduced to two based on the following observation: the inflection point of $T(Y_{i,j})$ i.e. the point in which the cubic pixel transformation changes concavity, corresponds to the stationary point (minimum or maximum) of $T'(Y_{i,j})$. For CASE3 and CASE4, this point always occurs outside of the $[0,255]$ interval. Hence, using coefficients from these two cases would significantly reduce the flexibility of the transformation, by preventing the occurrence of concavity changes in the interval spanned by luminance values.

Additionally, the relations between a_1 and a_2 that can be derived in CASE3 and CASE4 define unbounded regions (semi-planes). Thus, they do not permit the identification of a finite domain for the search of the optimal values of these coefficients.

For these reasons, we discard CASE3 and CASE4, and limit our analysis to CASE1 and CASE2. Explicit relations between a_1 and a_2 in the latter two cases are obtained simply solving the two systems of inequalities reported in the labels of Figure 5.10, after substituting Y_1 and Y_2 with (5.19). The results are two bounded plane regions:

$$R_{CASE1} : \begin{cases} 0 \leq a_1 \leq 2 \\ \frac{1-a_1}{255} < a_2 < \frac{3-2a_1}{255} \end{cases} \tag{5.20a}$$

$$R_{CASE2} : \begin{cases} 0 \leq a_1 \leq 4 \\ \frac{-3a_1-\sqrt{3(4a_1-a_1^2)}}{2\cdot 255} \leq a_2 \leq \frac{-3a_1+\sqrt{3(4a_1-a_1^2)}}{2\cdot 255} \end{cases} \tag{5.20b}$$

The union of R_{CASE1} and R_{CASE2} is the largest bounded area of $\mathbb{R}^2$ for which T is monotonically increasing in $[0:255]$ and defines the search domain used in the following to determine the optimal transformation coefficients.

5.4.2 Offline Phase

In Section 5.4.1 we have determined the mathematical form of the generic image transformation T used in LAPSE, and the search domain for its free parameters a_1 and a_2. The offline phase of LAPSE consists in the optimization of the value of these two parameters for a set of training images. The information gathered during training is then used to identify a relation among the optimal values of a_1 and a_2 and simple quantitative features of the target image.

During training, parameters are optimized according to the cost function:

$$F(I) = w_p \cdot P_{tot}(I) - \sigma(Luminance(I)) \tag{5.21}$$

In this equation, I represents the image pixel matrix and P_{tot} is its total power consumption on an OLED panel, obtained according to the model of (5.1). $Luminance(I)$ corresponds to the operation of extracting the luminance component of all pixels, e.g. from RGB using (5.3a) and σ is the standard deviation of the luminance matrix, defined as:

$$\sigma = \sqrt{\frac{\sum_{i=0,j=0}^{W,H}(Y_{i,j} - \mu)^2}{W \cdot H}} \tag{5.22}$$

where μ is the mean luminance:

$$\mu = \frac{\sum_{i=0,j=0}^{W,H} Y_{i,j}}{W \cdot H} \tag{5.23}$$

The function in (5.21) tries to concurrently *minimize power consumption* (P_{tot}) and *maximize contrast*. The latter is measured by means of σ, which quantifies the spread of the luminance histogram and is therefore a simple measure of contrast. The two goals are balanced by means of a weighting factor w_p.

The optimization performed during training is also constrained. In particular, we limit the amount of image alteration that can be introduced by the transformation. This constraint, not present in previous solutions [148, 151, 149, 150], controls the impact of our method limiting the "strength" of the modifications that can be applied to the image. Alteration is measured by the *MSSIM*, i.e. the mean of (5.2) over the entire image. During training, we only consider solutions for which $MSSIM(\mathrm{I}, \mathrm{I_t}) \geq MSSIM_{\min}$, where I and $\mathrm{I_t}$ are the input and transformed images, and $MSSIM_{\min}$ is a user-imposed threshold.

The pseudocode for the training phase of LAPSE is reported in Figure 5.11. For clarity, the two transformation parameters have been grouped to form a vector $\mathbf{a} = [a_1,\ a_2]$. Matrices **Y**, **Cb** and **Cr** indicate the three YCbCr components of image I, while the subroutines YCbCrComponents() and Image() encompass color space conversions and components separation/grouping. F and $\mathrm{F_{opt}}$ are the current and optimal values of the objective function. The remaining symbols are defined as in previous sections.

The additional constraint $P_{tot}(I_t) < P_{tot}(I)$ in line 9 ensures that the transformed image consumes less than the original. It is inserted because power reduction is the primary goal of LAPSE, and is always favored over pure contrast enhancement.

The initial settings of the parameters $\mathbf{a_{opt}} = [1,\ 0]$ corresponds to the identity transformation, i.e. $T(\mathbf{Y}) = \mathbf{Y}$, as evident from (5.17). The search space for $\mathbf{a_{opt}}$ is explored exhaustively, with a granularity that depends on the desired accuracy. In our experiments, we have considered a grid of 40000 points. Exhaustive search is necessary for global optimality, since both the objective function and the constraints are not convex [214]. However, the resulting computational effort is acceptable,

```
procedure LAPSE TRAINING
    for I ∈ Training Images do
        ($\mathbf{Y}$, $\mathbf{Cb}$, $\mathbf{Cr}$) = YCbCrComponents(I)
        $\mathbf{a_{opt}}$ = [1, 0]
        $F_{opt}$ = F(I)
        for $\mathbf{a} \in R_{CASE1} \cup R_{CASE2}$ do
            $\mathbf{Y}_t = T(\mathbf{Y}, \mathbf{a})$
            $I_t$ = Image($\mathbf{Y}_t$, $\mathbf{Cb}$, $\mathbf{Cr}$)
            if $F(I_t) < F_{opt}$ and $P_{tot}(I_t) \leq P_{tot}(I)$ and
            $MSSIM(I,I_t) \geq MSSIM_{min}$ then
                $F_{opt} = F(I_t)$
                $\mathbf{a_{opt}} = \mathbf{a}$
            end if
        end for
        Store [$\mathbf{a_{opt}}$, $\mu(\mathbf{Y})$, $\sigma(\mathbf{Y})$] in a database.
    end for
end procedure
```

Fig. 5.11 Pseudocode of the training phase for the proposed LAPSE method for OLED power reduction and image enhancement.

given the relatively limited search space and that training is performed only once, offline.

Linear fitting of Transformation Coefficients

The training algorithm of LAPSE yields the optimal parameters of T for a given image, according to the cost function (5.21). However, performing this procedure based on exhaustive search at runtime is clearly unfeasible.

To obtain an adaptive transformation that is flexible (i.e. image-adaptive) but simple enough for online usage, coefficients of T have to be put in relation with elementary features of the image being processed. Since the transformation affects the Y component of pixels, the features that we select are also luminance-related. The two simplest choices are the average *brightness* and *contrast*, measured by the luminance mean μ and standard deviation σ respectively [31]. The latter can be computed as shown in (5.23) and (5.22). More precisely, in order to simplify the hardware implementation of the transformation, as explained in Section 5.4.3, we substitute σ with σ^2.

Similarly to what we had done for the LABS technique, we use a *linear regression* model to put in relation $\mathbf{a}$ with μ and σ^2.

This model is built using the outputs from the previous training procedure, using ordinary least squares. Since we have two free parameters a_1 and a_2, the model coefficients are $\mathbf{P} \in \mathbb{R}^{2x2}$ and $\mathbf{q} \in \mathbb{R}^{2x1}$, and identify a regression *plane*:

$$\mathbf{a^T} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \mathbf{P} \cdot \begin{bmatrix} \mu \\ \sigma^2 \end{bmatrix} + \mathbf{q} \tag{5.24}$$

$\mathbf{P}$ and $\mathbf{q}$ are then used to determine the transformation parameters for new images.

The values of a_1 and a_2 used as training data, and consequently the values of $\mathbf{P}$ and $\mathbf{q}$, depend on the $MSSIM_{\text{min}}$ constraint set in the algorithm of Figure 5.11. Therefore, when estimating a_1 and a_2 for a new image, the regression model will output parameters that, apart from the estimation error, cause the transformation T to meet the same quality constraint.

Repeating training with different *MSSIM* constraints yields different fitting coefficients. For example, if training is performed setting first $MSSIM_{\text{min}} = 0.95$ and then $MSSIM_{\text{min}} = 0.90$, it will produce coefficients $(\mathbf{P_{0.95}}, \mathbf{q_{0.95}})$ and $(\mathbf{P_{0.90}}, \mathbf{q_{0.90}})$ respectively. When these are used for new images, the latter will produce larger power reductions in the display, but will also cause more visible image alterations. Therefore, fitting coefficients obtained with different *MSSIM* constraints allow to explore the power versus quality Pareto frontier.

Similarly to the case of LABS, the choice of a linear model has been dictated by its low evaluation complexity. In fact, while training is performed offline, the evaluation of the model according to (5.24) must be executed online, in real time. Therefore, the complexity of this regression should be kept as low as possible. In our experiment, we will show that this simple model is sufficient to obtain transformations that are very similar to the optimal ones, from both visual inspection and quantitative analysis.

5.4.3 Online Phase

Having determined the fitting coefficients that link the basic features of an image (μ, σ^2) to the optimal values of $\mathbf{a}$, we can adapt the pixel-by-pixel transformation expressed by (5.17) to the content of the OLED display at runtime. A flow diagram summarizing the online part of LAPSE is shown in Figure 5.12. Globally, these operations take an input image I_i in RGB format and transform it into an energy-efficient output image I_o, in the same format. Internally, the flow can be subdivided into three main phases. The complexity of each phase in terms of number of required operations, with respect to the size of the OLED panel, is reported in the figure.

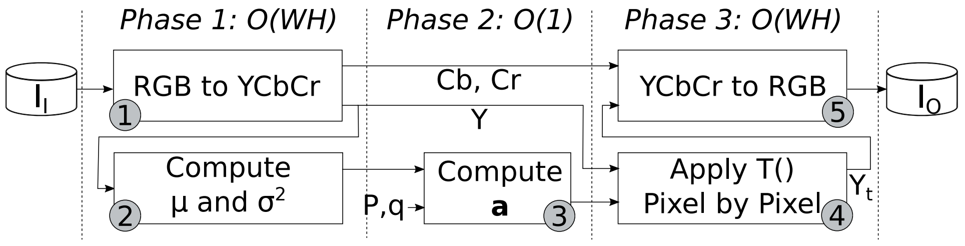


Fig. 5.12 Flow diagram for the online part of the proposed LAPSE method for OLED power reduction and image enhancement.

The first part, labeled *phase 1*, consists in the conversion from RGB to YCbCr space and in the computation of the average luminance and contrast of the input image. Both these tasks need to execute operations on every pixel, hence their complexity is $O(WH)$, i.e. linear with respect to the size of the panel. Notice, however, that the two tasks can be completely *overlapped*: as soon as one pixel has been converted to YCbCr space, its luminance can immediately be used for the calculation of μ and σ^2.

The section labeled *phase 2* consists in the evaluation of the fitting model. It takes as input the fitting coefficients $\mathbf{P}$ and $\mathbf{q}$, as well as the values of μ and σ^2, computed in the previous phase. Since the number of transformation parameters is 2 (a_1 and a_2) regardless of the size of the panel, the complexity of this phase is constant.

Phase 3 includes the actual application of the transformation T to the luminance component of each pixel, and the conversion of the transformed image to RGB. As for *phase 1* complexity is linear, since both tasks must be executed once per pixel. Again, the two tasks can be overlapped, applying the color space conversion to a pixel as soon as it has been transformed.

Overall, the scheme of Figure 5.12 is significantly simpler than those proposed in previous works [148, 151, 149, 150]. First, as detailed in the following, the only operations involved in the online transformation, apart from some basic control flow,

Phase	Add/Sub	Mul
RGB to YCbCr	$6WH$	$9WH$
Compute μ and σ^2	$2WH+1$	$WH+3$
Compute a_1 and a_2	4	2
Apply T()	$4WH$	$5WH$
YCbCr to RGB	$6WH$	$4WH$

Table 5.3 Number of additions/subtractions and multiplications required in each phase of the online computations that compose the proposed LAPSE method.

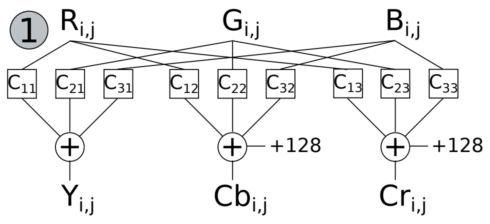
are additions, subtractions and multiplications. Second, very few of such operations are required to process each pixel of the image, and the overall complexity grows linearly with the OLED panel size. Table 5.3 shows a breakdown of the number of basic operations required in each block of Figure 5.12.

One important aspect of this flow is the possibility of changing coefficients **P** and **q** at runtime. According to the analysis of Section 5.4.2, this permits runtime switching of the transformation "quality mode", i.e. the amount of allowed image alteration. For example, this change can be triggered by external conditions (e.g. battery state of charge, ambient illumination), or by a user-driven quality setting. Conversely, previous methods for concurrent power reduction and contrast enhancement do not allow to *directly* set a maximum alteration constraint on their transformations. In [148, 151, 149], different degrees of alteration can be obtained acting on algorithm parameters, but this requires non-trivial tuning. Other works, such as [150], propose transformations that reach a target power saving, without taking alteration into account.

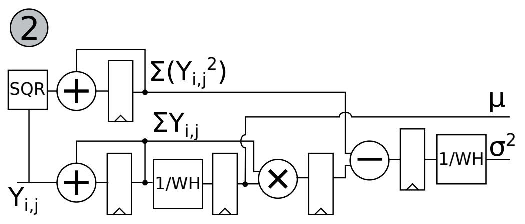
Hardware Implementation Details

While a software implementation of Figure 5.12 is relatively trivial, the online part of LAPSE also lends itself to hardware acceleration, for better performance and energy efficiency. One possible hardware implementation of each of the blocks that compose Figure 5.12 is shown in Figures 5.13.

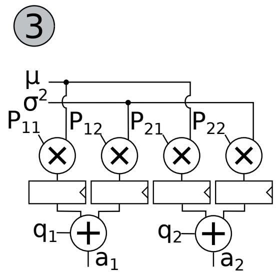
All circuits work on fixed-point data, and only use addition, multiplication and shift operations. The choice of σ^2 in place of σ as a feature eliminates the need of

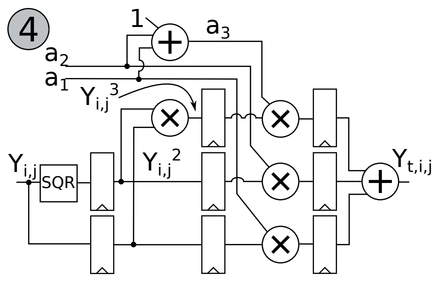


(a) RGB to YCbCr conversion

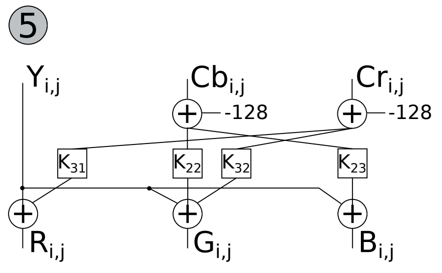


(b) Computation of μ and σ^2





(c) Fitting of a_1 and a_2 (d) Application of transformation T



(e) YCbCr to RGB conversion

Fig. 5.13 Hardware implementation of the different blocks that compose the online part of the proposed LAPSE method.

computing a square root. Moreover, the divisions required to compute μ and σ^2 can be implemented as multiplications via inversion. In fact, the denominator $W \cdot H$ is the size of the panel, which is constant.

The diagram of Figure 5.13a implements equation (5.3a) to convert each image pixel from RGB to YCbCr. Square blocks represent constant multiplications, which do not require full hardware multipliers.

Figure 5.13b shows the circuit used to compute μ and σ^2. The block labeled *SQR* represent the squaring operation (i.e. $Y_{i,j}^2$). Mean luminance is obtained using

(5.23), whereas for variance we use an alternative expression with respect to (5.22):

$$\sigma^2 = \frac{(\sum_{i=0,j=0}^{W,H} Y_{i,j}^2) - \mu \cdot \sum_{i=0,j=0}^{W,H} Y_{i,j}}{WH} \tag{5.25}$$

The advantage of this factorization with respect to the previous is that μ is not included in the summation. Hence, $\sum Y_{i,j}^2$ can be computed in parallel with $\sum Y_{i,j}$, and both μ and σ^2 can be calculated with a *single scan* of all pixels. Cancellation errors typical of floating point implementations of this equation do not occur in fixed point.

Figure 5.13c shows the hardware for evaluating the fitting model. This is the circuit with the largest power consumption, since it includes four multipliers, although with small bit-widths (12x12-bit according to the precision analysis described in the following). However, it is only active for two clock cycles per image. Thus, low-power optimization techniques such as clock and/or power gating can be used to dramatically reduce its energy impact.

The circuit in Figure 5.13d implements the actual pixel luminance transformation of (5.17). To this end, the parameter a_3 is first extracted from a_1 and a_2 using (5.16). In this operation, the constant 255 can be replaced with $256 = 2^8$ without a significant impact on the output error. Using a power of two allows to replace multiplication with proper wiring, with zero hardware cost. Thus, the computation of a_3 reduces to a simple addition. This circuit is pipelined, so that one transformed pixel is produced at each clock cycle. A non-pipelined implementation can be easily designed to obtain even smaller hardware, loosing in throughput.

Finally, Figure 5.13e shows the hardware for converting the transformed pixels back to RGB color-space, according to equation (5.3b). The circuit is similar to that of Figure 5.13a, with the difference that some constant multiplications can be replaced by proper wirings, thanks to the fact that the corresponding elements of matrix **K** are either 1 or 0 [206].

Given that both inputs and final outputs are 8-bit values, most intermediate operations do not need large bit-widths to obtain sufficient accuracy. For example, the constant factors in the multiplications of Figure 5.13a and 5.13e must be represented on 9-bit and 10-bit two's complement respectively, with proper ranges, to generate color-space conversions that are accurate up to one intensity unit, with respect to a double precision floating point reference. With similar considerations, it is possible to devise the bit-widths of all other operations. For the fitting model evaluation, we select bit-widths that make the hardware arithmetic error negligible with respect to the fitting *RMSE* on a set of test images [215, 208].

Since both RGB and JPEG YCbCr formats represent pixels on 24-bits, no additional memory is required to implement the transformation, except for the intermediate registers shown in the figures. In fact, pixels can be overwritten *in place* after color-space conversions and luminance transformations. The memory used for this purpose can be either the main system memory or the display frame buffer depending on the global system architecture.

It should be remarked that, as for the case of LABS, the ones presented are just some of the possible implementations for the tasks of Figure 5.12. For instance, most of the operations involved in the transformation lend themselves to parallelization. Hence, vectorized versions of the circuits can be designed, in order to improve throughput, at the cost of an equivalent increase in area and power. On the other hand, architectures that favor hardware reuse (e.g., non-pipelined) are also possible, in order to reduce the number of required adders and multipliers. In general, the optimal architecture depends on system-specific requirements.

5.4.4 Universal LAPSE

An interesting variant of the method described in Sections 5.4.1-5.4.3 consists in maintaining the cubic form of T, but making its parameters *independent* of the considered image. We call this variant *universal* LAPSE. In practice, rather than computing the optimal image-dependent parameters of the transformation through fitting, two constant values are assigned to a_1 and a_2 throughout the online phase. These constants are chosen as the *averages* of the a_1 and a_2 values obtained in the training algorithm of Section 5.4.2.

The advantage of the universal solution is a reduction in the complexity of the online phase. Specifically, the computation of μ and σ, as well as the fitting of a_1 and a_2 (i.e. tasks labeled 2 and 3 in Figure 5.12) are not necessary. The obvious drawback is that the transformation is not adapted to the considered image. As a consequence, its effect on visual quality, and especially on the *MSSIM* between input and output images, will vary significantly from one image to the other. The choice between universal and adaptive LAPSE identifies a tradeoff between quality of results and overhead impact, which will be analyzed further in our experimental results.

5.4.5 Experimental Results

Setup

In the experiments presented in this section, we have once again used the model of equation (5.1) to estimate the power consumption of an OLED panel. Model coefficients have been set to the same values used for LABS and in [148], i.e. $(\gamma, w_0, w_r, w_g, w_b) = (2.2, 0, 70, 115, 154)$. The *MSSIM* has been used to measure image similarity.

We have tested LAPSE on two data sets of still images: LIVE (29 images) [215] and BSDS (800+ images) [208], which had already been considered for LABS. For tests on videos we have also considered the same sources used in our experiments on LABS, i.e. the *Open Video Project* [210] and the *Derf's Test Media Collection* [211].

We have implemented the offline part of LAPSE in Python 3.5. In the algorithm of Figure 5.11, we set w_p to:

$$w_p = \frac{255}{W \cdot H[w_0 + (w_r + w_g + w_b)255^{\gamma}]} \tag{5.26}$$

The denominator of this equation is the maximum power consumed by a panel of size $W \cdot H$, corresponding to a totally white image. Thus, using this value for w_p normalizes power to the $[0 : 255]$ range, and makes it comparable to the values assumed by the luminance standard deviation $\sigma \in [0 : \frac{255}{2}]$. This guarantees a good balance between power reduction and contrast enhancement during training. The search space for $\mathbf{a_{opt}}$, defined by equations (5.20a) and (5.20b) has been divided into a regular grid of 40.000 points for exhaustive search. A smaller discretization of the search space does not produce a significant difference in the power saving and visual quality achieved during training.

The online part of the transformation has been implemented in both software and hardware. For the former we have written a single-thread program in C, which we have compiled with GCC version 6.3.0 for a high-end x86_64 platform (Intel Xeon

E3-12, 8MB L3, Linux Kernel v. 2.6). We have measured execution times with the Linux real time clock library, averaging the results of 1000 runs.

The hardware version has been designed at RTL using VHDL. It has been then synthesized with Synopsys Design Compiler v2016.03, targeting a 45nm CMOS standard-cell library from ST Microelectronics. The clock frequency for synthesis has been set to 1GHz for all blocks. The execution of the synthesized hardware has been evaluated by means of simulations in Mentor Graphics QuestaSim v10.6. Power consumption has been estimated in Synopsys PrimeTime Suite v2016.6, using annotated switching activity from simulations. Since the number of operations required by the online transformation is independent on data, the hardware time is constant for a given panel size.

Fitting and universal transformation

In a first experiment, we have evaluated the performance of our selected linear regression model to estimate the parameters of T at runtime. For a more complete evaluation, we have also compared it with polynomial models of different order. For this test we have used BSDS images [208] (similarly to the regression experiment for LABS), due to the fact that they are already split into training and test sets.

We have run the algorithm of Figure 5.11 on all BSDS images, and we have used the values of μ, σ^2, and $\mathbf{a}$ from the training subset to build the different models. We have then evaluated the goodness of each model on test images. As accuracy metric, we have used the (normalized) $RMSE$ of a_1 and a_2, and that of the output $MSSIM$. The latter has been computed transforming all test images with fitted a_1 and a_2 and evaluating the $MSSIM$ difference with respect to the image obtained with ideal parameters, directly generated by the algorithm of Figure 5.11.

The results of this evaluation for the constraint $MSSIM_{\min} = 0.80$ are shown in Figure 5.14, where the x axis shows the order of the polynomial. The 0 point corresponds to a *constant* polynomial, i.e., a model that estimates a_1 and a_2 as constant values, using the average of the training set. Evidently, this corresponds to *universal LAPSE* described in Section 5.4.4.

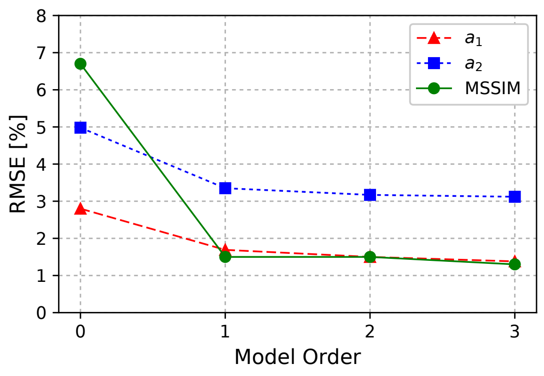


Fig. 5.14 Fitting *RMSE* for polynomial regression models of different order applied within our proposed LAPSE method to estimate transformation parameters from simple image features.

These results confirm that a linear model (order 1) is a good choice [3]. In fact, with respect to the constant approximation, all errors drop significantly. In particular, the *RMSE* on *MSSIM* decreases from 6.7% to 1.5%. On the other hand, increasing the model order beyond 1, the error on *MSSIM* remains approximately constant (1.3% for a 3rd order model). Thus, the linear solution represents a knee-point, and is a good tradeoff between evaluation complexity and accuracy.

Moreover, the low errors obtained for the different models also demonstrate that the mean and variance of the image luminance are sufficient features to determine close-to-optimal parameters for the transformation T of LAPSE. Although more complex features could provide even better fitting results, they would also require more complex calculations to be extracted, hence increasing the transformation overheads. Finally, notice that the universal solution could be still acceptable, depending on the system requirements. Obviously, it will produce images whose similarity with the input is sometimes significantly lower than the imposed constraint. However, this loss in accuracy must be traded off with the advantages in terms of implementation complexity, detailed in the following.

Subjective evaluation

Although simpler than previous solutions for concurrent power reduction and image enhancement in OLEDs, LAPSE still introduces some overheads for implementing

[3]The order of the fitting model (linear) should not be confused with the order of the polynomial used for transforming pixels (cubic).

the online part of the transformation. To justify these overheads, the subjective quality of the transformed images must be superior to that achieved by brightness scaling, e.g. using LABS or a similar technique. This would prove that LAPSE actually achieves the desired image enhancement.

To determine whether this is the case, we have performed a subjective evaluation test, using the entire Live data set, together with the first 50 images of the *BSDS* training set. The number of images has been limited in order to gather a sufficient number of votes per image. We have compared LAPSE with brightness scaling under the same power consumption conditions. To do so, we have first transformed all benchmark images with LAPSE, setting the minimum *MSSIM* constraint to 0.80. We have then computed the power saving obtained for each image. Finally, we have applied luminance scaling to the input images, setting the scaling factor k to a value that produces the *same* power saving as LAPSE. This value can be obtained numerically using a bisection method, as explained in Section 5.3.5.

The generated pairs of images have been shown to a pool of 169 subjects, not expert in image or video processing. Each subject has been asked to evaluate 10 pairs of images, and for each pair, select the version that he or she preferred. Results are shown in Table 5.4.

Data	**Images**	**LAPSE**	**Scaling**	**Draw**
Live	29	24 (82.8%)	5 (17.2%)	0 (0.0%)
BSDS	50	38 (76.0%)	11 (22.0%)	1 (2.0%)
Total	79	62 (78.5%)	16 (20.3%)	1 (1.3%)

Table 5.4 Results of a subjective evaluation test performed to compare our proposed LAPSE method with brightness scaling in same power conditions. Number and percentage of preferences for each data set.

Columns “LAPSE” and “Scaling” report the number of images for which the proposed approach was preferred to brightness scaling by the voters, and vice versa. The last column report draws. This result confirms that, for the large majority of images, LAPSE produces subjectively better outputs compared to luminance scaling, at the same power level.

Comparison with PCCE

PCCE, presented in [148], is one of the most popular image transformations for OLED power reduction and contrast enhancement. Although PCCE is significantly more complex than LAPSE, it achieves a similar objective, hence we have chosen this technique as state-of-the-art reference for comparison.

Differently from the previous comparison with brightness scaling, in this case it is not possible to compare the two methods for equal power consumption. In fact, both PCCE and LAPSE do not allow to fix the desired power saving. Rather, PCCE performs an unconstrained optimization of power and contrast, only controlled through the parameter β, whereas in our solution, power is minimized under a *MSSIM* constraint. Thus, we have compared the two methods in same similarity (*MSSIM*) conditions, and evaluated whether they obtain similar power savings and image enhancement scores.

To do so, we have applied PCCE to all the images in the BSDS and Live data sets. We have then computed the *MSSIM* between each original and transformed image, and set that value as a constraint in our framework. As metric of image enhancement we have used the popular EME score [216].

Table 5.5 shows the aggregate results obtained in this analysis. For both power and EME, the table reports average values and standard deviation intervals. Under the same *MSSIM* conditions, the two methods achieve almost identical power savings, although PCCE has less variation over different images. In terms of quantitative enhancement, PCCE is slightly better than our approach, due to its greater flexibility in the shape of the pixel transformation T. It must be remarked, however, that the primary goal of our method not to improve PCCE performance, but rather to achieve comparable results at a much lower overhead cost. Thus, results should be read taking into account the difference in complexity among the two methods.

A more detailed comparison for six example images is shown in Figure 5.15 and Table 5.6. In addition to the original images and to the PCCE outputs, two sets of LAPSE outputs are reported. The first has been obtained with *ideal* parameters, i.e. running the algorithm of Figure 5.11 directly on the target image, setting the *MSSIM* of the PCCE output as constraint. The second refers to the transformation with *fitted* parameters, i.e. computed according to the model of equation (5.24). These images are affected by fitting error, and are the actual outputs of LAPSE at runtime. Fitting

Data	Power Saving [%]		EME	
	LAPSE	**PCCE**	**LAPSE**	**PCCE**
Live	47.5 ± 15.9	49.5 ± 10.9	24 ± 16.1	28.7 ± 15.1
BSDS	44.2 ± 20.1	44.6 ± 10.6	19.7 ± 14.4	25.9 ± 12.8

Table 5.5 Comparison between the proposed LAPSE method and a state-of-the-art technique for concurrent power reduction and image enhancement in OLED displays: aggregate results.

coefficients for each image have been determined performing a full training using the *MSSIM* of PCCE as constraint, after removing the image from the training set.

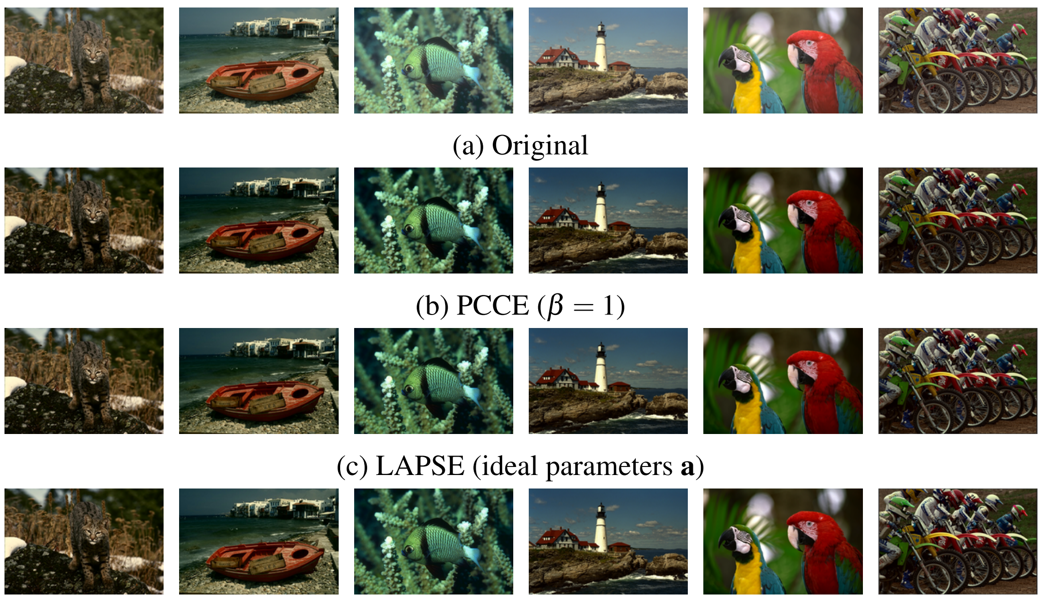

(a) Original
(b) PCCE ($\beta = 1$)
(c) LAPSE (ideal parameters **a**)
(d) LAPSE (fitted parameters **a**)

Fig. 5.15 Comparison between the proposed LAPSE method and a state-of-the-art technique for concurrent power reduction and image enhancement in OLED displays: example for six images.

These examples highlight the similarity between the outputs of our solution and those of PCCE. Looking at the images, it is very hard to notice any difference in terms of quality and level of detail.

To better evaluate the similarity of the two methods, Figure 5.16 shows the luminance transformation curves obtained by PCCE and by the two LAPSE variants for the six images of Figure 5.15. These graphs show that the luminance transformations

Data	**Power Saving [%]**			**EME**			***MSSIM***		
	PCCE	**LAPSE Ideal**	**LAPSE Fitted**	**PCCE**	**LAPSE Ideal**	**LAPSE Fitted**	**PCCE**	**LAPSE Ideal**	**LAPSE Fitted**
Cat	37.1	32.1	36.4	28.5	23.9	24.3	0.812	0.842	0.825
Boat	47.1	51.1	44.0	29.3	25.3	23.0	0.794	0.795	0.805
Fish	52.7	59.8	58.1	12.6	11.3	10.3	0.755	0.756	0.791
Lighthouse	61.0	62.2	60.3	17.2	15.6	14.7	0.764	0.768	0.782
Parrots	51.8	55.1	51.2	14.4	5.9	5.6	0.724	0.725	0.772
Bikes	49.3	42.0	36.4	31.6	33.7	36.1	0.778	0.778	0.788

Table 5.6 Comparison between the proposed LAPSE method and a state-of-the-art technique for concurrent power reduction and image enhancement in OLED displays: numerical results for six images.

obtained by LAPSE are very similar to the ones of PCCE. The occasional differences are due to the higher number of degrees of freedom available to PCCE for shaping the luminance intensity mapping, but they are typically very small.

Moreover, the examples also show that the fitting error does not worsen the performance of our transformation significantly, neither in terms of visual quality nor quantitative metrics. This is demonstrated both by the similar values of saving and EME in Table 5.6, and by the fact that the fitted luminance transformations in Figure 5.16 are almost exactly superimposed to the ideal ones.

Quality versus power tradeoff

LAPSE directly takes a maximum alteration (minimum $MSSIM$) constraint for its training phase. This allows to easily change "quality modes" at runtime, as explained in Section 5.4.3. Notice that none of the previous solutions for concurrent power reduction and contrast enhancement allows to set a similar constraint; some receive as input a target power consumption (e.g. [150]), while some others have parameters specifying the balance among power reduction and contrast increase (e.g. [148]). However, none of them allows to directly limit the alteration on the output image.

To show this novel aspect of our framework, we have trained LAPSE on BSDS images, setting three different $MSSIM$ constraints: 0.95, 0.85 and 0.75. Then, we have calculated regression coefficients for each of these conditions and we have used

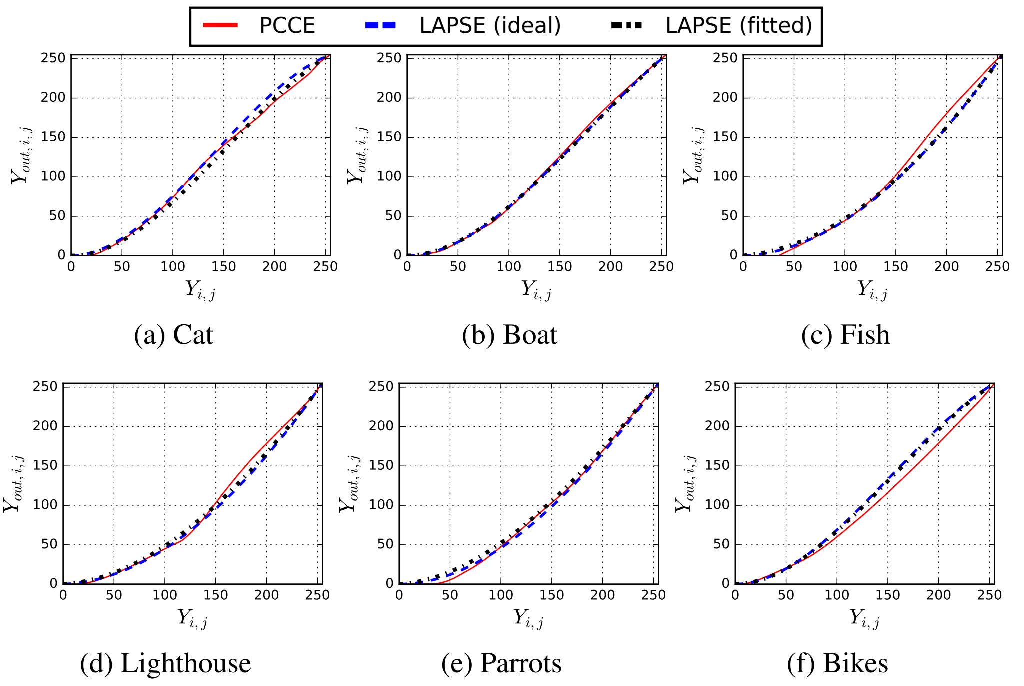


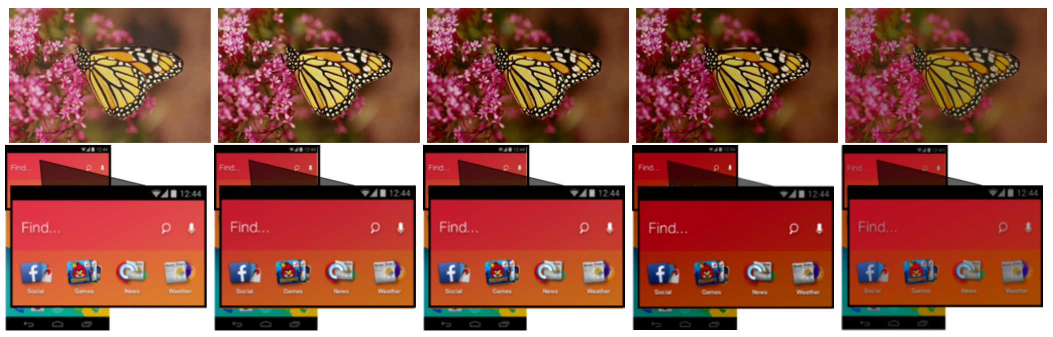
Fig. 5.16 Comparison between the proposed LAPSE method and a state-of-the-art technique for concurrent power reduction and image enhancement in OLED displays: transformation curves for six images.

them to transform two example images, not included in the training set. Results are shown in Figure 5.17 and Table 5.7.

Target MSSIM	**0.95**		**0.85**		**0.75**	
Image	**Saving [%]**	**Out MSSIM**	**Saving [%]**	**Out MSSIM**	**Saving [%]**	**Out MSSIM**
Butterfly	24.0	0.961	41.3	0.881	57.9	0.765
Android	27.7	0.977	43.7	0.849	54.4	0.777

Table 5.7 Examples of power versus quality tradeoff offered by the proposed LAPSE method, detailed results.

The last column of the figure shows the output of a simple brightness scaling, with a scaling factor that guarantees the same power saving as LAPSE in the case $MSSIM = 0.75$. Notice how the contrast and details (e.g. the butterfly wings) are highlighted better by LAPSE.



(a) Original (b) *MSSIM*=0.95 (c) *MSSIM*=0.85 (d) *MSSIM*=0.75 (e) Scaling

Fig. 5.17 Examples of the power versus quality tradeoff offered by the proposed LAPSE method.

For completeness, Table 5.8 shows the aggregate power saving and output *MSSIM* results when the experiment described above is performed on the entire Live data set and on the BSDS test set (fitting coefficients have been obtained considering the BSDS training set only). This table shows that our method obtains relevant savings even for *MSSIM* values close to 1, and that the fitting model is able to match the target *MSSIM* very accurately, on average.

Target	**MSSIM = 0.95**	**MSSIM = 0.85**	**MSSIM = 0.75**
Data	**Saving [%]**	**Saving [%]**	**Saving [%]**
Live	24.87 ± 5.91	39.38 ± 8.73	49.37 ± 10.41
BSDS (Test)	23.35 ± 7.77	37.51 ± 11.88	47.21 ± 13.84
Data	**Out MSSIM**	**Out MSSIM**	**Out MSSIM**
Live	0.959 ± 0.015	0.855 ± 0.026	0.773 ± 0.038
BSDS (Test)	0.950 ± 0.019	0.855 ± 0.042	0.776 ± 0.051

Table 5.8 Examples of power versus quality tradeoff offered by the proposed LAPSE method, aggregate results.

The Android screenshot example in Figure 5.17 highlights the generality of our approach, which yields good results also when applied to GUIs, despite not being expressively designed for them. In particular, the icons and text are much more contrasted and visible in Figure 5.17d than in Figure 5.17e, for identical power savings.

Performance on video sequences

The low overheads of LAPSE compared to previous solutions allow real-time application to video sequences. Some examples of transformed videos, put side by side with the original input have been uploaded to [217] for visual inspection. The sequences have been transformed using a fitted parameter vector **a**, trained under a constraint of $MSSIM_{min} = 0.80$. These examples show that, although **a** is adapted on a frame by frame basis, this does not generate visible flickering artifacts in the videos. The motivation is similar to the case of LABS: the parameters in **a** are function of the luminance mean and variance of each frame, and frames belonging to the same scene tend to have similar luminance. Thus, abrupt changes of the transformation coefficients only occur in correspondence of scene changes, and do not impact user experience.

Figure 5.18 shows the trend over time of the power saving (normalized to $[0:1]$, where 0 means no saving) and $MSSIM$ obtained by LAPSE on some of these video sequences. Each plot also reports the average $MSSIM$ over the entire sequence, as well as the total power saving i.e., $1 - \frac{P_{in,tot}}{P_{out,tot}}$ where $P_{in,tot}$ and $P_{out,tot}$ are the total power consumptions over all frames, for the original and transformed videos.

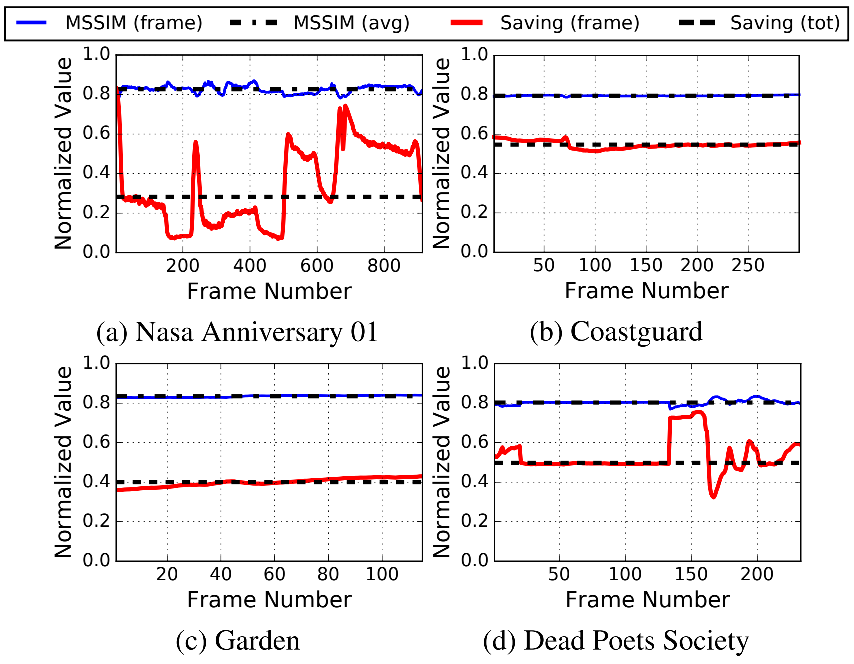


Fig. 5.18 Power saving and $MSSIM$ variation over time for some video sequences transformed with the proposed LAPSE method.

Notice how the *MSSIM* stays very close to the target value (0.80) throughout the videos. Instantaneous power saving, instead, can vary significantly over time, as in the case of the first and last videos, which are both characterized by varying luminance. These examples clearly show the effectiveness of LAPSE in limiting the amount of image alteration allowed, and automatically tuning the amount of power reduction accordingly.

Software implementation results

The software execution time of LAPSE, under the conditions described at the beginning of the experimental results sections is reported in Table 5.9. Two different image sizes are considered, and each cell of the table reports the average time over 1000 runs and the corresponding standard deviation. For comparison, we have also implemented the PCCE algorithm [148] and the *Noniterative PCCE* (NIPCCE) of [150] in C language and computed their execution times on the same platform. For PCCE, we set the $\beta = 1$, whereas for NIPCCE, we used 8x8 local windows and we set $TPCR = 0.7$, $w_{cl} = 0.05$, and $\gamma = 2.2$.

All three algorithms have been profiled in sections for a more detailed analysis. The acronyms used for the sections of PCCE and NIPCCE are as follows: HIST = luminance histogram computation, LHM = log-based histogram modification, PCCE = main optimization loop of PCCE, GA = global attribute computation, LA = local attribute computation, LT = local transformations computation, BI = bilinear interpolation, ADJ = final image adjustment.

The bottom right section of the table reports the execution time of color-space conversions (RGB to YCbCr and vice versa) alone.

While implementing the three algorithms, we have minimized the number of image scans grouping those operations that could be performed in the same loop (e.g. RGB to YCbCr conversion, and μ and σ^2 computation in LAPSE). Since these are single-thread programs, operations are still executed sequentially. However, overheads due to control operations and memory accesses are reduced. Moreover, we have not made use of any advanced image processing or mathematical library, nor of GPU acceleration. This choice has been made in order to build implementations that (although tested on a high-end processor) could be easily ported to less powerful

LAPSE			**PCCE**		
Task	**512x512 [ms]**	**1280x1280 [ms]**	**Task**	**512x512 [ms]**	**1280x1280 [ms]**
RGB-YCbCr and μ, σ^2	2.96 ± 0.35	15.19 ± 0.88	RGB-YCbCr and HIST	2.61 ± 0.35	13.68 ± 0.94
Compute **a**	$(3.23 \pm 5.52)10^{-4}$	$(5.98 \pm 1.56)10^{-4}$	LHM and PCCE	13.3 ± 0.72	13.6 ± 0.76
$T()$ and YCbCr-RGB	3.53 ± 0.32	19.49 ± 1.05	$T()$ and YCbCr-RGB	3.36 ± 0.31	13.62 ± 0.76
Total	6.49 ± 0.49	34.68 ± 1.62	**Total**	19.29 ± 1.07	45.01 ± 2.48

NIPCCE			**Color Conversions Only**		
Task	**512x512 [ms]**	**1280x1280 [ms]**	**Task**	**512x512 [ms]**	**1280x1280 [ms]**
RGB-YCbCr and GA	3.01 ± 0.33	13.96 ± 0.75	RGB-YCbCr	2.31 ± 0.37	12.75 ± 0.95
LT and LA	5.16 ± 0.29	31.67 ± 1.32	YCbCr-RGB	2.34 ± 0.21	13.94 ± 1.17
BI	4.58 ± 0.25	30.78 ± 1.11	**Total**	4.67 ± 0.48	26.69 ± 1.95
ADJ and YCbCr-RGB	2.59 ± 0.13	15.48 ± 0.72			
Total	15.35 ± 0.89	91.89 ± 3.15			

Table 5.9 Software execution time of the proposed LAPSE method and of two state-of-the-art algorithms for power reduction and image enhancement in OLEDs.

embedded devices, i.e. the final targets for LAPSE, which might not have these features available.

As shown in Table 5.9 the execution of LAPSE is dominated by colors-space conversions, regardless of the size of the considered image. The largest overhead is the application of T, which increases the duration of *phase 3*, compared to the pure YCbCr to RGB conversion. Overall, the transformation requires ≈ 2 ms more with respect to color conversions for a 512x512 image (6.49 ms vs 4.67 ms), and ≈ 8 ms more for a 1280x1280 image (34.68 ms vs 26.69 ms). Notice that the execution time of LAPSE is independent of the considered image, and only changes in response of variations in external system conditions (e.g. CPU load).

On the contrary, the second phase of PCCE requires more than 10 ms, independently on the size of the image (it operates on histogram bins). Moreover, while independent on size, the duration of this phase is highly dependent on image content.

Results in the table refer to two scaled versions of *Lena*, but the execution time could be even higher for other images.

Considering total execution times, LAPSE is approximately 3x faster than PCCE for 512x512 images, and 1.3x for 1280x1280 images. Taking into account only the additional time with respect to color space conversions, performance ratios become 8x and 2.3x respectively.

Noniterative PCCE [150] does not use the iterative optimization loop present in standard PCCE. However, this algorithm still includes complex operations (e.g. bilinear interpolation, local image filtering for structural sensitivity evaluation, and internal power consumption estimation). Moreover, since NIPCCE applies different transformations to different local windows of the image, its complexity increases significantly for large images. This is confirmed by the results in Table 5.9, which show that NIPCCE is faster than PCCE for 512x512 images, but significantly slower for 1280x1280 images. In both cases, NIPCCE remains significantly slower than LAPSE (2.3x times and 2.65x respectively). When excluding color space conversions, these speed-ups become 5.87x and 8.16x. Therefore, although this method improves the output image quality with respect to PCCE, as shown in [150], it is still not usable in a real-time setting.

Importantly, for 1280x1280 pixels images, even just the execution of the two color-space conversions is too long for real-time application, considering the typical constraints imposed by OLED refresh rates. An implementation that achieves the required performance could be obtained resorting to parallelism (multi-thread or GPU). However, this would require a high-end computing device exclusively dedicated to the transformation of images, with very large power and cost overheads.

This confirms the need for hardware acceleration, in order to obtain the desired benefits from this kind of image transformation. Software results are still useful to compare LAPSE with PCCE and NIPCCE, as no hardware acceleration for these methods has been proposed.

Hardware implementation results

The main advantage of the proposed polynomial transformation is that it is amenable to hardware implementation. To demonstrate it, we have implemented the architectures of Figure 5.13a-5.13e, under the conditions described at the beginning of this

Block	**Area [mm^2]**	**Power [mW]**	**Latency / Image [ms]**		**Energy / Image [J]**		
			512x512	**1280x1280**	**512x512**	**1280x1280**	**Breakdown [%]**
RGB to YCbCr	0.013	2.12	0.26	1.64	$5.55 \cdot 10^{-7}$	$3.47 \cdot 10^{-6}$	26.1
Compute μ and σ^2	0.029	1.81			$4.74 \cdot 10^{-7}$	$2.96 \cdot 10^{-6}$	22.3
Compute **a**	0.042	5.28	$2 \cdot 10^{-6}$	$2 \cdot 10^{-6}$	$1.06 \cdot 10^{-11}$	$1.06 \cdot 10^{-11}$	≈ 0
Apply $T()$	0.044	3.10	0.26	1.64	$8.13 \cdot 10^{-7}$	$5.08 \cdot 10^{-6}$	38.2
YCbCr to RGB	0.008	1.09			$2.85 \cdot 10^{-7}$	$1.78 \cdot 10^{-6}$	13.4
Total	0.138	4.06	0.52	3.28	$2.13 \cdot 10^{-6}$	$13.29 \cdot 10^{-6}$	100

Table 5.10 Hardware implementation results for the proposed LAPSE method.

section. A breakdown of the main figures of merit of each hardware component of the online transformation is reported in Table 5.10.

Notice that, differently from software, hardware blocks that belong to the same phase execute in a fully parallel fashion. The energy consumption is assumed to be zero when a block is not being used (e.g., the *RGB to YCbCr* block does not consume during phases 2 and 3). This simplification is quite accurate assuming power and/or clock gating are used. The power consumption reported in the last row is the average during the transformation of an image (i.e., the total energy divided by the execution time).

Hardware execution times are almost one order of magnitude faster than software. Even for the higher resolution image, execution completes in $\approx 3.3ms$. This result is more than sufficient for a real time implementation of the transformation on a HD OLED panel. Nonetheless, the total energy consumption is still extremely low. For sake of comparison, in the experiments for LABS we have mentioned that a small 240x320 pixels AMOLED panel consumes approximately $3.9 \cdot 10^{-3}$ J per frame [213]. Therefore, even though the panel is smaller than the images considered in this experiment, its energy consumption is still three orders of magnitude larger than that required to implement LAPSE. From this point of view, LAPSE and LABS achieve similar results.

Table 5.10 also helps in assessing the advantages of universal LAPSE (Section 5.4.4). In this variant, the computations of μ, σ^2 and **a** are not needed, and the corresponding hardware can be removed. While the latter has practically no impact on energy (being active in very few clock cycles), the former contributes to the 22.3% of the total consumption. This is reported in the last column of Table 5.10 which shows a breakdown of the energy consumption of the different blocks. Hence, universal LAPSE is expected to consume $\approx 78\%$ of the energy of the adaptive version. Due to operation overlapping, the advantage in latency is practically null. As explained above, this energy reduction comes at the cost of a worse matching of the expected *MSSIM* by the transformation. Whether this is acceptable depends on the system; adaptive LAPSE might be preferable for high-end devices, e.g. smartphones, whereas the universal solution could be interesting for low cost products like smartwatches.

5.5 Final Remarks

In this chapter, we have described two different approaches to achieve energy efficiency in OLED displays. Our two proposals tackle the problem in different ways: while LABS performs a simple brightness scaling transformation, trying to balance the alteration introduced in the image with the achievable power savings, LAPSE achieves concurrent power reduction and image enhancement under a maximum alteration constraint.

Due to these different goals, LABS and LAPSE are not comparable in a straightforward way. Nonetheless, there are common aspects and differences that are worth being analyzed in this section, to summarize the concepts introduced in the chapter.

Both methods have been designed with the goal of favoring a simple software implementation, and a straight forward conversion into hardware for acceleration. The results of Section 5.3.5 and Section 5.4.5 have shown that the hardware versions of both transformations achieve the required performance to allow real-time application even for large image sizes. Moreover, both have a negligible energy overhead compared to the consumption of a real OLED panel. These overheads are actually similar for LABS and LAPSE. However, for the former even a software implementation achieves sufficient performance for real time application on large images. LAPSE, on the contrary, *requires* hardware acceleration (or at least some form of parallelization) to be used at runtime.

In terms of output image quality, the subjective test performed in Section 5.4.5 has shown that users tend to prefer the enhanced images produced by LAPSE to simple brightness-scaled outputs such as those generated by LABS. However, it is important to remark that pure visual quality is still partially subjective (differently, for example, from metrics like *PSNR* or *MSSIM*, which however require a golden reference). The voters in the test have been asked to simply evaluate the "beauty" of images and consequently they have favored LAPSE. Nonetheless, in a different setting, the low distortion introduced by the linear transformation of LABS could be considered more desirable.

In summary, the choice between LABS and LAPSE reduces to possibility of implementing a customized hardware accelerator and to the desired type of image transformation. The former method is straight-forward, and could even be implemented on off-the-shelf devices by just modifying the operating system display drivers. The latter requires larger design and implementation costs due to custom hardware, but generates enhanced output images.

Chapter 6

Conclusions and Future Work

The energy-quality scalability paradigm is one of the most promising ways to design future generations of digital devices. Modern computing systems are increasingly being integrated into everyday objects, such as the end-nodes of the IoT. Extracting and elaborating information in these connected objects involves measuring the physical word and interacting with human beings; often, the nature of such information is statistical. In summary, there is a trend towards applications whose outputs can be provided with different "levels of quality" while still remaining useful.

This trend in applications, together with the strong need for energy efficiency, the diminishing returns offered by technology scaling, and the high energetic "cost" of computing precise results, has caused a strong interest towards EQ scalability. An EQ scalable system can work in different quality modes, and leverage quality relaxations to achieve higher energy efficiency. Quality modes are selected at runtime depending on the current task, context and input data.

In the previous chapters of this thesis, we have presented several new techniques for designing the components of an EQ scalable system. As anticipated in Chapter 1, we have focused on making our proposals mature for a real industrial application, which is an often lacking aspect in previous literature.

This has involved focusing on *generality* and *automation* aspects, as well as enforcing the compatibility of our novel ideas with existing industrial *best practices*. Moreover, it has required a careful analysis of the *overheads* related to the implementation of our proposed techniques.

A large portion of the effort that brought to this thesis has been spent in developing techniques for components that are not part of the processing “brain”of a system, starting from the observation that these are often relevant (or even dominant) contributors to the total energy consumption. By doing so, we have shown that many of the same principles applied for processing elements can be transported to other components.

Concerning processing, we have proposed two methods for designing hardware data-path operators, using either the principles of reduced-precision redundancy or those of dynamic accuracy scaling. For this second case, we have actually investigated two alternative design solutions: a more flexible approach which however requires the availability of technology-specific knobs, and an application-driven solution that does not. In these works, the compatibility with best practices has been guaranteed by the integration with commercial EDA tools, which is a fundamental aspect in all the techniques described in Chapter 3. Similarly, automation has been achieved making the proposed flows as “plug-and-play” as possible, with none or minimal user intervention involved. The overheads of each method in terms of area and energy have been assessed in detail and compared with each other. Generality has been obtained making our techniques applicable to a broad family of hardware blocks, rather than being specifically designed for one particular type of unit.

In the context of serial interconnects we have designed two EQ scalable encodings called ADE and Serial-T0. These encodings exploit the error resilience of input data, e.g. coming from sensors, as well as some simple assumptions on their input statistics, to obtain large energy reductions by introducing controlled approximations during data transmission. In this case, compatibility with best practices has been enforced by analyzing the integration of the proposed encodings with standard protocols for serial buses. Both algorithms have been designed with the goal of minimizing overheads, by using very simple procedures that lend themselves to hardware acceleration. In terms of generality, the best results have been obtained with the ADE method, which achieves relevant power savings for a wide variety of input types. While Serial-T0 is somewhat more limited, it achieves state-of-the-art savings in the transmission of image pixels; consequently, this solution is still relevant for the broad range of applications that involve cameras of different types and/or displays.

Finally, we have also proposed two different approaches for achieving energy efficiency in OLED displays, at the cost of quality losses in the displayed images. LABS performs an adaptive form of brightness scaling, optimizing the balance between power saving and image alteration for each image. LAPSE achieves concurrent power saving and image enhancement by means of an adaptive cubic transformation function. For each of these methods we have proposed fully automatic frameworks to compute the required transformation parameters, based on the offline training of simple regression models, using sets of representative images. We have targeted generality by considering generic pictures and videos, rather than focusing on specific categories such as GUIs. Most importantly, we have put a lot of effort in the reduction of the overheads associated with the two proposed image transformations, thus making them *really* applicable at runtime.

A fundamental aspect linked with generality in EQ scalable design is *quality configurability*. This is common to all the components considered in this thesis, and it consists in the possibility of dynamically changing the quality mode at runtime, as opposed to designing a component for a fixed quality target. All our proposed techniques, be it in processing, serial interconnects or displays, allow such kind of configurability. The appropriate quality is sometimes selected automatically (e.g. in LABS) while in other cases it is set by a command from a higher level of the computing stack or from the user (e.g. the maximum error threshold in ST0).

Future Perspectives

While we believe that the innovations presented in this thesis have brought to some interesting results, we are also convinced that a lot more remains to be done in the context of EQ scalable design.

Up to now, most research has only focused on ways to design the *components* (hardware or software) of an EQ scalable system. Although in this thesis we have considered other parts of the system besides processing elements, we have still only focused on one component at a time. However, as for any other type of design, a full-system perspective is fundamental for the development of real complex products. While some system-level approaches have been proposed in literature (see Section 2.5) these focus mainly on combining techniques for processing elements at multiple abstraction levels. No approach has yet really considered the *entire* system from the point of view of EQ scalability, in order to build an architecture similar to the one shown in Figure 1.3 that includes including processing, memory, peripherals and interconnects, all part of the “feedback loop”.

Such *holistic* application of the EQ scalable design paradigm brings about new issues. In fact, there is a need for a unified framework for setting the available quality knobs (often of very different nature for different components), taking into account the impact that each component has on total energy and on output quality. Moreover, the interactions among different components, which influence the respective quality of results, should also be considered.

This framework should answer questions such as: *"in response to a change in the requested output quality, e.g. due to a modification in the usage context, which EQ scalable component(s) should be tuned and how, in order to achieve the maximum possible savings?"*. This is not a trivial problem, since the application-level quality is only measured at the final outputs, often with a domain-specific metric, and the contribution of each component is generally not straight-forward to estimate. Additionally, tuning the quality-mode of one component may influence the others. For example, if a processing phase is performed at a certain quality level, the subsequent data transmission or display phases should be tuned accordingly, as an excessive effort with respect to the quality of the inputs would result in wasted energy.

To the best of our knowledge, there are no effective and general solutions to this *holistic quality management* problem. The investigation of system-level techniques for EQ scalable design and the study of possible solutions to this problem will be some of the main objects of our future work.

References

[1] Yann Lecun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[2] Davide Rossi, Igor Loi, Antonio Pullini, and Luca Benini. *Ultra-Low-Power Digital Architectures for the Internet of Things*, pages 69–93. Springer International Publishing, Cham, 2017.

[3] Robert H. Dennard, Fritz H. Gaensslen, Hwa-Nien Yu, V. Leo Rideout, Ernest Bassous, and Andre R. LeBlanc. Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5):256–268, 1974.

[4] Mark Bohr. A 30 Year Retrospective on Dennard' s MOSFET Scaling Paper. *IEEE Solid-State Circuits Newsletter*, 12(1):11–13, 2007.

[5] Massimo Alioto. Energy-quality scalable adaptive VLSI circuits and systems beyond approximate computing. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2017*, pages 127–132, 2017.

[6] ITRS. International Technology Roadmap for Semiconductors 2.0: Executive Report. Technical report, 2015.

[7] Phillip Stanley-marbell and Martin Rinard. Error-Efficient Computing Systems. 11(4):362–461, 2017.

[8] Qiang Xu, Nam Sung Kim, and T Mytkowicz. Approximate Computing: A Survey. *Design Test, IEEE*, 33(1):8–22, feb 2016.

[9] 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 ACM/EDAC/IEEE Design Automation Conference (DAC), 2013*, page 1, 2013.

[10] Andrew Tanenbaum. *Computer Networks*. Prentice Hall Professional Technical Reference, 4th edition, 2002.

[11] Gregory K. Wallace. The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, apr 1992.

[12] Vinay K. Chippa, Swagath Venkataramani, Srimat T. Chakradhar, Kaushik Roy, and Anand Raghunathan. Approximate computing: An integrated hardware approach. In *Conference Record - Asilomar Conference on Signals, Systems and Computers*, pages 111–117, nov 2013.

[13] Swagath Venkataramani, Anand Raghunathan, Jie Liu, and Mohammed Shoaib. Scalable-effort classifiers for energy-efficient machine learning. In *Proceedings of the 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), 2015*, pages 1–6, 2015.

[14] Rajamohana Hegde and Naresh R. Shanbhag. Energy-efficient signal processing via algorithmic noise-tolerance. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED), 1999*, pages 30–35, 1999.

[15] Srimat T. Chakradhar and Anand Raghunathan. Best-effort computing: Re-thinking parallel software and hardware. In *Proceedings of the 47th ACM/IEEE Design Automation Conference (DAC), 2010*, pages 865–870, 2010.

[16] Armin Alaghi and John P Hayes. Survey of Stochastic Computing. *ACM Trans. Embed. Comput. Syst.*, 12(2s):92:1—-92:19, 2013.

[17] Daniele Jahier Pagliari, Massimo Poncino, and Enrico Macii. Energy-Efficient Digital Processing via Approximate Computing. In Nicola Bombieri, Massimo Poncino, and Graziano Pravadelli, editors, *Smart Systems Integration and Simulation*, chapter 4, pages 55–89. Springer International Publishing, Cham, 2016.

[18] Muhammad Shafique, Rehan Hafiz, Semeen Rehman, Walaa El-Harouni, and Jörg Henkel. Invited - Cross-layer Approximate Computing: From Logic to Architectures. In *Proceedings of the 53rd Annual Design Automation Conference*, DAC '16, pages 99:1—-99:6, New York, NY, USA, 2016. ACM.

[19] Jie Han and M Orshansky. Approximate computing: An emerging paradigm for energy-efficient design. In *18th IEEE European Test Symposium (ETS), 2013*, pages 1–6, 2013.

[20] Qian Zhang, Feng Yuan, Rong Ye, and Qiang Xu. ApproxIt: An approximate computing framework for iterative methods. In *Proceedings of the 51st ACM/EDAC/IEEE Design Automation Conference (DAC), 2014*, pages 1–6, 2014.

[21] Enrico Macii. *Ultra low-power electronics and design*. Springer US, 2004.

[22] Doochul Shin and Sandeep K. Gupta. Approximate logic synthesis for error tolerant applications. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, pages 957–960, 2010.

[23] Andrew B Kahng and Seokhyeong Kang. Accuracy-configurable adder for approximate arithmetic designs. In *Proceedings of the 49th ACM/EDAC/IEEE Design Automation Conference (DAC), 2012*, pages 820–825, 2012.

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

[25] Jinghang Liang, Jie Han, and F Lombardi. New Metrics for the Reliability of Approximate and Probabilistic Adders. *IEEE Transactions on Computers*, 62(9):1760–1771, 2013.

[26] Doochul Shin and Sandeep K Gupta. A new circuit simplification method for error tolerant applications. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2011*, pages 1–6, 2011.

[27] Rami A Abdallah and Naresh R Shanbhag. Minimum-Energy Operation Via Error Resiliency. *IEEE Embedded Systems Letters*, 2(4):115–118, 2010.

[28] Byonghyo Shim, S R Sridhara, and Naresh R Shanbhag. Reliable low-power digital signal processing via reduced precision redundancy. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 12(5):497–510, 2004.

[29] Kumud Nepal, Yueting Li, R Iris Bahar, and Sherief Reda. ABACUS: A Technique for Automated Behavioral Synthesis of Approximate Computing Circuits. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, pages 361:1—-361:6, 3001 Leuven, Belgium, Belgium, 2014.

[30] Jiayuan Meng, Srimat Chakradhar, and Anand Raghunathan. Best-effort parallel execution framework for recognition and mining applications. In *IPDPS 2009 - Proceedings of the 2009 IEEE International Parallel and Distributed Processing Symposium*, pages 1–12, 2009.

[31] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[32] Israel Koren. *Computer Arithmetic Algorithms*. A. K. Peters, Ltd., Natick, MA, USA, 2nd edition, 2001.

[33] Daniele Jahier Pagliari, Yves Durand, David Coriat, Anca Molnos, Edith Beigne, Enrico Macii, and Massimo Poncino. A methodology for the design of dynamic accuracy operators by runtime back bias. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2017*, pages 1165–1170, 2017.

[34] Bert Moons and Marian Verhelst. DVAS: Dynamic Voltage Accuracy Scaling for increased energy-efficiency in approximate computing. In *IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED), 2015*, pages 237–242, 2015.

[35] Jan M. Rabaey. *Digital integrated circuits : a design perspective*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[36] Xuebei Yang and K Mohanram. Unequal-error-protection codes in SRAMs for mobile multimedia applications. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2011*, pages 21–27, nov 2011.

[37] Yun-Nan Chang and K K Parhi. Architectures for digital filters using stochastic computing. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2013*, pages 2697–2701, 2013.

[38] Daniele Jahier Pagliari, Massimo Poncino, and Enrico MacIi. Low-overhead adaptive constrast enhancement and power reduction for OLEDs. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2016*, pages 343–348, San Jose, CA, USA, 2016.

[39] Vinay K Chippa, Kaushik Roy, Srimat T Chakradhar, and Anand Raghunathan. Managing the Quality vs. Efficiency Trade-off Using Dynamic Effort Scaling. *ACM Trans. Embed. Comput. Syst.*, 12(2s):90:1—-90:23, 2013.

[40] B R Gaines. Stochastic Computing. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), pages 149–156, New York, NY, USA, 1967. ACM.

[41] W J Poppelbaum, C Afuso, and J W Esch. Stochastic Computing Elements and Systems. In *Proceedings of the November 14-16, 1967, Fall Joint Computer Conference*, AFIPS '67 (Fall), pages 635–644, New York, NY, USA, 1967. ACM.

[42] Warren J. Gross, Vincent C. Gaudet, and A Milner. Stochastic Implementation of LDPC Decoders. In *Conference Record of the Thirty-Ninth Asilomar Conference on Signals, Systems and Computers, 2005*, pages 713–717, 2005.

[43] Krishna Palem and Avinash Lingamneni. What to Do About the End of Moore's Law, Probably! In *Other*, pages 1–6, 2012.

[44] Lakshmi N B Chakrapani and Krishna V Palem. A Probabilistic Boolean Logic for Energy Efficient Circuit and System Design. In *Proceedings of the 2010 Asia and South Pacific Design Automation Conference*, ASPDAC '10, pages 628–635, Piscataway, NJ, USA, 2010. IEEE Press.

[45] Swagath Venkataramani, Amit Sabne, Vivek Kozhikkottu, Kaushik Roy, and Anand Raghunathan. SALSA: Systematic Logic Synthesis of Approximate Circuits. In *Proceedings of the 49th ACM/EDAC/IEEE Design Automation Conference (DAC), 2012*, page 796, 2012.

[46] Adrian Sampson, Werner Dietl, Emily Fortuna, Danushen Gnanapragasam, Luis Ceze, and Dan Grossman. EnerJ: Approximate Data Types for Safe and General Low-power Computation. *SIGPLAN Not.*, 46(6):164–174, jun 2011.

[47] Swagath Venkataramani, Vinay K Chippa, Srimat T Chakradhar, Kaushik Roy, and Anand Raghunathan. Quality Programmable Vector Processors for Approximate Computing. In *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-46, pages 1–12, New York, NY, USA, 2013. ACM.

[48] Bert Moons, Roel Uytterhoeven, Wim Dehaene, and Marian Verhelst. DVAFS: Trading computational accuracy for energy through dynamic-voltage-accuracy-frequency-scaling. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2017*, pages 488–493, 2017.

[49] Aaron Carroll and G Heiser. An analysis of power consumption in a smartphone. In *Proceedings of the USENIX annual technical conference*, USENIX-ATC'10, page 21, Berkeley, CA, USA, 2010. USENIX Association.

[50] Daniele Jahier Pagliari, Andrea Calimera, Enrico Macii, and Massimo Poncino. An automated design flow for approximate circuits based on reduced precision redundancy. In *Proceedings of the 33rd IEEE International Conference on Computer Design, ICCD 2015*, pages 86–93, 2015.

[51] Daniele Jahier Pagliari, Enrico Macii, and Massimo Poncino. Serial T0: Approximate Bus Encoding for Energy-efficient Transmission of Sensor Signals. In *Proceedings of the 53rd ACM/EDAC/IEEE Design Automation Conference (DAC), 2016*, DAC '16, pages 14:1—-14:6, New York, NY, USA, 2016. ACM.

[52] Daniele Jahier Pagliari, Enrico Macii, and Massimo Poncino. Approximate Differential Encoding for Energy-Efficient Serial Communication. In *Proceedings of the 26th Edition of the Great Lakes Symposium on VLSI*, GLSVLSI '16, pages 421–426, New York, NY, USA, 2016. ACM.

[53] Daniele Jahier Pagliari, Enrico Macii, and Massimo Poncino. Zero-Transition Serial Encoding for Image Sensors. *IEEE Sensors Journal*, 17(8):2563–2571, apr 2017.

[54] Daniele Jahier Pagliari, Enrico Macii, and Massimo Poncino. Approximate Energy-Efficient Encoding for Serial Interfaces. *ACM Transactions on Design Automation of Electronic Systems*, 22(4):1–25, 2017.

[55] Daniele Jahier Pagliari, Enrico Macii, and Massimo Poncino. Optimal content-dependent dynamic brightness scaling for oled displays. In *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS), 2017*, pages 1–6, 2017.

[56] Daniele Jahier Pagliari, Yves Durand, David Coriat, Edith Beigne, Enrico Macii, and Massimo Poncino. Fine-grain Back Biasing for the Design of Energy-Quality Scalable Operators. *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, X(X):1–14, In Press.

[57] Daniele Jahier Pagliari and Massimo Poncino. Application-Driven Synthesis of Energy-Efficient Reconfigurable-Precision Operators. In *IEEE International Symposium on Circuits and Systems (ISCAS), 2018*, pages 1–4, In Press.

[58] Daniele Jahier Pagliari, Enrico Macii, and Massimo Poncino. LAPSE: Low-Overhead Adaptive Power Saving and Contrast Enhancement for OLEDs. *IEEE Transaction on Image Processing*, X(X):1–15, Under Review.

[59] Hamid R. Mahdiani, Ali Ahmadi, Sied M. 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, apr 2010.

[60] 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.

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

[62] 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.

[63] Ning Zhu, Wang Ling Goh, and Kiat Seng Yeo. An enhanced low-power high-speed Adder For Error-Tolerant application. In *Proceedings of the 2009 12th International Symposium on Integrated Circuits, ISIC '09*, pages 69–72, 2009.

[64] Ning Zhu, Wang Ling Goh, and Kiat Seng Yeo. Ultra low-power high-speed flexible Probabilistic Adder for Error-Tolerant Applications. In *2011 International SoC Design Conference, ISSOC 2011*, pages 393–396, 2011.

[65] Ning Zhu, Wang Ling Goh, Gang Wang, and Kiat Seng Yeo. Enhanced low-power high-speed adder for error-tolerant application. In *2010 International SoC Design Conference, ISOCC 2010*, pages 323–327, 2010.

[66] Shih-Lien Lu. Speeding up processing with approximation circuits. *Computer*, 37(3):67–73, mar 2004.

[67] Jiawei Huang, John Lach, and Gabriel Robins. A methodology for energy-quality tradeoff using imprecise hardware. In *Proceedings of the 49th ACM/EDAC/IEEE Design Automation Conference (DAC), 2012*, pages 504–509, 2012.

[68] 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*, GLSVLSI '15, pages 343–348, New York, NY, USA, 2015. ACM.

[69] Farzad Farshchi, Muhammad S. Abrishami, and Sied M. Fakhraie. New approximate multiplier for low power digital signal processing. In *The 17th CSI International Symposium on Computer Architecture Digital Systems (CADS 2013)*, pages 25–30, 2013.

[70] Khaing Yin Kyaw, Wang Ling Goh, and Kiat Seng Yeo. Low-power high-speed multiplier for error-tolerant application. In *2010 IEEE International Conference of Electron Devices and Solid-State Circuits (EDSSC)*, pages 1–4, 2010.

[71] P Kulkarni, P Gupta, and M Ercegovac. Trading Accuracy for Power with an Underdesigned Multiplier Architecture. In *VLSI Design (VLSI Design), 2011 24th International Conference on*, pages 346–351, 2011.

[72] Linbin Chen, Jie Han, Weiqiang Liu, and Fabrizio Lombardi. Design of Approximate Unsigned Integer Non-restoring Divider for Inexact Computing. In *Proceedings of the 25th Edition on Great Lakes Symposium on VLSI*, GLSVLSI '15, pages 51–56, New York, NY, USA, 2015. ACM.

[73] Ajay K. Verma, Philip Brisk, and Paolo Ienne. Variable Latency Speculative Addition: A New Paradigm for Arithmetic Circuit Design. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2008*, pages 1250–1255, 2008.

[74] Rong Ye, Ting Wang, Feng Yuan, Rakesh Kumar, and Qiang Xu. On reconfiguration-oriented approximate adder design and its application. In *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 48–54, nov 2013.

[75] Manuel de la Guia Solaz, Wei Han, and Richard Conway. A Flexible Low Power DSP With a Programmable Truncated Multiplier. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 59(11):2555–2568, nov 2012.

[76] Cong Liu, Jie Han, and Fabrizio Lombardi. A Low-power, High-performance Approximate Multiplier with Configurable Partial Error Recovery. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, pages 95:1—-95:4, 3001 Leuven, Belgium, Belgium, 2014.

[77] Benjamin Barrois, Olivier Sentieys, and Daniel Menard. The hidden cost of functional approximation against careful data sizing - A case study. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2017*, pages 181–186, 2017.

[78] Bert Moons, Roel Uytterhoeven, Wim Dehaene, and Marian Verhelst. Envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable Convolutional Neural Network processor in 28nm FDSOI. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 246–247, feb 2017.

[79] Ashish Ranjan, Arnab Raha, Swagath Venkataramani, Kaushik Roy, and Anand Raghunathan. ASLAN: Synthesis of approximate sequential circuits. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, pages 1–6, 2014.

[80] Avinash Lingamneni, Christian Enz, Krishna Palem, and Christian Piguet. Synthesizing Parsimonious Inexact Circuits Through Probabilistic Design Techniques. *ACM Trans. Embed. Comput. Syst.*, 12(2s):93:1—-93:26, 2013.

[81] Swagath Venkataramani, Kaushik Roy, and Anand Raghunathan. Substitute-and-simplify: A unified design paradigm for approximate and quality configurable circuits. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2013*, pages 1367–1372, 2013.

[82] Chaofan Li, Wei Luo, Sachin S. Sapatnekar, and Jiang Hu. Joint precision optimization and high level synthesis for approximate computing. In *Proceedings of the 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), 2015*, pages 1–6, 2015.

[83] Andrew B. Kahng, Seokhyeong Kang, Rakesh Kumar, and John Sartori. Slack redistribution for graceful degradation under voltage overscaling. In *15th Asia and South Pacific Design Automation Conference (ASP-DAC), 2010*, pages 825–831, 2010.

[84] Debabrata Mohapatra, Vinay K. Chippa, Anand Raghunathan, and Kaushik Roy. Design of voltage-scalable meta-functions for approximate computing. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2011*, pages 1–6, 2011.

[85] Rajamohana Hegde and Naresh R. Shanbhag. Soft digital signal processing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 9(6):813–823, 2001.

[86] Byonghyo Shim and Naresh R. Shanbhag. Performance analysis of algorithmic noise-tolerance techniques. In *Proceedings of the International Symposium on Circuits and Systems (ISCAS), 2003*, volume 4, pages IV–113–IV–116 vol.4, 2003.

[87] Sai Zhang and Naresh R. Shanbhag. Embedded Algorithmic Noise-Tolerance for Signal Processing and Machine Learning Systems via Data Path Decomposition. *IEEE Transactions on Signal Processing*, 64(13):3338–3350, 2016.

[88] Nilanjan Banerjee, Georgios Karakonstantis, and Kaushik Roy. Process Variation Tolerant Low Power DCT Architecture. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2007*, pages 1–6, 2007.

[89] Debabrata Mohapatra, Georgios Karakonstantis, and Kaushik Roy. Significance Driven Computation: A Voltage-scalable, Variation-aware, Quality-tuning Motion Estimator. In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED), 2009*, pages 195–200, New York, NY, USA, 2009. ACM.

[90] Georgios Karakonstantis, Debabrata Mohapatra, and Kaushik Roy. System level DSP synthesis using voltage overscaling, unequal error protection and adaptive quality tuning. In *IEEE Workshop on Signal Processing Systems (SiPS), 2009*, pages 133–138, 2009.

[91] Ku He, A Gerstlauer, and M Orshansky. Controlled timing-error acceptance for low energy IDCT design. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2011*, pages 1–6, 2011.

[92] Dan Ernst, Shidhartha Das, Seokwoo Lee, David Blaauw, Todd Austin, Trevor Mudge, Nam Sung Kim, and Krisztian Flautner. Razor: circuit-level correction of timing errors for low-power operation. *IEEE Micro*, 24(6):10–20, nov 2004.

[93] S Ghosh, S Bhunia, and K Roy. CRISTA: A New Paradigm for Low-Power, Variation-Tolerant, and Adaptive Circuit Synthesis Using Critical Path Isolation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(11):1947–1956, nov 2007.

[94] Shidhartha Das, Carlos Tokunaga, Sanjay Pant, Wei-Hsiang Ma, Sudherssen Kalaiselvan, Kevin Lai, David M. Bull, and David T. Blaauw. Razorii: In situ error detection and correction for pvt and ser tolerance. *IEEE Journal of Solid-State Circuits*, 44(1):32–48, Jan 2009.

[95] Larkhoon Leem, Hyungmin Cho, Jason Bau, Quinn A. Jacobson, and Subhasish Mitra. ERSA: Error Resilient System Architecture for probabilistic applications. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2010*, pages 1560–1565, 2010.

[96] Yavuz Yetim, Margaret Martonosi, and Sharad Malik. Extracting Useful Computation from Error-prone Processors for Streaming Applications. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2013*, pages 202–207, San Jose, CA, USA, 2013.

[97] Hadi Esmaeilzadeh, Adrian Sampson, Luis Ceze, and Doug Burger. Neural acceleration for general-purpose approximate programs. In *IEEE Micro*, volume 33, pages 16–27, 2013.

[98] Swagath Venkataramani, Ashish Ranjan, Kaushik Roy, and Anand Raghunathan. AxNN: Energy-efficient Neuromorphic Systems Using Approximate Computing. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED), 2014*, pages 27–32, 2014.

[99] Bert Moons, B De Brabandere, L Van Gool, and Marian Verhelst. Energy-efficient ConvNets through approximate computing. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–8, 2016.

[100] Qian Zhang, Ting Wang, Ye Tian, Feng Yuan, and Qiang Xu. ApproxANN: An approximate computing framework for artificial neural network. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2015*, pages 701–706, 2015.

[101] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1. *arXiv:1602.02830*, 2016.

[102] Renzo Andri, Lukas Cavigelli, Davide Rossi, and Luca Benini. YodaNN: An ultra-low power convolutional neural network accelerator based on binary weights. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 236–241, 2016.

[103] Eunhyeok Park, Dongyoung Kim, Soobeom Kim, Yong-Deok Kim, Gunhee Kim, Sungroh Yoon, and Sungjoo Yoo. Big/little deep neural network for ultra low power inference. In *IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pages 124–132, 2015.

[104] Hokchhay Tann, Soheil Hashemi, R. Iris Bahar, and Sherief Reda. Run-time configurable deep neural networks for energy-accuracy trade-off. In *IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES)*, pages 1–10, 2016.

[105] Swagath Venkataramani, Anand Raghunathan, Jie Liu, and Mohammed Shoaib. Scalable-effort classifiers for energy-efficient machine learning. In *Proceedings of the 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), 2015*, pages 1–6, 2015.

[106] Vinay K. Chippa, Debabrata Mohapatra, Anand Raghunathan, Kaushik Roy, and Srimat T. Chakradhar. Scalable Effort Hardware Design: Exploiting Algorithmic Resilience for Energy Efficiency. In *Proceedings of the 47th ACM/IEEE Design Automation Conference (DAC), 2010*, pages 555–560, 2010.

[107] 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 - SIGSOFT/FSE '11*, ESEC/FSE '11, page 124, New York, NY, USA, 2011. ACM.

[108] Michael Carbin, Sasa Misailovic, and Martin C. Rinard. Verifying quantitative reliability for programs that execute on unreliable hardware. In *Proceedings of the 2013 ACM SIGPLAN international conference on Object oriented programming systems languages & applications - OOPSLA '13*, pages 33–52, New York, NY, USA, 2013. ACM.

[109] Vinay Chippa, Anand Raghunathan, Kaushik Roy, and Srimat Chakradhar. Dynamic effort scaling: Managing the quality-efficiency tradeoff. In *Proceedings of the 48th ACM/EDAC/IEEE Design Automation Conference (DAC), 2011*, pages 603–608, 2011.

[110] Woongki Baek and Trishul M Chilimbi. Green: A Framework for Supporting Energy-conscious Programming Using Controlled Approximation. In *Proceedings of the 31st ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '10, pages 198–209, New York, NY, USA, 2010. ACM.

[111] Mehrzad Samadi, Janghaeng Lee, D. Anoushe Jamshidi, Amir Hormati, and Scott Mahlke. SAGE: Self-Tuning Approximation for Graphics Engines. In *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture - MICRO-46*, pages 13–24, 2013.

[112] Daya S. Khudia, Babak Zamirai, Mehrzad Samadi, and Scott Mahlke. Rumba: An online quality management system for approximate computing. In *42nd ACM/IEEE Annual International Symposium on Computer Architecture (ISCA), 2015*, pages 554–566, 2015.

[113] Ting Wang, Qian Zhang, Nam Sung Kim, and Qiang Xu. On Effective and Efficient Quality Management for Approximate Computing. In *Proceedings of the 2016 International Symposium on Low Power Electronics and Design (ISLPED)*, pages 156–161, 2016.

[114] Divya Mahajan, Amir Yazdanbaksh, Jongse Park, Bradley Thwaites, and Hadi Esmaeilzadeh. Towards Statistical Guarantees in Controlling Quality Tradeoffs for Approximate Acceleration. In *43rd ACM/IEEE Annual International Symposium on Computer Architecture (ISCA), 2016*, pages 66–77, 2016.

[115] Ik Joon Chang, Debabrata Mohapatra, and Kaushik Roy. A Priority-based 6T/8T hybrid SRAM architecture for aggressive voltage scaling in video applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(2):101–112, 2011.

[116] Samira Ataei and James E. Stine. A 64 kB Approximate SRAM Architecture for Low-Power Video Applications. *IEEE Embedded Systems Letters*, 10(1):10–13, 2018.

[117] Song Liu, Karthik Pattabiraman, Thomas Moscibroda, and Benjamin G. Zorn. Flikker: Saving DRAM Refresh-power Through Critical Data Partitioning. In *Proceedings of the sixteenth international conference on Architectural support*

for programming languages and operating systems (ASPLOS), page 213, New York, NY, USA, 2011.

[118] Arnab Raha, Hrishikesh Jayakumar, Soubhagya Sutar, and Vijay Raghunathan. Quality-aware data allocation in approximate DRAM. In *International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES), 2015*, pages 89–98, 2015.

[119] Matthias Jung, Éder Zulian, Deepak M Mathew, Matthias Herrmann, Christian Brugger, Christian Weis, and Norbert Wehn. Omitting Refresh: A Case Study for Commodity and Wide I/O DRAMs. In *Proceedings of the International Symposium on Memory Systems (MEMSYS), 2015*, pages 85–91, New York, NY, USA, 2015.

[120] Matthias Jung, Deepak M Mathew, Christian Weis, and Norbert Wehn. Invited - Approximate computing with partially unreliable dynamic random access memory - approximate DRAM. In *Proceedings of the 53rd ACM/EDAC/IEEE Design Automation Conference (DAC), 2016*, pages 1–4, 2016.

[121] Adrian Sampson, Jacob Nelson, Karin Strauss, and Luis Ceze. Approximate Storage in Solid-State Memories. *ACM Transactions on Computer Systems*, 32(3):1–23, sep 2014.

[122] Frank Vahid and Tony Givargis. *Embedded System Design: A Unified Hardware/Software Approach*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1999.

[123] Motorola. *MC68HC11 Reference Manual*, 1989.

[124] NXP. *I2C-bus specification and user manual*. NXP Semiconductors, 2014.

[125] ISO. *ISO 11898-1:2015, Road vehicles – Controller area network (CAN)*. International Organization for Standardization, 2015.

[126] Kangmin Lee, Se-Joong Lee, and Hoi-Jun Yoo. SILENT: serialized low energy transmission coding for on-chip interconnection networks. In *IEEE/ACM International Conference on Computer Aided Design (ICCAD), 2004*, pages 448–451, 2004.

[127] Xianglong Ren, Deyuan Gao, Xiaoya Fan, and Jianfeng An. Adaptive Low-Power Transmission Coding for Serial Links in Network-on-Chip. *Procedia Engineering*, 29:1618–1624, 2012.

[128] Jian Zeng, Jian-Yang Zhou, and Rung-Bin Lin. Transition inversion coding with parity check for off-chip serial transmission. In *21st IEEE International Conference on Electronics, Circuits and Systems (ICECS), 2014*, pages 634–637, 2014.

[129] Somrita Ghosh, Prasun Ghosal, Nabanita Das, Saraju P Mohanty, and Oghenekarho Okobiah. Data Correlation Aware Serial Encoding for Low Switching Power On-Chip Communication. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2014*, pages 124–129, 2014.

[130] Sabino Salerno, Alberto Bocca, Enrico Macii, and Massimo Poncino. Limited Intra-Word Transition Codes: An Energy-Efficient Bus Encoding for LCD Display Interfaces. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED), 2004*, pages 206–211, 2004.

[131] Massimo Poncino and Enrico Macii. Low-energy RGB color approximation for digital LCD interfaces. *IEEE Transactions on Consumer Electronics*, 52(3):1004–1012, 2006.

[132] Phillip Stanley-Marbell and Martin Rinard. Reducing Serial I/O Power in Error-tolerant Applications by Efficient Lossy Encoding. In *Proceedings of the 53rd ACM/EDAC/IEEE Design Automation Conference (DAC), 2016*, pages 62:1—-62:6, New York, NY, USA, 2016.

[133] Younghyun Kim, Setareh Behroozi, Vijay Raghunathan, and Anand Raghunathan. AXSERBUS: A quality-configurable approximate serial bus for energy-efficient sensing. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED), 2017*, pages 1–6, 2017.

[134] Achintya K. Bhowmik and Robert J. Brennan. System-Level Display Power Reduction Technologies for Portable Computing and Communications Devices. In *2007 IEEE International Conference on Portable Information Devices*, pages 1–5, 2007.

[135] Wei-Chung Cheng and Massoud Pedram. Power minimization in a backlit TFT-LCD display by concurrent brightness and contrast scaling. *IEEE Transactions on Consumer Electronics*, 50(1):25–32, feb 2004.

[136] Ali Iranli, Wonbok Lee, and Massoud Pedram. Backlight dimming in power-aware mobile displays. In *Proceedings of the 43rd ACM/IEEE Design Automation Conference (DAC), 2006*, pages 604–607, 2006.

[137] Hojun Shim, Naehyuck Chang, and Massoud Pedram. A backlight power management framework for battery-operated multimedia systems. *IEEE Design and Test of Computers*, 21(5):388–396, sep 2004.

[138] Chih-Chang Lai and Ching-Chih Tsai. Backlight power reduction and image contrast enhancement using adaptive dimming for global backlight applications. *IEEE Transactions on Consumer Electronics*, 54(2):669–674, may 2008.

[139] Chan Young Jang, Suk-Ju Kang, and Young Hwan Kim. Perceived Distortion-Based Progressive LCD Backlight Dimming Method. *Journal of Display Technology*, 12(10):1130–1138, oct 2016.

[140] Mian Dong and Lin Zhong. Chameleon: A Color-Adaptive Web Browser for Mobile OLED Displays. *IEEE Transactions on Mobile Computing*, 11(5):724–738, may 2012.

[141] Mian Dong and Lin Zhong. Power Modeling and Optimization for OLED Displays. *IEEE Transactions on Mobile Computing*, 11(9):1587–1599, sep 2012.

[142] Parthasarathy Ranganathan, Erik Geelhoed, Meera Manahan, and Ken Nicholas. Energy-aware user interfaces and energy-adaptive displays. *Computer*, 39(3):31–38, mar 2006.

[143] Chun-Han Lin, Chih-Kai Kang, and Pi-Cheng Hsiu. Catch Your Attention. In *Proceedings of the 51st ACM/EDAC/IEEE Design Automation Conference (DAC), 2014*, pages 1–6, New York, New York, USA, 2014. ACM Press.

[144] Donghwa Shin, Younghyun Kim, Naehyuck Chang, and M. Pedram. Dynamic Driver Supply Voltage Scaling for Organic Light Emitting Diode Displays. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(7):1017–1030, jul 2013.

[145] Xiang Chen, Jian Zeng, Yiran Chen, Wei Zhang, and Hai Li. Fine-grained dynamic voltage scaling on OLED display. In *17th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 807–812. IEEE, jan 2012.

[146] Suk-Ju Kang. Image-Quality-Based Power Control Technique for Organic Light Emitting Diode Displays. *Journal of Display Technology*, 11(1):104–109, jan 2015.

[147] Suk-ju Kang. Perceptual Quality-Aware Power Reduction Technique for Organic Light Emitting Diodes. *Journal of Display Technology*, 12(6):519–525, jun 2016.

[148] Chulwoo Lee, Chul Lee, Young-Yoon Lee, and Chang-Su Kim. Power-Constrained Contrast Enhancement for Emissive Displays Based on Histogram Equalization. *IEEE Transactions on Image Processing*, 21(1):80–93, jan 2012.

[149] Yeon-Oh Nam, Dong-Yoon Choi, and Byung Cheol Song. Power-Constrained Contrast Enhancement Algorithm Using Multiscale Retinex for OLED Display. *IEEE Transactions on Image Processing*, 23(8):3308–3320, aug 2014.

[150] Chan Young Jang, Suk-Ju Kang, and Young Hwan Kim. Noniterative Power-Constrained Contrast Enhancement Algorithm for OLED Display. *Journal of Display Technology*, 12(11):1257–1267, nov 2016.

[151] Yan-Tsung Peng, Fan-Chieh Cheng, Li-Ming Jan, and Shanq-Jang Ruan. Histogram shrinking for power-saving contrast enhancement. In *2013 IEEE International Conference on Image Processing*, pages 891–894, sep 2013.

[152] Jeyong Shin and Rae-Hong Park. Power-constrained contrast enhancement for organic light-emitting diode display using locality-preserving histogram equalisation. *IET Image Processing*, 10(7):542–551, jul 2016.

[153] Chih-Kai Kang, Chun-Han Lin, and Pi-Cheng Hsiu. A win-win camera: Quality-enhanced power-saving images on mobile OLED displays. In *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 267–272, 2015.

[154] Xiang Chen, Jiachen Mao, Jiafei Gao, Kent W. Nixon, and Yiran Chen. MORPh. In *Proceedings of the 53rd ACM/EDAC/IEEE Design Automation Conference (DAC), 2016*, pages 1–6, New York, New York, USA, 2016. ACM Press.

[155] Opencores. https://opencores.org.

[156] Alan V Oppenheim, Ronald W Schafer, and John R Buck. *Discrete-time Signal Processing (2Nd Ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1999.

[157] Jack E Volder. The CORDIC Trigonometric Computing Technique. *IRE Transactions on Electronic Computers*, EC-8(3):330–334, 1959.

[158] Pooja Choudhary and Abhijit Karmakar. CORDIC based implementation of Fast Fourier Transform. In *2nd International Conference on Computer and Communication Technology (ICCCT), 2011*, pages 550–555, 2011.

[159] Pierre-Emmanuel Gaillardon, Edith Beigne, Suzanne Lesecq, and Giovanni De Micheli. A survey on low-power techniques with emerging technologies: From devices to systems. *J. Emerg. Technol. Comput. Syst.*, 12(2):12:1–12:26, September 2015.

[160] Edith Beigné, Alexandre Valentian, Ivan Miro-Panades, Robin Wilson, Philippe Flatresse, Fady Abouzeid, Christian Benoist, Thomasand Bernard, Sebastien Bernard, Olivier Billoint, Sylvain Clerc, Bastien Giraud, Anuj Grover, Julien Le Coz, Olivier Noel, Jean Philippe an Thomas, and Yvaine Thonnart. A 460 mhz at 397 mv, 2.6 ghz at 1.3 v, 32 bits vliw dsp embedding f max tracking. *IEEE Journal of Solid-State Circuits*, 50(1):125–136, Jan 2015.

[161] Giovanni De Micheli. *Synthesis and Optimization of Digital Circuits*. McGraw-Hill Higher Education, 1st edition, 1994.

[162] Jingcao Hu, Youngsoo Shin, Nagu Dhanwada, and Radu Marculescu. Architecting voltage islands in core-based system-on-a-chip designs. In *Proceedings of the IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED), 2004*, pages 180–185, 2004.

[163] Elnaz Ebrahimi, Rafael T. Possignolo, and Jose Renau. Level shifter design for voltage stacking. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4, May 2017.

[164] Saurabh Dighe, Sriram R. Vangal, Paolo Aseron, Shasi Kumar, Tiju Jacob, Keith A. Bowman, Jason Howard, James Tschanz, Vasantha Erraguntla, Nitin Borkar, Vivek K. De a, and Shekhar Borkar. Within-die variation-aware dynamic-voltage-frequency-scaling with optimal core allocation and thread hopping for the 80-core teraflops processor. *IEEE Journal of Solid-State Circuits*, 46(1):184–193, Jan 2011.

[165] Sergio M. Martin, Krisztian Flautner, Trevor Mudge, and David Blaauw. Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads. In *IEEE/ACM International Conference on Computer Aided Design (ICCAD), 2002*, pages 721–725, Nov 2002.

[166] Martin Cochet, Bertrand Pelloux-Prayer, Mehdi Saligane, Sylvain Clerc, Philippe Roche, Jean-Luc Autran, and Dennis Sylvester. Experimental model of adaptive body biasing for energy efficiency in 28nm utbb fd-soi. In *2014 SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*, pages 1–2, Oct 2014.

[167] Diego Puschini, Jorge Rodas, Edith Beigne, Mauricio Altieri, and Suzanne Lesecq. Body bias usage in utbb fdsoi designs: A parametric exploration approach. *Solid-State Electronics*, 117:138 – 145, 2016.

[168] Kimiyoshi Usami and Mark Horowitz. Clustered voltage scaling technique for low-power design. In *Proceedings of the IEEE/ACM International Symposium on Low Power Design (ISLPED), 1995*, pages 3–8, New York, NY, USA, 1995.

[169] T. Kutzschebauch and Leon Stok. Regularity driven logic synthesis. In *IEEE/ACM International Conference on Computer Aided Design (ICCAD), 2000*, pages 439–446, 2000.

[170] Qi Wang and S. B. K. Vrudhula. Algorithms for minimizing standby power in deep submicrometer, dual-vt cmos circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(3):306–318, Mar 2002.

[171] Kaushik Roy, Saibal Mukhopadhyay, and Hamid Mahmoodi-Meimand. Leakage current mechanisms and leakage reduction techniques in deep-submicrometer cmos circuits. *Proceedings of the IEEE*, 91(2):305–327, Feb 2003.

[172] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Low precision arithmetic for deep learning. *arXiv:1412.7024*, 2014.

[173] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep Learning with Limited Numerical Precision. *arXiv:1503.02551*, 2015.

[174] Wouter Bracke, Robert Puers, and Chris Van Hoof. *Ultra Low Power Capacitive Sensor Interfaces*. Springer Netherlands, 1st edition, 2007.

[175] Nathan Ickes, Daniel Finchelstein, and Anantha P. Chandrakasan. A 10-pj/instruction, 4-mips micropower dsp for sensor applications. In *Proceedings of the IEEE Asian Solid-State Circuits Conference (A-SSCC), 2008*, pages 289–292, Nov 2008.

[176] J.S. Chitode. *Information Coding Techniques*. Technical Publications, 2007.

[177] Kodak. Image database. http://r0k.us/graphics/kodak.

[178] Muhammad Shoaib, Hans Scholten, and Paul J. M. Havinga. Towards physical activity recognition using smartphone sensors. In *Proceedings of the 10th IEEE International Conference on Ubiquitous Intelligence and Computing and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)*, pages 80–87, 2013.

[179] Ary L. Goldberger, Luis A.N. Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen Ch. Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung-Kang Peng, and H. Eugene Stanley. Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000.

[180] OSR. Open speech repository. http://www.voiptroubleshooter.com/open_speech/index.html.

[181] Paul E. Landman and Jan M. Rabaey. Architectural Power Analysis: The Dual Bit Type Method. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 3(2):173–187, 1995.

[182] Philips. *I2S bus specification*. Philips Semiconductors, 1996.

[183] EIA. *Interface between data terminal equipment and data communication equipment employing serial binary data interchange*. Electronic Industries Association: Engineering Department, 1969.

[184] MIPI. *Display Interface Specifications*. MIPI Alliance, 2004.

[185] Gabriele Manganaro. *Advanced Data Converters*. Cambridge University Press, 2011.

[186] Maxim. *MAX11102/03/05/06/10/11/15/16/17 2Msps/3Msps, Low-Power, Serial 12-/10-/8-Bit ADCs. Datasheet*. Maxim Integrated, 2013.

[187] Luca Benini, Giovanni De Micheli, Enrico Macii, Donatella Sciuto, and Cristina Silvano. Asymptotic zero-transition activity encoding for address busses in low-power microprocessor-based systems. In *Proceedings of the Seventh Great Lakes Symposium on VLSI (GLS-VLSI), 1997*, pages 77–82, 1997.

[188] Yuen-Chuan Lu, Zong-Yi Chen, and Pao-Chi Chang. Low power multi-lane mipi csi-2 receiver design and hardware implementations. In *IEEE International Symposium on Consumer Electronics (ISCE), 2013*, pages 199–200, June 2013.

[189] MIPI Alliance. *Physical Layer Specifications*, 2009.

[190] NXP. *MMA8452Q, 3-axis, 12-bit/8-bit digital accelerometer*. NXP Semiconductors, 2016.

[191] Freescale. *Xtrinsic MAG3110 Three-Axis, Digital Magnetometer*. Freescale Semiconductor, 2013.

[192] STM. *A3G4250D MEMS motion sensor: 3-axis digital output gyroscope*. ST Microelectronics, 2012.

[193] Ray Smith. An overview of the tesseract ocr engine. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR), 2007*, pages 629–633, 2007.

[194] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2001*, pages I–511–I–518, 2001.

[195] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51 – 59, 1996.

[196] Open source computer vision (open cv) library. http://opencv.org/.

[197] Libor Spacek. Face recognition data. http://cswww.essex.ac.uk/mv/allfaces/index.html, 2007.

[198] Simon Lucas. *Robust Word Recognition Dataset*. ICDAR, 2003.

[199] David Salomon. *Data Compression: The Complete Reference*. Springer, 2006.

[200] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, 2001.

[201] Mark I. Montrose. *EMC and the Printed Circuit Board: Design, Theory, and Layout Made Simple*. Wiley-IEEE Press, 1998.

[202] James Norman Bardsley. International OLED technology roadmap. *IEEE Journal on Selected Topics in Quantum Electronics*, 10(1):3–9, jan 2004.

[203] Vitor C. Bender, Tiago B. Marchesan, and J. Marcos Alonso. Solid-State Lighting: A Concise Review of the State of the Art on LED and OLED Modeling. *IEEE Industrial Electronics Magazine*, 9(2):6–16, jun 2015.

[204] Tajana Simunic, Luca Benini, Peter Glynn, and Giovanni De Micheli. Event-driven power management. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(7):840–857, jul 2001.

[205] Vipin Kamble and K.M. Bhurchandi. No-reference image quality assessment algorithms: A survey. *Optik - International Journal for Light and Electron Optics*, 126(11-12):1090–1097, jun 2015.

[206] Andreas Koschan and Mongi A. Abidi. *Digital Color Image Processing*. Wiley, 2008.

[207] ITU-T. Information technology - digital compression and coding of continuous-tone still images; jpeg file interchange format (jfif), May 2011.

[208] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, May 2011.

[209] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *Proceedings of the 10th European Conference on Computer Vision: Part I*, ECCV '08, pages 304–317, Berlin, Heidelberg, 2008. Springer-Verlag.

[210] The open video project. open-video.org.

[211] Derf's test media collection. https://media.xiph.org/video/derf/.

[212] https://vimeo.com/album/4457406.

[213] US Micro. *USMP-A24240TP AMOLED Product Specification, Ver. 1.2*. US Micro Products, 2008.

[214] D. Brunet, E. R. Vrscay, and Z. Wang. On the mathematical properties of the structural similarity index. *IEEE Transactions on Image Processing*, 21(4):1488–1499, April 2012.

[215] H.R. Sheikh, Z. Wang, L. Cormack, and A.C. Bovik. Live image quality assessment database release 2.

[216] Sos S. Agaian, Karen P. Lentz, and Artyom M. Grigoryan. A new measure of image enhancement. In *IASTED International Conference on Signal Processing & Communication*, 2000.

[217] https://vimeo.com/album/4604862.

Appendix A

A List of Publications

This appendix contains a complete list of the publications co-authored by Daniele Jahier Pagliari during the years of his PhD. This list also includes publications not strictly related to the content presented in this thesis, but still in the domain of EDA and digital design. Overall, Daniele Jahier Pagliari has co-authored:

- 4 accepted papers in international peer-reviewed journals
- 12 accepted papers in international peer-reviewed conferences
- 2 book chapters
- 1 patent

International Peer-Reviewed Journals

J.1 Daniele Jahier Pagliari, Enrico Macii and Massimo Poncino, Zero-Transition Serial Encoding for Image Sensors, In: *IEEE Sensors Journal*, vol. 17, n. 8, pp. 2563–2571, ISSN: 1530-437X, Apr. 2017.

J.2 Daniele Jahier Pagliari, Mario R. Casu and Luca P. Carloni, Accelerators for Breast Cancer Detection, In: *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, n. 3, pp. 80:1–80:25, ISSN: 1539-9087, July 2017.

J.3 Daniele Jahier Pagliari, Enrico Macii and Massimo Poncino, Approximate Energy-Efficient Encoding for Serial Interfaces, In: *ACM Transactions on*

Design Automation of Electronic Systems (TODAES), vol. 22, n. 4, pp. 64:1–64:25, ISSN: 1084-4309, July 2017.

J.4 Daniele Jahier Pagliari, Yves Durand, David Coriat, Edith Beigne, Macii Enrico and Massimo Poncino, Fine-grain Back Biasing for the Design of Energy-Quality Scalable Operators, In: *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, vol. X, n. X, pp. 1–14, In Press.

International Peer-Reviewed Conferences

C.1 Daniele Jahier Pagliari, Andrea Calimera, Enrico Macii and Massimo Poncino, An automated design flow for approximate circuits based on reduced precision redundancy, In: *Proceedings of the 33rd International Conference on Computer Design (ICCD)*, pp. 86–93, ISBN: 978-1-4673-7166-7, 2015.

C.2 Daniele Jahier Pagliari, Mario R. Casu and Luca P. Carloni, Acceleration of microwave imaging algorithms for breast cancer detection via High-Level Synthesis, In: *Proceedings of the 33rd International Conference on Computer Design (ICCD)*, pp. 475–478, ISBN: 978-1-4673-7166-7, 2015.

C.3 Daniele Jahier Pagliari, Azzurra Pulimeno, Marco Vacca, Jorge A. Tobon, Francesca Vipiana, Mario R. Casu, Raffaele Solimene and Luca P. Carloni, A low-cost, fast, and accurate microwave imaging system for breast cancer detection, In: *Proceedings of the IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pp. 1–4, ISBN: 978-1-4799-7234-0 , 2015.

C.4 Daniele Jahier Pagliari, Massimo Poncino and Enrico Macii, Low-overhead adaptive constrast enhancement and power reduction for OLEDs, In: *Proceedings of the Design, Automation & Test in Europe Conference and Exhibition (DATE)*, pp. 343–348, ISBN: 978-3-9815370-6-2, 2016.

C.5 Daniele Jahier Pagliari, Enrico Macii and Massimo Poncino, Approximate differential encoding for energy-efficient serial communication, In: *Proceedings of the 26th edition on Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 421–426, ISBN: 978-1-4503-4274-2, 2016.

C.6 Daniele Jahier Pagliari, Enrico Macii and Massimo Poncino, Serial T0: approximate bus encoding for energy-efficient transmission of sensor signals, In:

Proceedings of the 53rd ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1–6, ISBN: 978-1-4503-4236-0, 2016.

C.7 Daniele Jahier Pagliari, Yves Durand, David Coriat, Anca Molnos, Edith Beigne, Enrico Macii and Massimo Poncino, A methodology for the design of dynamic accuracy operators by runtime back bias, In: *Proceedings of the Design, Automation & Test in Europe Conference and Exhibition (DATE)*, pp. 1165–1170, ISBN: 978-3-9815370-8-6, 2017.

C.8 Daniele Jahier Pagliari, Enrico Macii and Massimo Poncino, Optimal content-dependent dynamic brightness scaling for OLED displays, In: *Proceedings of the 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pp. 1–6, ISBN: 978-1-5090-6462-5, 2017.

C.9 Daniele Jahier Pagliari, Enrico Macii and Massimo Poncino, Optimal content-dependent dynamic brightness scaling for OLED displays, In: *Proceedings of the 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pp. 1–6, ISBN: 978-1-5090-6462-5, 2017.

C.10 Daniele Jahier Pagliari, Valentino Peluso, Yukai Chen, Andrea Calimera, Enrico Macii and Massimo Poncino, All-digital embedded meters for on-line power estimation, In: *Proceedings of the Design, Automation & Test in Europe Conference and Exhibition (DATE)*, pp. 1–6, ISBN: 978-3-9819263-0-9, 2018.

C.11 Daniele Jahier Pagliari and Massimo Poncino, Application-driven synthesis of energy-efficient reconfigurable-precision operators, In: *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–4, ISBN: XXX, In Press.

C.12 Yukai Chen, Daniele Jahier Pagliari, Enrico Macii and Massimo Poncino, Battery-aware Design Exploration of Scheduling Policies for Multi-sensor Devices, In: *Proceedings of the 28th edition on Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 1–6, ISBN: XXX, In Press.

Book Chapters

B.1 Daniele Jahier Pagliari, Massimo Poncino and Enrico Macii, Energy-Efficient Digital Processing via Approximate Computing, In: *Smart Systems Integration and Simulation*, pp. 55–89, ISBN: 978-3-319-47913-2, Springer International Publishing, 2016.

B.2 Azzurra Pulimeno, Marco Vacca, Mario R. Casu, Jorge A. Tobon, Francesca Vipiana, Daniele Jahier Pagliari, Raffaele Solimene and Luca P. Carloni, Microwave imaging for breast cancer detection: a COTS-based prototype, In: *Applications in Electronics Pervading Industry, Environment and Society*, pp. 25–34, ISBN: 978-3-319-47913-2, Springer International Publishing, 2017.

Patents

P.1 Daniele Jahier Pagliari, Massimo Poncino, Yves Durand and Edith Beigne, Processing circuit capable of dynamically modifying its precision, Under Filing.