Nonlinear Operators for Image Processing: Design, Implementation and Modelling Techniques for Power Estimation

Giuseppe Bernacchia

November 28, 1999

University of Trieste

Ph.D. Program in Information Engineering

— XII Cycle —

Ph.D. Dissertation

Nonlinear Operators for Image Processing: Design, Implementation and Modeling Techniques for Power Estimation

Advisor: Prof. Giovanni L. Sicuranza,
University of Trieste.

Coordinator: Prof. Giuseppe O. Longo,
University of Trieste.

Ph.D. Candidate: Giuseppe Bernacchia.

Acknowledgements

I feel greatly indebted to many people at the Image Processing Laboratory at the University of Trieste for their support and help for my work there.

First of all I must thank my academic advisor, prof. Giovanni Sicuranza, who provided great advices and encouragement throughout my research activity.

I am also grateful to Eng. Sergio Carrato and Dott. Stefano Marsi, who, as friends, greately helped me with suggestions. Last but not least, I must thank prof. Giovanni Ramponi, whose work in the field of rational functions constitues the basis for this thesis.

A special thank goes to prof. Marios Papaefthymiou, at the Advanced Computer Architecture Laboratory, University of Michigan, USA. He greately encouraged and supported me during the period I spent at the University of Michigan. The stimuli he provided me led us to very good results and satisfaction about our work.

My appreciation is extended to prof. Moncef Gabbouj, at the University of Technology of Tampere, Finland, who also helped me with the study of rational functions and nonlinear operators.

A special thank to all my friends at the Image Processing Laboratory of Trieste, for the good time we spent together.

Finally, I want to express my thanks to my family and, most of all, to my girlfriend Cristina, for their constant love and support.

To Cristina

Contents

Introduction

In the past few years, multimedia application have been growing very fast, being applied to a large variety of fields. Applications like video conference, medical diagnostic, mobile phones, military applications require to handle large amount of data at high rate. Images as well as voice data processing are therefore very important and they have been subjected to a lot of efforts in order to find always faster and effective algorithms.

Among image processing algorithms, we believe that rational operators assume an important role, due to their versatility and effectiveness in data processing. In the last years, several algorithms have been proposed, demonstrating that these operators can be very suitable in different applications with very good results.

The aim of this work is to implement some of these algorithm and, therefore, demonstrate that rational filters, in particular, can be implemented without requiring large sized systems and they can operate at very high frequencies.

Once the basic building block of a rational based system has been implemented, it can be successfully reused in many other applications.

From the designer point of view, it is important to have a general framework, which makes it able to study various configurations of the system to be implemented and analyse the trade-off among the design variables.

In particular, to meet the need for versatile tools for power estimation, we developed a new macro modelling technique, which allows the designer to estimate the power dissipated by a circuit quickly and accurately.

The thesis is organized as follows:

In chapter 1 we present some of the algorithms which have been studied for implementation. Only a brief overview is given, leaving to the interested reader some references in literature.

In chapter 2 we discuss the basic architectures used for the implementations. Pipelined structures have been mainly used for this thesis, but an overview of the nowaday available approaches for timing optimization is presented.

In chapter 3 we present two of the implementation designed for this thesis. The approaches followed are ASIC driven and FPGA drive. They require different techniques and different solution for the design of the system, therefore it is interesting to see what can be done in both the cases.

Finally, in chapter 4, we describe our macromodelling techniques for power estimation, giving a brief overview of the upto now proposed techniques and showing the advantages our method brings to the design.

Introduzione

Negli ultimi anni passati le applicazioni multimediali hanno visto uno sviluppo notevole, trovando applicazione in un gran numero di campi. Applicazioni come video conferenze, diagnostica medica, telefonia mobile e applicazioni militari necessitano il trattamento di una gran mole di dati ad alta velocitá. Pertanto, l'elaborazione di immagini e di dati vocali é molto importante ed é stata oggetto di numerosi sforzi, nel tentativo di trovare algoritmi sempre piú veloci ed efficai.

Tra gli algoritmi proposti, noi crediamo che gli operatori razionali svolgano un ruolo molto importante, grazie alla loro versatilitá ed efficacia nell'elaborazione di dati.

Negli ultimi anni sono stati proposti diversi algoritmi, dimostrando che questio operatori possono essere molto vantaggiosi in diverse applicazioni, producendo buoni risultati.

Lo scopo di questo lavoro é di realizzare alcuni di questi algoritmi e, quindi, dimostrare che i filtri razionali, in particolare, possono essere realizzati senza ricorrere a sistemi di grandi dimensioni e possono raggiungere frequenze operative molto alte.

Una volta che il blocco fondamentale di un sistema basato su operatori razionali sia stato realizzato, esso puó essere riusato con successo in molte altre applicazioni.

Dal punto di vista del progettista, é importante avere uno schema generale di studio, che lo renda capace di studiare le varie configurazioni del sistema da realizzare e di analizzare i compromessi tra le variabili di progetto.

In particolare, per soddisfare l'esigenza di metodi versatili per la stima della potenza, abbiamo sviluppato una tecnica di macro model-

lizazione che permette al progettista di stimare velocemente ed accuratamente la potenza dissipata da un circuito.

La tesi é organizzata come segue:

Nel Capitolo 1 alcuni sono presentati alcuni algoritmi studiati per la realizzazione. Ne viene data solo una veloce descrizione, lasciando comunque al lettore interessato dei riferimenti bibliografici.

Nel Capitolo 2 vengono discusse le architetture fondamentali usate per la realizzazione. Principalmente sono state usate architetture a pipeline, ma viene data anche una descrizione degli approcci oggigiorno disponibili per l'ottimizzazione delle temporizzazioni.

Nel Capitolo 3 sono presentate le realizzazioni di due sistemi studiati per questa tesi. Gli approci seguiti si basano su ASIC e FPGA. Richiedono tecniche e soluzioni diverse per il progetto del sistema, per cui é interessante vedere cosa puó essere fatto nei due casi.

Infine, nel Capitolo 4, descriviamo la nostra tecnica di macro modellizazione per la stima di potenza, dando una breve visione delle tecniche finora proposte e facendo vedere quali sono i vantaggi che il nostro metodo comporta per il progetto.

Chapter 1

Rational Operators

Introduction

In the recent years there has been an increasing interest in signal processing techniques due to the wide spreading of multimedia applications. In particular, lot of efforts have been spent studying image and voice data handling algorithms. Focusing our attention of image processing, the developed techniques cover a large field of application, like video conference, digital image transmission, image handling for photography applications, medical diagnostic, video walls, office automation, robotic, military applications, etc.. All of them require several different algorithms to process the data, involving interpolation, enhancement, noise reduction, decimation, etc.. The solutions proposed for each algorithm are numerous and their effectiveness often depends on the specific application. Starting from simple linear function, the range of nowaday disposable operator has been widely extended.

Among the various nonlinear operators, polynomial and, most of all, rational operators demonstrate to often outperform other nonlinear operators in operations like interpolation and filtering. Moreover, their analytical representation makes them very suitable for modular hardware implementation. As pointed out in [105], rational functions exhibit the nice feature of error correction. If both numerator and denominator of the rational function are evaluated with reduct precision

coefficients and/or operations, the result is not affected by the same level of degradation but still shows a high level of precision. This behaviour has been exploited in old calculators to obtain good approximations of basic functions with reduced operands word length.

The design of rational operators is usually based on a weighted combination of nonlinear filters with lowpass or highpass behaviour, the coefficients of which are heuristically chosen. Even though several rational operators have been proposed in the last few years, a general framework for their analysis and design has not been developed. Some hints about the constraints a rational operator should respect on order to achieve determined features are presented in [31].

In the following we present a brief review of the algorithms analyzed and studied for implementation. We suppose the reader is confident with the basic concepts and algorithm regarding image processing.

1.1 Rational Filter for Image Smoothing

One of the most important algorithms for image processing is reported in [41]. It is an operator for image smoothing, i.e. a filter for noise reduction.

The linear filters offer the advantage of simplicity, but their results are not very satisfactory. In particular, they produce an unacceptable bluring phenomenon on the image, with loss of details.

The basic idea underlying the rational operator is to modulate the coefficient of a linear filter in order to limit its action when details are present.

The generic 1-D linear operator acting on a length 3 mask can be represented by the following expression:

$$y_n = w(x_{n-1} + x_{n+1}) + (1 - 2w)x_n \tag{1.1}$$

where $[x_{n-1}, x_n, x_{n+1}]$ is the input sequence and y_n is the output sample. It is characterized by a lowpass behaviour if the parameter w assumes values in the range $0 < w \leq \frac{1}{3}$.

In order to modify the behaviour of this operator in presence of details, a function indicating relevant changes in the signal is introduced.

In particular, this can be achieved in an effective way by the squared difference of the two extreme samples, $xn-1, x_{n+1}$. If it is large,it is assumed that the mask is positioned across a signal transition, therefore the filter response is made less selective. On the other hand, if the squared difference is small, it is supposed that fluctuation in the signal due to noise are present and the selectiveness of the linear filter is enhanced.

This approach yields the following rational operator:

$$y_n = x_n + w\frac{x_{n-1} + x_{n+1} - 2x_n}{wk(x_{n-1} - x_{n+1})^2 + 1} \tag{1.2}$$

The parameter k should take positive values and is used to control the filter. If $k \simeq 0$, the Eq.1.2 reduces to Eq.1.1. If $k \to \infty$, the filter has no effect and $y_n \simeq x_n$. For intermediate values of k, the squared difference $(x_{n-1} - x_{n+1})^2$ perceives the presence of details and reduces the smoothing effect of the operator.

The filtering with the rational operator can be iterated several times. If p passes are performed, the smoothing effect on uniform areas is equivalent to the result obtained with a lowpass linear filter having size $(2p+1)$.

A practical and effective way to extend the proposed method to 2-D data is to apply it in a 3x3 mask:

$$\begin{pmatrix} x_{m-1,n-1} & x_{m-1,n} & x_{m-1,n+1} \\ x_{m,n-1} & x_{m,n} & x_{m-1,n+1} \\ x_{m+1,n-1} & x_{m+1,n} & x_{m-1,+1} \end{pmatrix}$$

Thus the following operator is obtained:

$$y_{m,n} = x_{m,n} + w\frac{x_{m-1,n}+x_{m+1,n}-2x_{m,n}}{wk(x_{m-1,n}-x_{m+1,n})^2+1} \tag{1.3}$$

$$+ w\frac{x_{m,n-1}+x_{m,n+1}-2x_{m,n}}{wk(x_{m,n-1}-x_{m,n+1})^2+1} \tag{1.4}$$

$$+ w\frac{x_{m-1,n-1}+x_{m+1,n+1}-2x_{m,n}}{wk(x_{m-1,n-1}-x_{m+1,n+1})^2+\sqrt{2}} \tag{1.5}$$

$$+ w\frac{x_{m-1,n+1}+x_{m+1,n-1}-2x_{m,n}}{wk(x_{m-1,n+1}-x_{m+1,n-1})^2+\sqrt{2}} \tag{1.6}$$

In [41], results of comparison with different filtering techniques are reported. The rational operator outperforms other algorithms in all the presented cases and under different noise conditions.

This basic operator constitutes the basis for a whole family of image processing algorithms, as will be seen in the following.

1.2 Median-Rational Hybrid Filter - MRHF

This algorithms has been presented in [46]. In this paper the author tried to avoid the poor performances of the rational filter adding a preliminary stage composed by several median filters. It is well known that median filters work very well when impulsive noise is superimposed to the image. On the other hand, rational operators are very well suited for Gaussian noise. The combination of the two filtering techniques results to be very powerful with Gaussian-impulsive combined noise.

The MRHF works on a 3x3 support mask. It is composed by three input median filters, Φ_1, Φ_2, Φ_3. Φ_2 is always a center weighted median filter acting on a plus-shaped mask:

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 0 & x_{m-1,n} & 0 \\ x_{m,n-1} & 3 \times x_{m,n} & x_{m-1,n+1} \\ 0 & x_{m+1,n} & 0 \end{pmatrix}$$

This operator find the median among the values in the mask, repeating three times the central term:

$\Phi_2 = med\{x_{m-1,n}, x_{m,n-1}, x_{m,n}, x_{m,n}, x_{m,n}, x_{m-1,n+1}, x_{m+1,n}\}$

In this way, the operator outputs the actual central term in the mask, unless it is the lowest or the highest value.

The other two median filters can be application-specified, i.e. they can be chosen so that the filtering capability of the global operator is optimized. In the cases reported in [46], these two filters act on a plus-shaped and on a cross-shaped mask respectively:

$$\Phi_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \Phi_3 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

The second stage of the filter is a rational operator having a structure similar to Eq.1.2:

$$y_n = \Phi_2 + \frac{\sum_{i=1}^{3} \alpha_i \Phi_i}{h + k(\Phi_1 - \Phi_3)^2} \tag{1.7}$$

where the coefficients $\alpha = [\alpha_1, \alpha_2, \alpha_3]$ satisfy the condition $\sum_{i=1}^{3} \alpha_i = 0$. In this particular case $\alpha = [1, -2, 1]$, so that the expression in Eq.1.2 is obtained. Th values of h and k are taken from [41], and are 6.25 and 0.01 respectively.

The results reported in [46] show that the MRHF outperforms the median filters when impulsive noise is present and is superior to the rational filter in the case of Gaussian and contaminated Gaussian.

1.3 Vector Median Rational Hybrid Filter - VMRHF

The hybrid filter previously presented has been extended in [47] to be able to operate on colour images. Traditional approaches solve the problem of multichannel filtering considering each channel separately. However, the correlation existing among the channels suggests that non-separable algorithms would have better performances.

In vector approaches, each pixel is an m-dimensional vector. In the case of colour images m= 3; the vector direction signifies their chromaticity, while the vector module is related to the brightness.

Let $\underline{f}$ be a multichannel signal and x_i the i-th pixel in the image, so that $\underline{f}(x_i) = \underline{f}_i$ is the value assumed by the function $\underline{f}$ in x_i. Te VMRHF is defined by the following expression:

$$\underline{y}(\underline{f}_i) = \underline{\Phi}_2(\underline{f}_i) + \frac{\sum_{j=1}^{3} \alpha_j \underline{\Phi}_j(\underline{f}_i)}{h + k||\underline{\Phi}_1(\underline{f}_i) - \underline{\Phi}_3(\underline{f}_i)||_2} \tag{1.8}$$

where $||\cdot||_2$ is the L_2 norm and $\alpha = [\alpha_1, \alpha_2, \alpha_3]$, as in the previous case, are chosen as $\alpha = [1, -2, 1]$. It is important to notice that even if the numerator maintains the separable structure of a lowpass

filter, the weighting denominator depends on all the three image vector components, therefore the algorithm becomes non-separable.

The functions $\Phi_{1,3}$ are vector median filters acting on a plus-shaped and on a cross-shaped mask respectively, while Φ_2 is a vector center weighted median filter with plus-shaped support mask. In [47], the VMRHF has been compared with the vector median filter and with the directional-distance filter, which are two widely known filter for multichannel signal analysis. The VMRHF shows outperforming feature over all the range of noise used for the tests.

1.4 Rational Interpolator

The rational operators are very suitable to be used in interpolation problems, due to their nonlinearity which makes them very sensitive to details in the image.

The key observation of the algorithm presented in [36] is that any real world image can be considered as obtained from a higher resolution image after lowpass filtering and decimation, the anti-aliasing lowpass filtering been accomplished explicitly or by the acquisition system.

When an ideal edge is present, this filtering operation symmetrically or asymmetrically modified the value of adjacent pixels according to the position of the very edge in the high resolution image. After decimation, an analysis of the pixels values in the low resolution image gives sub-pixel information on the position of the edge. Therefore, the interpolation itself can be more precise than in the linear interpolator case.

Let consider first a 1-D interpolator and suppose the supporting mask has length 4, that is, 4 pixels are considered for the reconstruction of the central value: $[a\,b\,c\,d] \rightarrow x$, with $b < x < c$. The linear interpolator would calculate x as the mean of b and c: $x = \frac{b-c}{2}$. As in the case of the rational noise-smoothing filter, a weighting term can be included in the linear operator in order to detect the presence of details.

The 1-D operator can be expressed by the following equations:

$$x = \quad \mu b * (1-\mu)c \tag{1.9}$$

$$\mu = \frac{k(c-d)^2+1}{k((a-b)^2+(c-d)^2)+2} \tag{1.10}$$

where,again, k controls the amount of nonlinearity introduced by the rational function.

Since this operator is noonlinear, it is able to introduce frequencies between $\pi/2$ and π, which has been lost during the lowpass/decimation steps.

The extension to the two-dimensional case is not as straightforward as it could appear. First of all, the 1-D operator is applied twice, once in the vertical and once in the horizontal direction. After this step, the horizontal and vertical edges in the image are detected and partially reconstructed. In the following picture, the o represent original values, $\cdot$ represent unknown values, while $\oplus$ are reconstructed values. The picture shows the pixels obtained after the 1-D operator has been applied in the vertical and horizontal directions.

$$\begin{bmatrix} o & \cdot & o & \cdot & o & \cdot & o \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ o & \cdot & o & \cdot & o & \cdot & o \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ o & \cdot & o & \cdot & o & \cdot & o \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ o & \cdot & o & \cdot & o & \cdot & o \end{bmatrix} \rightarrow \begin{bmatrix} o & \oplus & o & \oplus & o & \oplus & o \\ \oplus & \cdot & \oplus & \cdot & \oplus & \cdot & \oplus \\ o & \oplus & o & \oplus & o & \oplus & o \\ \oplus & \cdot & \oplus & \cdot & \oplus & \cdot & \oplus \\ o & \oplus & o & \oplus & o & \oplus & o \\ \oplus & \cdot & \oplus & \cdot & \oplus & \cdot & \oplus \\ o & \oplus & o & \oplus & o & \oplus & o \end{bmatrix}$$

For the evaluation of the missing pixels, two approaches can be used. The first requires a further application of the 1-D operator on both vertical and horizontal directions for each missing pixel. The final value is obtained as the average of the two results. This method presents the disadvantage to produce jagged diagonal borders, which is not very nice from a subjective point of view.

A non-separable operator has been introduced to solve the problem in [40]. After the first phase has been terminated, a 3x3 mask centered on the missing pixel is considered:

$$\begin{bmatrix} a & e & b \\ h & x & f \\ d & g & c \end{bmatrix}$$

The operator has the following expression:

$$x = \frac{w_{ac}(a+c) + w_{bd}(b+d) + w_{eg}(e+g) + w_{fh}(f+h)}{2(w_{ab} + w_{bd} + w_{eg} + w_{fh})} \tag{1.11}$$

with

$$w_{ac} = \frac{1}{1+k(a-c)^2} \tag{1.12}$$

$$w_{bd} = \frac{1}{1+k(b-d)^2} \tag{1.13}$$

$$w_{eg} = \frac{1}{1+k(e-g)^2} \tag{1.14}$$

$$w_{fh} = \frac{1}{1+k(f-h)^2} \tag{1.15}$$

The results proposed in [40] show that this solution allows to obtain better results than the cubic operator.

1.5 Artifact Removing Filter

The last algorithm we present is the artifact removing rational operator used for MPEG2 coded images. In MPEG process, square pixels blocks are coded without considering its boundary. Therefore, a noisy blocking effect shows up in the reconstructed image. For instance, the decoded image appears as a puzzle of square blocks, even if they are not present in the original image.

The artifact could be removed with a simple lowpass filter, however this causes a blur also of the details which should be preserved.

In [35], a simple filtering algorithm is proposed, based on the results presented in [41]. The operator acts on a 3x3 mask and, referring to 1.1, has the following expression:

$$y_{m,n} = x_{m,n} + w\frac{x_{m-1,n}+x_{m+1,n}-2x_{m,n}}{wk(x_{m-1,n}-x_{m+1,n})^4+1} \tag{1.16}$$

$$+w\frac{x_{m-1,n-1}+x_{m+1,n+1}-2x_{m,n}}{wk(x_{m-1,n-1}-x_{m+1,n+1})^4+1} \tag{1.17}$$

$$+w\frac{x_{m-1,n+1}+x_{m+1,n-1}-2x_{m,n}}{wk(x_{m-1,n+1}-x_{m+1,n-1})^4+1} \tag{1.18}$$

The operator has to be applied twice: once to remove the vertical edges and once to remove the horizontal ones. In each case, the pixels lying close to the edge and across the central pixel are not considered in the filtering

The complexity of the operator can be reduced considering the square function instead of the 4-th power as in Eq.artifact.

1.6 Conclusion

In this chapter we briefly presented the rational operators and the derived filters, which have been studied to be implemented. As can be seen form the equations describing each of the operators, they are quite similar and, therefore, they present similar problems for the implementation. However, their similarity is also their power, since with almost the same function several different operator can be obtained.

This interesting feature has been exploited to design a reprogrammable filter, as will be further explained in Chapter 3.

Chapter 2

Architectures

High-performance, special purpose computer systems are typically used to meet specific application requirements or to off-load computations that are especially taxing to general-purpose computers. As hardware cost and size continue to drop, special-purpose systems are more and more developed for areas like image or signal processing.

Cost-effectiveness has always been a concern in designing special-purpose systems; the cost should be low enough to justify their limited applicability.

Fortunately , special-purpose design costs can be reduced using an appropriate architecture. If a structure can truly be decomposed into few types of simple substructures or building blocks, great saving can be achieved.

Beside the design costs, a fundamental role is played by the speed of the circuit. In most of the image processing or signal application, the system should be able to perform millions of operations per second. One way to increase the speed is to reduce the size of transistors, but this requires a technological change, which is not always possible. Moreover, such changes are very expensive. The other useful technique available for performance increase is concurrency. The degree of concurrency, or operation parallelism, in a system is basically determined by the underlying algorithm. Massive parallelism can be achieved is designed to introduce high degree of pipelining and multiprocessing.

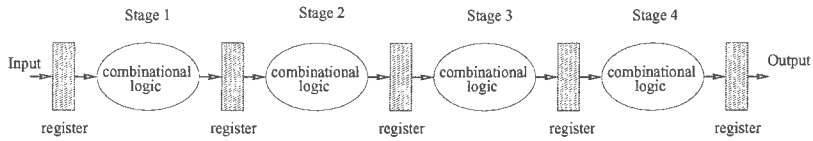

Figure 2.1: A pipeline system.

A pipeline circuit (see Fig.2.1)is a synchronous digital circuit constructed partitioning the combinational logic into stages and inserting a set of memory elements, generally called *internal register* or simply *register*, to provide temporary storage for the computed data between successive pipelined stages. This temporary storage is enabled and disabled in a carefully controlled periodic manner, thereby synchronizing the data signals to prevent any corruption of data due to interference between successive waves of data travelling trough the pipeline.

Performance optimization of such synchronous systems requires accurate timing analysis of the design. The process is usually iterative and different optimization techniques are needed for different design levels.

On the pipeline boundary, input and output synchronizers are needed to store the pipeline inputs and outputs, respectively. If the input and output synchronizers are distinct, we call the pipeline an *open* pipeline. If the input and output registers are actually implemented by the same register, resulting in a pipeline with a ring topology, we call the pipeline a *closed* pipeline.

Two types of synchronizers, either edge-triggered flip-flops or level-sensitive latches, are used in pipeline circuits.

To achieve maximum performances in a pipeline system, all the pipeline stages must perform useful work all the time. In this ideal case the pipeline logic must be equally divided among the stages so that computations in all stages finish at exactly the same time. However, due to practical constraints or improper design of the given pipeline, one stage may require more time to compute than the other stages. The performance will decrease, since standard clocking schemes are limited by the compute time of the slowest stage.

Another factor limiting the performances is the I/O bandwidth.

Usually, a special-purpose system is designed to operate at very high speed. Often its operating frequency is much higher than that of the attached host, where host means a computer, a memory , a real time device, etc.. Since the host supplies the input data, its bandwidth strongly influences the performance of the special-purpose circuit. As an example, suppose that the I/O bandwidth is 10 millions byte per second; assuming the system needs to read 2 bytes to be able to perform its operation, the maximum rate will be only 5 millions operations per second, no matter how fast the system can operate.

A huge improvement can be achieved if multiple computations are performed per I/O access. The drawback of this approach is the need of memory elements for storing the data inside the system. Thus the question is how to arrange a computation together with an appropriate memory structure so that computation time is balanced with I/O time.

Two design techniques have been proposed to solve the above described problems for synchronous systems, starting from the basic pipeline architecture: *systolic architectures* ([55]), and *wave pipelining*. In the last few years asynchronous design phase come to a new life, thanks to the efforts spent trying to avoid or limit the problems of synchronous systems.

In the following we present a brief review of these three techniques.

2.0.1 Systolic Architectures

A systolic system consists of a set of interconnected cells, each capable of performing some simple operation. Because simple, regular communication and control structures have substantial advantages over complicated ones in design and implementation, cells in a systolic system are tipically interconnected to form a systolic array or a systolic tree.

Information in a systolic system flows between cells in a pipelined fashion, and communication with the outside world occurs only at the boundary cells. The local-communication protocol exploited in this way allows a higher operating frequency, since long wires (i.e. huge capacitances) are avoided. In a computation, if the total number of operations is larger than the total number of input and output elements, then the computation is *compute-bound*, otherwise it is I/O-bound.

The basic property of systolic systems is the multiple use of each input data, therefore a high throughput can be achieved even with modest I/O bandwidths. To meet this criterion, global data communications, such broadcast or unbounded fan-in, or local data communication are used. The latter is usually preferable for modularity sake. In this case the input data travels through the array of cells in sequence, so that each cell can use it. The problem with global communication is the need for long wire, which increases power dissipation and/or decreases the global speed.

The processing power of a systolic system comes from the concurrent use of all its constituting cells. This could be done either pipelining the stages involved in the computation of each single result, or multiprocessing many results in parallel or with a combination of both. See [55] for a description of various architectures.

Examples of algorithms easily implementable with systolic arrays are the convolution of two sequences, polynomial multiplication [70][56], matrix-matrix multiplication, matrix-vector multiplication, matrix triangularization, FFT computation, etc..

2.0.2 Wave-pipelining

Although a perfectly balanced pipeline is hard or even impossible to design in practice, aggressive clocking techniques can be used to improve performances (see [62]). These techniques make an intensive use of level-sensitive latches to improve the features of the pipeline system.

A level-sensitive latch is a synchronizing device which is sensitive to the level of the clock signal. When its level is high, a *positive* latch is in its enabled state. This state allows the data at the output to assume the value of the data at the input, providing a direct propagation path trough the latch. When the clock level is low, the positive level-sensitive latch is disabled and will hold its output value constant, regardless of changes to the data input. The *negative* level-sensitive latch has a reverse operation mode.

The data value stored in the latch when it is disabled must be stable at the input for some time before and after the transition from the enabled state to the disabled state. The minimum input stable time

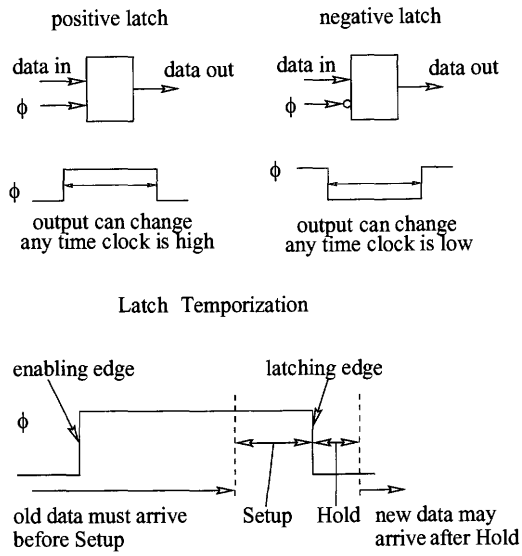

Figure 2.2: Level-sensitive latches.

before the transition is called *setup*, while the minimum stable time after the transition is called *hold*.

Circuits with level-sensitive latches can allow the signals to flow through the latches unimpeded, allowing the computation time to be "*borrowed*" from preceding pipeline stages. Borrowing can allow a reduction in the cycle time of the clock, thereby increasing the circuit performance.

Borrowing is not possible when flip-flops are used instead of latches. Synchronous circuits with edge triggered flip-flops require that all flip-flop input signals be fully stabilized before enabling the clock edge. The propagation into the next stage can only start after the clock edge.

Since the clock minimum period is determined by the longest delay path, all the paths having shorter length are characterized by some idle time, i.e. they are inactive for a fraction of the useful period.

Although there are performance advantages for circuits with level-sensitive latches, these circuits are difficult to analyze accurately, since the data can depart from a latch at any time during the active interval.

This *flow-trough* property generally leads to complex equations for the timing analysis of such circuits. These timings cannot be determined by analyzing only a single stage of the pipeline, but the parameters of all prior stages are required.

Usually overly conservative *worst case* assumptions about data arrival and departure times are used when analyzing a latch-controlled synchronous circuit.

Research on latch-controlled synchronous circuits basically involves three areas:

1. The timing verification problem, which determines whether the circuit can operate correctly with a prespecified clock schedule.

2. The optimal clocking problem, which determines the clock schedule achieving the maximum performance of the circuit.

3. The timing-drive optimization problem, which attempts to improve some aspects of a design while maintaining or improving its performance. This problem covers a wide range of optimization techniques, including:

    - *retiming*: repositioning the latches in the circuit to reduce the clock period;
    - *logic resynthesis*: restructuring the combinational logic of two ore more pipeline stages and then retiming;
    - *transistor sizing*:altering the size of transistors to reduce critical delays;
    - *part selection*: selecting a set of already optimized circuits from a library to meet pre-specified performances;
    - *gate reordering*: reordering the gate input assignments to reduce path delays when gate inputs are logically symmetric but input to output delays vary;
    - *placement and routing*: changing the placement and routing of critical interconnects.

An aggressive design technique used to improve the actual throughput of a pipeline system is *wave pipelining.*

In ordinary pipeline circuit with a single phase clock, an average of one data wave at a time resides in the combinational circuit between a pair of successive registers. When a new value is clocked into the input register of a stage, the value is allowed to propagate to the output register before the input register can accept a new value.

In contrast, wave pipelining allows multiple data waves to propagate simultaneously through the combinational circuit between two consecutive registers. This is achieved using a clock period that is less than the maximum propagation delay between registers. With wave pipelining, the inherent temporary storage capability provided by the capacitances in the devices and interconnections is exploited to store the data wave values temporarily and thereby reduce the number of registers.

The minimum cycle time of a wave-pipelined circuit is limited only by the difference between the maximum and minimum delays through the logic and by the basic device switching speed, not by the actual maximum delay of the circuit.

In order to obtain the highest possible wave-pipelining clocking rate, all path delays from input to output registers must be equalized.

On the other hand, there are some problems concerning the design of a wave-pipeline which are not present in an ordinary pipeline system. For instance, for an ordinary pipeline the clock frequency can be slowed down arbitrarily; however, once a circuit has been wave-pipelined, it operating speed is constrained to a narrow range of frequencies for specified degree of wave-pipelining. As this degree is increased for higher performances, the range of allowed clocking speed becomes narrower. This condition imposes strict constraints on other system components interfacing the wave-pipeline system. Moreover, since the wave-pipeline is very sensitive to path delay variations, stricter regulations of environmental and fabrication variations must be used.

An other concern regarding wave-pipelines is the unavailability of an effective mechanism for stopping and starting them. Such mechanisms are needed to handle pipeline stalls and low-speed testing.

2.0.3 Asynchronous Architectures

In the traditional synchronous clocking scheme all the system components are synchronized to a global clock signal. As systems continue to grow in size and, most of all, in complexity, the difficulty of coordinating the activity of all its components becomes critical. At chip level, the main source of power dissipation often is represented by the clock buffers used to distribute the synchronizing signal to all the sub-circuits. Moreover the complexity in the clock distributing net increases the probability of clock skews, with possible disfunctioning of the system.

Asynchronous circuits provide a different approach. In an asynchronous systems, events are restricted to a particular *sequence*, without concerning the time at which these events occur. For correct functioning, it is important that the correct order of events be maintained within the circuit.

Among the possible asynchronous architectures, *self-timed* circuits have been quite well studied because of their interesting features. Instead of letting the signals flow through the circuit whenever they are able (unstructured asynchronous systems), self-timed circuits enforce a simple communication protocol between circuit elements.

This communication protocol is often defined in terms of a pair of signals that require an action or acknowledge that the requested action has been completed. One module, the *sender*, request an event to another module, the *receiver*. Once the receiver has completed the requested operation, it send an *acknowledge* event back to the sender to complete the transaction (see Fig.2.3).

A common asynchronous protocol is the *bundle data* protocol. It uses two-phase transition signaling for control, i.e. it uses transitions on signal wires to communicate the request and acknowledge events between circuit modules. Only the transitions are meaningful; however, a transition from low to high is the same as a transition from high to low and the particular state of each wire is not important.

A bundled data path uses a single se of control wires to indicate the validity of a *bundle* of data wires. This requires that the data bundle and the control wires be constructed such that the value on the data bundle is stable at the receiver before the signal appears on the control

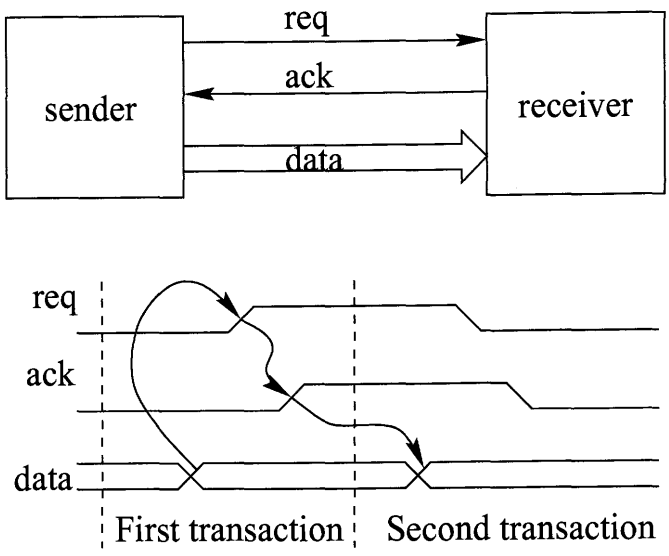

Figure 2.3: Asynchronous protocol.

wire.

Control circuits for transition signalling may be built from the set of simple building blocks shown in Fig.2.4. The *XOR* gate provides the merge function for two transition signals, since a transition event on either the first input *OR* the second input will produce a transition on the output. For correct functioning the two inputs may not arrive so close that the output transitions cannot be distinguished. Therefore they must be mutually exclusive.

The *Muller C-element* provides the logical *and* function for transition events.It is usually represented with an AND gate with a C inside. This is not a simple combinational gate, but includes some internal states. When both inputs are at the same level, the output is also at that level, but a transition occurs only after both inputs have changed.

The *toggle* element alternates incoming transitions between two inputs. The *select* module is similar to the toggle element, but instead of simply alternating, the output is chosen according to the level of a second input when the transition arrives.

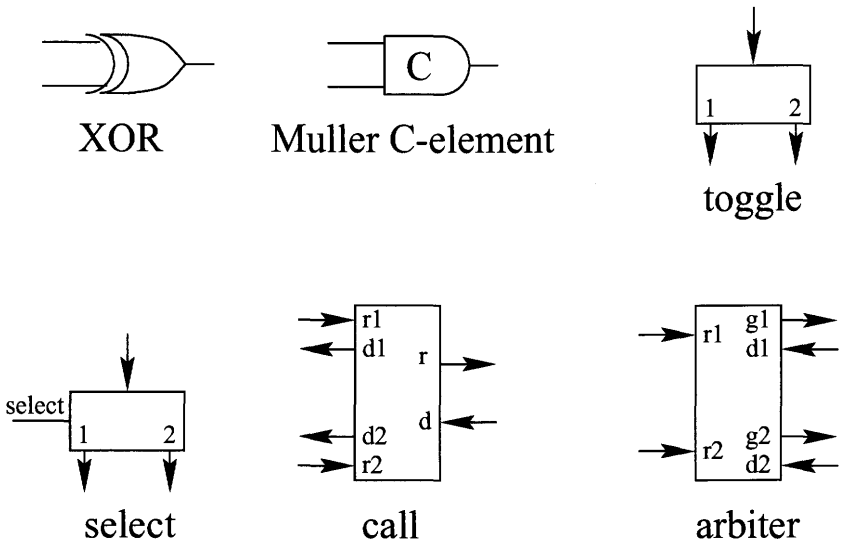

Figure 2.4: Asynchronous system building blocks.

The *call* module allows two self-timed modules to share the resources of a third module.

The *arbiter* module is used to provide mutual exclusion between two unrelated systems.

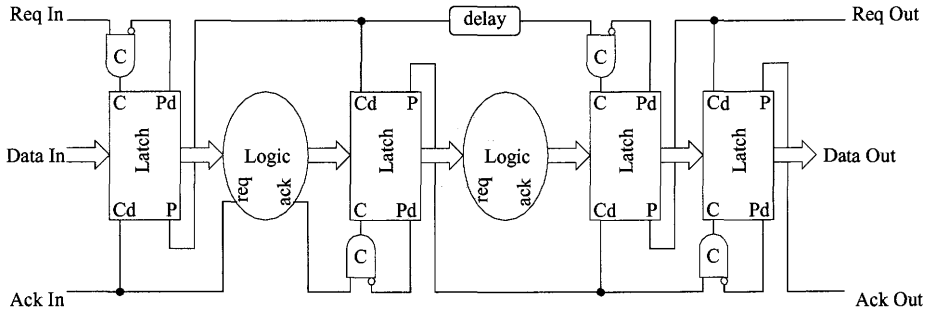

Figure 2.5: Micropipeline structure.

To complete the construction of a self-timed system, some storage elements are required. In particular, using the *bundled protocol*, a tran-

sition controlled storage element has to be used. However, such modules are usually complex, slow or bulky, therefore this kind of architecture is mostly used with dynamic logic gates, which incorporates the memory element in the very structure of the gate.

The two-phase bundled protocol offers a number of advantages in its simplicity and composability. In [71] an elegant methodology for constructing self-timed systems is described (see Fig.2.5).

It consists of three parts: a control network consisting of a C-element per stage, an event-controlled latch in each stage, some combinational logic between two successive stages, if needed. The combinational logic can signal the completion of its operations by itself or a delay can be used. In this latter case the time delay long enough to ensure a correct functioning for the worst case path.

If no combinational logic is inserted, the micropipeline works as simple asynchronous FIFO.

Suppose the state of the leftmost C-element is low. When a *req-in* transition occurs at its input, the element will produce a transition in the latch. The acknowledge from the latch produces a similar transition in the first C-element to the right. Meanwhile, the C-element in the first stage will not produce another request to the latch until there are transitions on both *req-in* (signaling that there are more data to be processed) and *Cd*, Completion done, in the next stage, signaling that the next stage has finished its operations.

In [61] a new asynchronous scheme based on the micropipeline structure is presented. It exploits the features of the Differential Cascode Voltage Switch Logic (DCVSL). In this case the registers are included in the gate structure, therefore a very compact system can be realized.

The asynchronous approach is very suitable for image processing, because it easily adapts to the input data frequency. In this way the operations are always executed at the maximum speed, allowing either a reduction in power consumption, since the modules do not work in the idel time, or a resources sharing, if the same modules are used to process data belonging to different streams. In [63],[64],[65],[68],[59] several implementation of asynchronous systems are proposed. [58] contains a good overview of the state of the art for asynchronous systems. The

main drawback of this approach is that the design procedures are quite complex, due to timing constraints and fault tolerance (see [66],[69]).

Chapter 3

Implementations

In this chapter we propose and analize the implementation of several image processing algorithms. Two main approaches have been followed for these designs:

- implementation of an ASIC (Application Specific Integrated Circuit)
- implementation on an FPGA (Field Programmable Gate Array).

The design philosophy underlying these approaches is very different as they respond to different needs of the designer.

The design of an ASIC allows a greater flexibility and freedom, since the designer can choose the technology, the architecture and circuit solutions that better satisfy the constraints. Nevertheless this approach is very time-demanding and expensive and is more sensible to design mistakes and technology variations.

On the other side the FPGA approach limits the possibilities of the designer, since they allow only fixed dimensions (i.e. fixed number of memory elements and combinational blocks), usually they are not very suited for completely user specified structures, while they performs very well if the built-in operators are exploited. But FPGA offer the significant advantage that they are cheap and can be easily reprogrammed, therefore design errors can be corrected.

The choice of the particular solution obviously depends on the applications for which a circuit has been designed. In order to investigate the effectiveness, robustness and flexibility of some of the algorithms we implemented both the approaches.

3.1 A Reprogrammable Rational Operator

The first two operators act on a 3x3 mask, as will be shown further on. The interpolator requires a 1-D window of length 4, therefore it introduces a slight complication in the final structure of the reconfigurable filter.

The first operator can be represented by the following expression (see Fig.3.1 a)):

$$e_n = e + \frac{w(a+i-2e)}{wk(a-i)^2+1} + \frac{w(c+g-2e)}{wk(c-g)^2+1} + \frac{w(b+h-2e)}{wk(b-h)^2+\sqrt{2}} + \frac{w(d+f-2e)}{wk(d-f)^2+\sqrt{2}}$$

but it can be better rewritten as follows:

$$e_n = e + \alpha\frac{a+i-2e}{(a-i)^2+\beta} + \alpha\frac{c+g-2e}{(c-g)^2+\beta} + \alpha\frac{b+h-2e}{(b-h)^2+\beta^1} + \alpha\frac{d+f-2e}{(d-f)^2+\beta^1} \tag{3.1}$$

with $\alpha = 1/k$, $\beta = 1/wk$ and $\beta^1 = \sqrt{2}/wk$. The parameter k should assume positive values and it is used to control the filter. If $k = 0$ then we obtain a linear filter, if $k \rightarrow \infty$ then the filter has no effects and $e_n \simeq e$.

The second algorithm is used to remove the artifacts introduced by the MPEG compression. In this case the image appears like a mosaic because the compressor-coder acts separately on one block at a time; if two adjacent blocks belong to areas with different luminance the effect of the compression is to enhance this difference and therefore a border line between the blocks will appear. Since the noise appears like evidently false borders along the horizontal and vertical directions, this operator

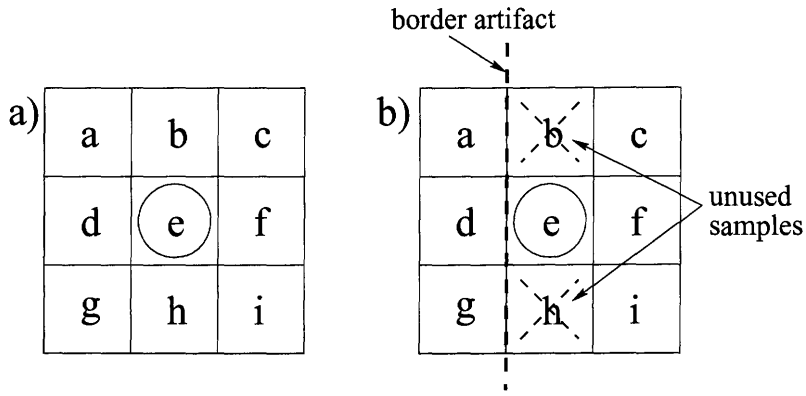

Figure 3.1: Masks for noise-smoothing filter (a)) and for artifacts removal (b))

has to be applied twice, once per direction. Each time, the neighbours along the operation direction of the mask's central pixel are removed, as in the example of Fig.3.1 b).

The function associated with this algorithm is:

$$e_n = e + \alpha\frac{a+i-2e}{(a-i)^2+\beta} + \alpha\frac{c+g-2e}{(c-g)^2+\beta} + \alpha\frac{d+f-2e}{(d-f)^2+\beta} \tag{3.2}$$

with $\alpha = 1/2k$, $\beta = 1/wk$.

The last algorithm is a rational interpolator. The structure of this operator is quite complex as is its application to the image. In fact it requires two steps in order to output all the interpolated pixels. In Fig.3.2 the supporting mask for the interpolation is shown. The basic operator is monodimensional and is represented by the following function (see Fig.3.2 a) and b)):

$$x = \mu b + (1-\mu)c \tag{3.3}$$

with

$$\alpha = k((b-a)^2 + (c-a)^2) + 1 \tag{3.4}$$

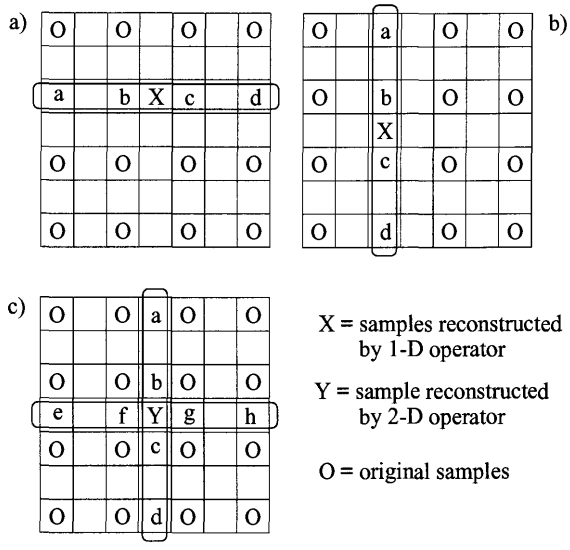

Figure 3.2: Input masks for 1-D and 2-D interpolators. Case a) and b) are related to the 1-D operator, c) is the mask for the two-dimensional interpolator.

$$\beta = k((b-d)^2 + (c-d)^2) + 1$$

$$\mu = \frac{\alpha}{\alpha + \beta} \tag{3.5}$$

then we can rewrite the algorithm as follows:

$$x = \frac{1}{1+\frac{\beta}{\alpha}} \cdot b + \frac{1}{1+\frac{\alpha}{\beta}} \cdot c \tag{3.6}$$

In the second step we use previously calculated samples to estimate the missing values (Fig.3.2). The expression of the operator is quite different from that of the first step:

$$\begin{aligned} x = {} & \mu_{ai}(a+i) + \mu_{cg}(c+g) + \\ & \mu_{df}(d+f) + \mu_{bh}(b+h) \end{aligned} \tag{3.7}$$

with

$$\alpha = \frac{1}{k(a-i)^2+1} \quad \beta = \frac{1}{k(c-g)^2+1}$$
$$\gamma = \frac{1}{k(d-f)^2+1} \quad \delta = \frac{1}{k(b-h)^2+1}$$

and

$$\mu_{ab} = \frac{\alpha}{\alpha+\beta+\gamma+\delta} \quad \mu_{cd} = \frac{\beta}{\alpha+\beta+\gamma+\delta}$$
$$\mu_{ef} = \frac{\gamma}{\alpha+\beta+\gamma+\delta} \quad \mu_{gh} = \frac{\delta}{\alpha+\beta+\gamma+\delta} \tag{3.8}$$

This choice leads us to avoid jagged diagonal border lines in the reconstructed image.

3.1.1 Analysis of the involved arithmetic functions

The keys for the design of this reconfigurable system are the introduction of even rough approximations in the evaluation of the rational functions and the exploitation of common structures inside the operators.

The latter is easily explained considering the expressions 3.1, 3.2, 3.4, 3.5. All the algorithms use a template of the form:

$$y = a_0 \frac{x_0 + x_1 - 2x_2}{(x_0 - x_1)^2 + a_1} \tag{3.9}$$

therefore all of them require the following basic arithmetic functions for the computation:

1. adder/subtracter
2. square function
3. divider
4. multiplier

Apart from the adder/subtracter these operators require a careful analysis of their implementation, because they are both time and area expansive in terms of involved resources.

The realization of the exact rational function would require a huge amount of hardware due to the complexity of the single terms constituting the operator itself. This translates into a low speed or a long latency time for the system. Therefore it isnecessary to verify if the algorithm is robust regarding approximations in the calculation of the various coefficients.

Tests over several images allow us to conclude that under certain constraints it is not necessary to have an extremely good approximation; moreover, in many cases it is sufficient to use a few bits per function in order to have an appreciable result, according to our driving criteria of small dimension and high speed.

Divider

Divisions and multiplications are traditionally the most expansive arithmetic functions to deal with. The complexity of the algorithms for realizing this operator is similar, as reported in many papers. But as we mentioned above, in the case of the rational filter, we can calculate the result of an operation with a certain approximation.

In order to simplify the implementation of the division we operate in two steps: first of all we calculate the inverse of the divisor and then we multiply it by the numerator. Therefore we now analyse the calculation of the inverse and after that we propose our solution for a multiplier.

In order to have a good approximation over the whole range of the inputs [0,255] we choose to represent the number in a floating point notation. Thus we have a normalised mantissa of the form $0.1xxxxxxx$, where the symbol x indicates an unknown value in the range [0 , 1], and an exponent.

Since in the floating point notation, the first bit after the point (i.e. the MSB) is always 1, we can consider just 3 bits of the mantissa ($d = \{1x_0x_1x_2\}$) in order to have an approximation of the function $1/x$ over the range $[1/2, 1)$, thus we obtain the following corresponce table for the divisor and its inverse:

Input	Inverse
0.1000	1.1111111
0.1001	1.1100011
0.1010	1.1001100
0.1011	1.0111010
0.1100	1.0101010
0.1101	1.0011101
0.1110	1.0010010
0.1111	1.0001000

Table 3.1: Bit-level correspondence between input d and inverse function $y = \frac{1}{d}$.

If we represent the result with 8 bits, $d^{-1} = y = \{y_0 y_1 y_2 y_3 y_4 y_5 y_6 y_7\}$, with y_0 MSB, the Boolean functions correspondent to Tab.3.1 are [1]:

$$y_0 = 1$$

$$y_1 = \overline{x}_0 \wedge (\overline{x}_1 \vee \overline{x}_2)$$

$$y_2 = x_2 \wedge (\overline{x}_0 \vee \overline{x}_1)$$

$$y_3 = (\overline{x}_0 \wedge \overline{x}_1 \wedge \overline{x}_2) \vee (\overline{x}_0 \wedge x_1 \wedge x_2) \vee (x_0 \wedge \overline{x}_1 \wedge x_2) \vee (x_0 \wedge x_1 \wedge \overline{x}_2)$$

$$y_4 = (\overline{x}_1 \wedge \overline{x}_2) \vee (\overline{x}_0 \wedge x_1) \vee (x_0 \wedge x_2)$$

$$y_5 = \overline{x}_1 \wedge (x_0 \vee \overline{x}_2)$$

$$y_6 = (x_0 \wedge \overline{x}_2) \vee (\overline{x}_0 \wedge x_2) \vee (\overline{x}_1 \vee \overline{x}_2)$$

$$y_7 = \overline{x}_1 \wedge (\overline{x}_0 \vee x_2)$$

Implementing in this way the inverse calculation we can achieve a maximum percentage error of 12% (Fig.3.3) with a complexity of about 20 NAND equivalent gates.

If we use a 0.7 μm technology the speed of this block is more then 400 MHz. Similar values are possible using four bits instead of three, but for our purposes this is not necessary.

[1] We consider $\wedge$ and $\vee$ as the logical **and** and **or** function respectively.

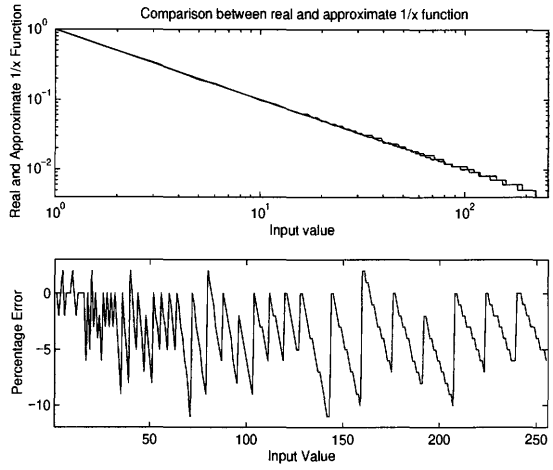

Figure 3.3: Comparison between the real and the approximate 1/x function. Curve of the percentage error

Square function

In the rational function the role of the squared difference is to perceive details in the signal; therefore we can consider just an estimate of this difference allowing us to decide whether we encountered a detail or not. As for the inverter, we consider the input in the floating point notation. In our system we do not use the square function for other purposes than this calculation, then the input of this block is in the range [0,255]. Since the range of the numbers is limited, we need just 4 bits for the exponent, plus 1 for its sign.

As for the division, if we represent $x = \{1x_0x_1x_2\}$ and the result with 8 bits, $x^2 = y = \{y_0y_1y_2y_3y_4y_5y_6y_7\}$, with y_0 MSB, we obtain Tab.3.2 and the correspondent Boolean functions:

$$y_0 = 1$$

$$y_1 = x_1 \wedge (x_0 \vee \overline{x}_2)$$

$$y_2 = x_2 \wedge (x_0 \vee \overline{x}_1)$$

$$y_3 = (x_0 \wedge \overline{x}_1) \vee (x_1 \wedge \overline{x}_2) \vee (\overline{x}_1 \wedge x_2)$$

Input	Square
0.1000	0.10001111
0.1001	0.10110011
0.1010	0.11011011
0.1011	0.10000011
0.1100	0.10011011
0.1101	0.10110101
0.1110	0.11010001
0.1111	0.11101111

Table 3.2: Bit-level correnpondence between input x and square function $y = x^2$.

$$y_4 = (\overline{x}_0 \wedge \overline{x}_2) \vee (\overline{x}_1 \wedge \overline{x}_2) \vee (x_0 \wedge x_1 \wedge x_2)$$

$$y_5 = (x_0 \wedge x_2) \vee (\overline{x}_0 \wedge \overline{x}_1 \wedge \overline{x}_2)$$

$$y_6 = (\overline{x}_1 \wedge \overline{x}_2) \vee (\overline{x}_1 \wedge \overline{x}_2) \vee (x_1 \wedge x_2)$$

$$y_7 = 1$$

$$ctrl = \overline{x}_0 \wedge (\overline{x}_1 \vee \overline{x}_2)$$

The last bit *ctrl* is used to correct the exponent for the input configurations [1000], [1001] and [1010].

As shown in Fig.3.4 in this case we also achieve a maximum percentage error of 12%; the number of gates per block is about 25, comprising the signal *ctrl* for the update of the exponent. The speed of the implemented function is more than 400 MHz, as for the inverse function.

Approximated sum

We can see in the equations 3.1, 3.2 and 3.4 that for each algorithm we must compute a sum of this type:

$$(x - y)^2 + \beta \tag{3.10}$$

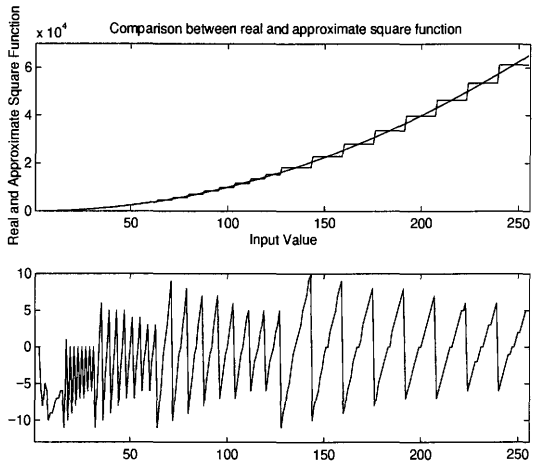

Figure 3.4: Comparison between the real and the approximate square function. Curve of the percentage error

where the parameter β avoids a 0 valued sum and is used as a threshold for the nonlinear contribution of the squared difference. As mentioned above, this squared difference gives us an estimate of the features of the image and therefore we can approximate it. Moreover it is necessary to have just a few bits in order to calculate the inverse of the sum in eq.3.10 using the mentioned divider. From these considerations we may decide to simplify also eq.3.10.

More specifically we can check if the squared difference is greater than the parameter β or not. If the check is positive, i.e. $(x-y)^2 > \beta$, then we can consider $(x-y)^2$ only, otherwise we take β. This comparison becomes critical if the terms are comparable; in this case tests indicate that the result was not very satisfying.

A further step is to calculate the sum with 4 bits for each operand; in this way we reduce the costs of the comparison and those of the sum at the same time.

In Fig.3.5 we can see the curves relative to the theoretical and to the approximate function reported in eq.(3.12) and the distribution of the percentage error. This curve shows that the maximum error corresponds

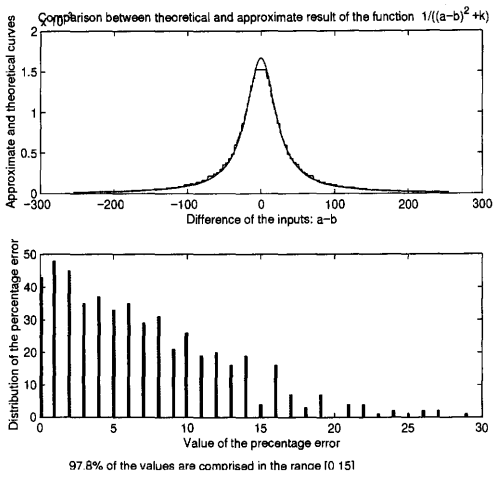

Figure 3.5: Comparison between the result obtained with the theoretical function $\frac{1}{(x-y)^2+\beta}$ and the one obtained with the approximated function.

to comparable values of the squared difference and the parameter, as we said above. In all the other cases the error is well controlled: about 98% of the results give an error below 15%.

If we consider the simplifications in the square function and in the addition and the quantisation of the parameter β these results are very good.

Multiplier

In this implementation of a reconfigurable system we include two types of multipliers: one is a radix-4 multiplier and the other is of the shift-and-add type. The reason for this choice will be explained in the next section. Let us see now the features of these blocks.

Radix-4 Multiplier

The implementation of a conventional binary multiplier requires $n \times m$ full-adders, where n and m are the number of bits of the operands. Moreover we must wait until all the bits of the results are calculated

before using them, because the final carry-chain extends along all the positions of the output.

In order to avoid this we can use particular arithmetics, which exploit the features of redundant notation [72]. Another advantage of this kind of arithmetics is that Most Significant Bit (MSB)-first operators can be realized [73].

The counterpart is that they require a conversion block from conventional binary to signed digit notation and vice-versa.

The possibility to realize a MSB-first multiplier allows us to determine *a priori* the number of bits in the output and thus the approximation we want to use.

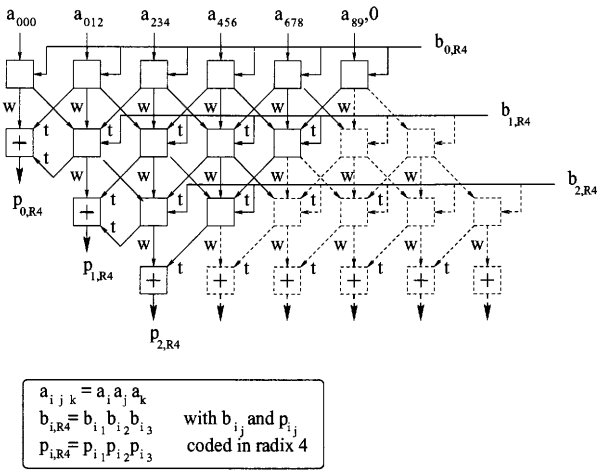

Figure 3.6: Structure of a radix-4 multiplier

In Fig.3.6 a simplified structure of a radix-4 multiplier can be seen. This particular picture refers to the case of a multiplication of a 10 bit number $a = \{a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9\}$ by a 6 bit number $b = \{b_0 b_1 b_2 b_3 b_4 b_5\}$ but with only 3 radix-4 digit in the output (equivalent to 6 binary digit).

The dashed cells are omitted because we do not need them to calculate the output.

Input b is converted to the radix-4 binary number b_{r4}; each digit can assume one of the value $0, \pm 1, \pm 2$.

Input a is not converted to radix-4 notation, but it is extended to the LSB end with a 0 and to the other extreme replicating twice the sign bit a_0. After this extension the input is partitioned into 6 overlapping slices: $(..., (a_{j-1}a_ja_{j+1}), (a_{j+1}a_{j+2}a_{j+3}), ...)$.

If the current digit of the input b_{r4} is 0 or ± 1 the two leftmost bits of each slice are used to calculate the partial product. If the digit of b_{r4} is ± 2 then we use the two rightmost bits of the slice, performing a left shift (multiplication by 2) as required by the second input.

Each cell has as output two terms: the local sum w and a transfer bit t, i.e. the carry to the next position. The term w has a radix-4 notation too, while t is a conventional binary bit. Moreover each cell transfers to the correspondent cell in the next stage the input slice of a.

The final cells in the figure indicate a radix-4 adder stage, which performs the final sum in order to produce the output digit p_i.

Since we use a small number of bits, both at the input and at the output, we do not appeal to particular conversion algorithms neither for the binary to radix-4 number conversion nor for the opposite from radix-4 to binary. Both the operations are accomplished by simple stages which sum in an opportune way the incoming digits.

The total latency time of this block, including the conversion stages is 8 clock cycles.

Shift-and-add Multiplier

As reported in [105], rational functions exhibit the interesting feature of error autocorrection between numerator and denominator. This feature has been exploited for the previous implementations of the inverse and square function. Even though the rational operator is intrinsically robust, there are particular applications where approximations must fullfill specific constraints. The rational interpolator offers a good example of such behavior.

Usually the goal of interpolation is to find a value x given the extremes a and b, in such a way that the interpolated image can be considered a "*good*" reconstruction of the original one. With "*good*" we mean that the interpolator must satisfy both analytical constraint, like edge and detail preserving, as well as subjective constraint, for exam-

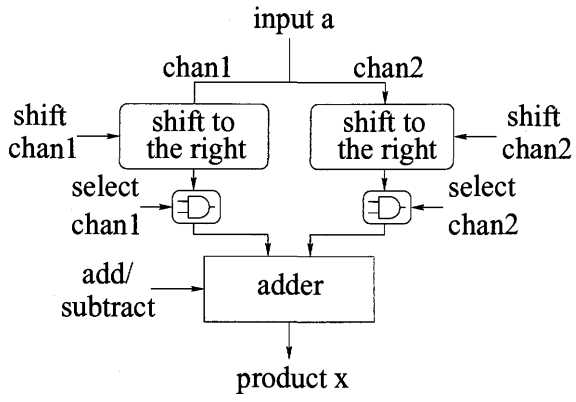

Figure 3.7: Structure of a shift-and-add multiplier

ple keeping the background texture uniform, etc.. In particular, considering Fig.3.2 a), the new value x must be included in the range $[min\{b,c\}, max\{b,c\}]$. If particular attention is not given to the approximations introduced in the rational operator, this condition is not fullfilled and the result is that the background appears jagged or corrugated when it should not be. This does not mean that approximations can not be used, however. In fact the interpolator could be realized in such a way that the coefficient μ in eq.3.4 can assume only a restricted number of values, while preserving the previous condition.

As can be seen in Fig.3.7 the multiplier is composed by one right shifter, one adder and one delay element. For each value of the coefficients the result can be calculated as a sum of two terms, obtained dividing by power of 2 (i.e. shifting to the right) the input value.

For example, suppose that we want to multiply a number a by $\frac{3}{4}$. The coefficient can be written as $1 - \frac{1}{4}$; thus a shift of the first channel by two positions and its subtraction from the input is sufficient.

The advantages of this kind of structure are that it is very easy to implement and it has a very low latency time compared to a conventional multiplier. In fact it can be realized with three blocks: two shifters in parallel and one adder. But since the inputs to the adder have more than 8 bits, this block must be split into two sub-blocks and therefore 3 clock cycles are needed to perform the multiplication.

The counterpart of this structure is that it can handle only a limited number of coefficients.

Using lookup tables

The realization of the noise-smoothing edge preserving filter and that of the deblocking filter are really effective and easy to do.

However some considerations concerning the interpolation problem must be made .

As mentioned above, all the simplifications work well under certain constraints. For the interpolator (refer to eq.3.3 and Fig.3.2 a)), the coefficients μ and $(1-\mu)$ must satisfy the obvious condition:

$$\mu + (1-\mu) = 1 \tag{3.11}$$

which allows the value of the calculated sample x to be in the range $b \leq x \leq c$.

But if approximated functions are used in order to obtain these coefficients, they do not satisfy this condition anymore. The result is an annoying texture effect in the image, due to the approximations in the values, which are out of the admitted range.

On the other hand, it is impossible to have a full-bit precision structure for the interpolator only. We must exploit the robustness of the algorithm to approximations in other ways.

A simple but effective tool used to calculate functions is a lookup table (LUT): it is fast for reasonable dimensions and can incorporate all the approximations we want on the function.

The addresses for the table are obtained from the block which computes terms of the type

$$\frac{1}{(x-y)^2+\beta} \tag{3.12}$$

From eq.3.6 we can see that the required function is

$$\frac{1}{1+\frac{\beta}{\alpha}} \tag{3.13}$$

where both α and β are as in eq.(3.12). This function is rough quantised to simplify its evaluation; in Tab.3.3 we report the considered

Value	$x_0x_1x_2$
0	000
1/8	001
1/4	011
1/2	010 or 101
3/4	100
7/8	110
1	111

Table 3.3: Quantised values for the interpolator μ coefficient.

values and their associated codes. Values for $1-\mu$ are obatined negating the output bits.

Referring to Fig.3.7, the $x_0x_1x_2$ bits determine if the input a has to be shifted in channel-2 or not, whether both channels are used for the final sum and whether the channel-2 addend has to be inverted or not:

$$shift = \begin{cases} (\overline{x}_0 \wedge \overline{x}_1 \wedge x_2) \vee (x_0 \wedge x_1 \wedge \overline{x}_2) & \text{shift by 3 positions} \\ (\overline{x}_0 \wedge x_1 \wedge x_2) \vee (x_0 \wedge \overline{x}_1 \wedge \overline{x}_2) & \text{shift by 2 positions} \\ (\overline{x}_0 \wedge x_1 \wedge \overline{x}_2) \vee (x_0 \wedge \overline{x}_1 \wedge x_2) & \text{shift by 1 position} \\ (x_0 \wedge x_1 \wedge x_2) \vee (\overline{x}_0 \wedge \overline{x}_1 \wedge \overline{x}_2) & \text{no shift required} \end{cases}$$

$$add/subtract = select\,chan1 = x_0$$

$$select\,chan2 = (x_0 \oplus x_1) \vee (x_0 \oplus x_2)$$

For this implementation we choose to realize the LUT with a combinatorial circuitry for two main reasons:

- the values to be stored in the table are just 7, thus it would be a waste to use a ROM or a RAM;
- it is possible to realize a very compact circuitry, the speed of which is much higher than that of a conventional memory.

The realized LUT for the 1-D interpolator needs about 30 gates and has a speed greater than 500 MHz if implemented with a 0.7 μm technology.

The LUT for the two-dimensional algorithm requires a few more blocks in order to condition the input values of the form in eq.(3.12) and this makes the latency time increase.

In this case the function is quantised to the values:

$$0\,,\frac{1}{8}\,,\frac{1}{4}\,,\frac{1}{2}\,,\frac{3}{4}\,,1$$

The dimension of the table is almost twice that of the 1-D operator, but the speed is just slightly less.

Using the lookup table is not restrictive for our purposes, but it gives us more flexibility for other algorithms which can be implemented in this way.

3.1.2 Implementation of the system

In Fig.3.8 we present the overall structure of the reconfigurable system.

The input blocks indicated as PLF are Programmable Linear Filters. They should realize a linear operator of the form:

$$y_n \;=\; w_1 \times a \;+\; w_2 \times b \;+\; w_3 \times c \tag{3.14}$$

a,b and c are samples in the filter mask. w_i are programmable coefficients, which can assume the following quantised values:

$$0\,,\pm\frac{1}{8}\,,\pm\frac{1}{4}\,,\pm\frac{3}{8}\,,\pm\frac{1}{2}\,,\pm\frac{3}{4},\;\pm\frac{7}{8}\,,\pm 1\,,\pm\frac{3}{2}\,,\pm\frac{7}{4}\,,\pm 2 \tag{3.15}$$

The product $w_i \times x$ is calculated with a shift-and-add multiplier. Depending on the selected algorithm the outputs of the PLFs enter either in the radix-4 or in the shift-and-add multiplier, where they are multiplied by the coefficients provided by the core section; after that the various contributions are summed together with a two-stage adder. Finally a multiplier is used to eventually scale the final output for other purposes.

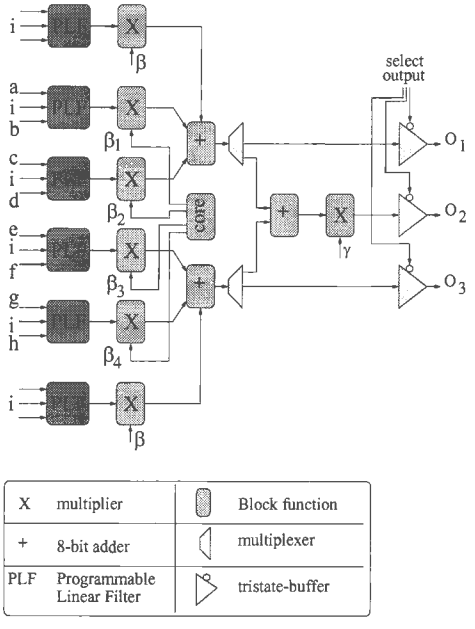

Figure 3.8: Overall structure of the filter

The main flow is subdivided into two channels in order to have the possibility to implement, for example, two 1-D filters instead of one 2-D. To this end three-state buffers make it possible to select the output channels.

The structure of the core of the system is more complicated, because we must consider all the possible paths among the blocks in order to realize a specified function. As can be seen from Fig.3.9, the input data are sent to the first stage, where the squared differences are computed. Depending on the algorithm the result is summed with the parameter β, like in eq.(3.1,3.2) or with another squared difference (eq.(3.4)).

Thereafter a path to an inverter or to an adder can be chosen: in the first case we compute a term like in eq.(3.12), which can be used directly for the noise-smoothing filter after a multiplication by the parameter α, like in eq.3.1,3.2.

If we want to implement either the 1-D or the 2-D approximated

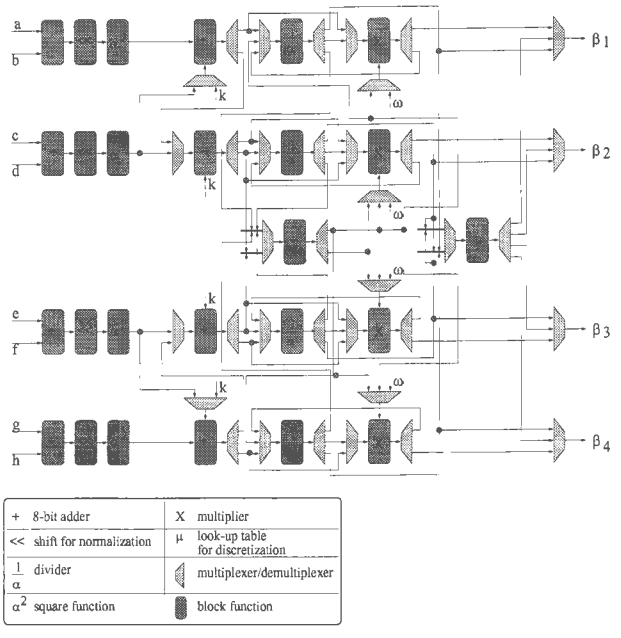

Figure 3.9: Structure of the filter's core.

interpolator we can send the output of the inverter to the lookup table conditioning block, which prepares the addresses for the table itself. Otherwise, if we want to compute the interpolator coefficients in the correct way, we can use an extended adder in order to have the sum at the denominator, as in eq.3.5 and 3.8.

This term multiplies each of the addenda in order to calculate the final coefficients.

In the case of the deblocking filter the parameter k is calculated outside the system and then sent as parameter.

We designed the whole system as an extensive bit-level pipelined architecture [55], which allows us to achieve a throughput of one new calculated sample per clock cycle.

Each function is constituted by a combinatorial circuitry and by a memory stage. Since in such a structure the minimum clock period depends on the execution time of the slowest block, it is necessary to

design each function in order to have a delay time as similar as possible from one block the each other.

Therefore the choice of the reference time is very critical, because if it is to small, a high number of memory cells will be required; on the other hand a too large delay time does not allow a sufficiently high clock frequency. We take as reference delay time the delay of an 8-bit adder.

Since the system has few blocks and the paths among them are not numerous, we designed the reconfiguring circuitry as a simple shift register. Some multiplexers drive the signals to one path or another according to the value stored in a dedicated memory cell.

An input FIFO stores the incoming samples in order to preserve them for the next calculation and reduce the number of input signals. In fact, when an image is scanned for filtering, at each step only three new samples enter the mask, while the others will simply change their position.

3.1.3 Results

In order to test the system we have created a simulator in C code, which performs bit per bit all the functionality of each block. We then applied the simulator to some images for the various algorithms.

In Fig.3.10 the images related to the noise-smoothing algorithm are shown. The original image was corrupted with Gaussian noise; the reconstructed image is obtained after only one passage of the filter (the filter parameters are the same as reported in [41]).

Fig.3.11 is related to the interpolation algorithm. The original image was first filtered with a low-pass filter and than decimated by a factor of 2.

Fig.3.12 refers to the deblocking algorithm. The image was coded with JPEG at 0.3 bpp, as reported in [34]. In this particular case the result must be considered very good, because the improvement apported to the image by the theoretical algorithm is just 0.3822 dB with respect to the decoded image. Moreover, since we use for the calculation the square function instead of the fourth power, we must trigger slightly the parameters of the filter.

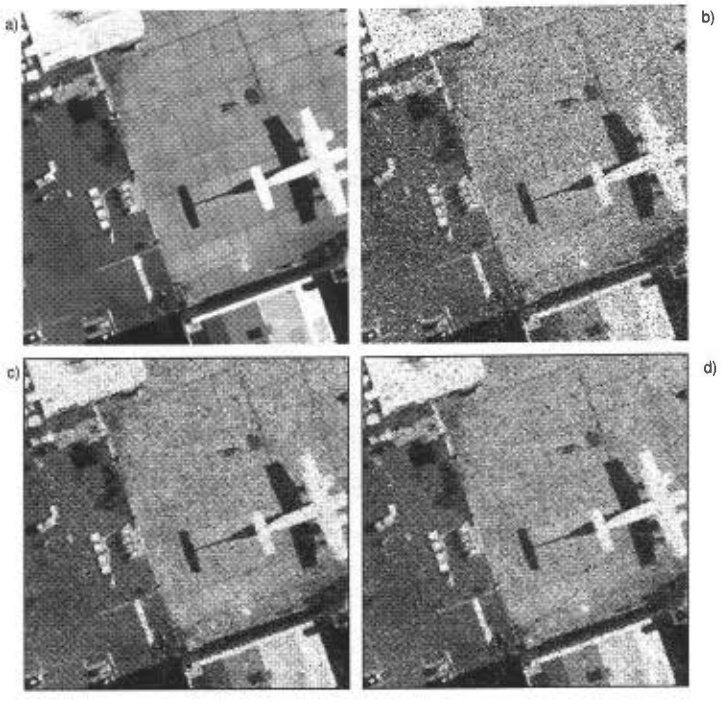

Figure 3.10: Noise-smoothing Filter: a) original image, b) corrupted image, c) image filtered with our system, d) image filtered with the theoretical algorithm

In Tab.3.4 the values of the PSNR and of the MSE for each algorithm and each image are reported.

For each algorithm the correspondent latency time in the system is indicated: as can be seen from the values, the three algorithms require almost the same time. This is due to a balance between the time required for the calculation of the nonlinear function and that required for a multiplication.

As we mentioned above the radix-4 multiplier requires 8 clock cycles in order to perform the operation; the shift-and-add multiplier needs just 3 clock cycles. The 5 cycles of difference balance the higher time used to calculate the nonlinear function in the various cases.

Figure 3.11: Interpolator: a) original image, b) filtered/decimated image, c) image reconstructed by our system, d) image reconstructed with theoretical operator.

The 2-D interpolator needs slightly more time than the 1-D operator due to the more complex circuitry for the lookup table.

3.1.4 Physical features

In Fig.3.13 we can see a layout of the whole system. We designed this structure with the Cadence EDA tool using a 0.7 μm CMOS standard cells technology. With those parameters, the number of gates of the whole chip is about 12.000, almost equally distributed between the core section and the upper section.

The size of the core of the chip is about $4.3 \times 4.6\ mm^2$, which grows up to $7 \times 5.1\ mm^2$ if we consider the pads too.

Figure 3.12: Blocking artifacts removing Filter: a) original image, b)decoded image, c) image filtered with our system, d) image filtered with the theoretical algorithm

As we mentioned above this architecture has been designed to achieve at least a clock frequency of 200 Mhz. Higher frequencies can be achieved by splitting the various blocks into two parts, at the expense of greater dimensions.

		Hardware	Theoretical
Noise-Smoothing Filter	PSNR	20.2975	20.3226
	MSE	607.2893	603.6926
	latency	30	—
Interpolator	PSNR	34.2484	32.7312
	MSE	24.4481	34.6709
	latency	31 (1D) 34 (2D)	— —
Deblocking Filter	PSNR	30.0939	30.2752
	MSE	63.6337	61.0318
	latency	30	—

Table 3.4: Comparison between theoretical and implemented rational operators.

3.2 Median-Rational Hybrid Filter

The second operator we implemented is a hybrid filter. As we said in the first chapter, rational operators perform very well for Gaussian noise, but their performances decrease when impulsive or long-tailed Gaussian noises are present.

In [24, 25] a hybrid median-rational filter, MRHF, has been proposed. It combines the good performances of the median filters for impulsive noise with the ability of the rational operators to deal with Gaussian noise.

The MRHF has been further modified in [] to operate on colour images too.

Since the structure of this operator is very simple, we decided to try to implement it on an FPGA.

The general form of the MRHF is the following:

$$y(n) = \Phi_2(n) + \frac{\sum_{i=1}^{3} \alpha_i \Phi_i(n)}{k + h(\Phi_1(n) - \Phi_3(n))^2} \tag{3.16}$$

where the coefficients $\alpha = [1, -2, 1]^T$ satisfy the condition: $\sum_{i=1}^{3} \alpha_i = 0$. h and k are some positive constants used to control the nonlinear effect, as in [41]. Phi_i are median operators acting on user-specified masks.

The structure used for the MHRF is shown in Fig.3.14. In this case the sub-functions Φ_1 and Φ_3 are two bidirectional vector median filters acting respectively on a plus-shaped, PMF, and on a cross-shaped, CMF, mask. The term Φ_2 is obtained from a Center Weighted Median Filter, CWMF, acting on a plus mask as shown in the following:

$$\Phi_1 \rightarrow \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} \Phi_2 \rightarrow \begin{pmatrix} 0 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 0 \end{pmatrix} \Phi_3 \rightarrow \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

3.2.1 The Median Filters

The implementation of a median filter requires several comparisons among its inputs. In the specific case, the base mask is of length 5, therefore at least 9 comparisons are necessary to completely sort the data.

Since the system had to be implemented on a FPGA, we derived the structure of these sub-filters from a Xilinx application note. It is based on the fact that the sign bit in a normal subtraction can be used to determine which of the inputs is the greatest. Since the input data are all positive, the sign bit of the subtraction corresponds to the carry bit at the most significant position negated. Therefore it is not necessary to actually perform the subtraction, since the very carry chain is needed.

This approach is very suitable for implementation on Xilinx FPGA, since they have a built-in optimized carry-chain (see Fig.3.16). Using this block still leaves the designer with the possibility to use the other internal Lookup Tables in order to perform other logic operations on the same values used for the median.

Given two 8-bit numbers a and b, the application note supposes the availability of a block able to output both the maximum and the minimum between the two numbers. This operation is easily accomplished using the output carry of the subtraction, as previously mentioned, as select bit of an 8-bit 2-to-1 multiplexer (see Fig.3.15).

In Fig.3.17,3.18 the architectures of the 3 sub-filters are shown. For

the sake of clearness we show the two structures of the PMF and CWMF as separate.

In the design phase we supposed that 3 input data are available at each clock cycle. This is compatible with an input line memory able to store two lines of the incoming image at a time. As the new data arrive, they replace the old ones, which are no longer useful. In Fig.3.19 we reported a simple example. When the new data is available, the filter mask is shifted to the right and processed. All the system works in a pipeline architecture, therefore memory elements are required for each block.

The CWMF can exploit part of the structure of the PMF, since they act on the same mask, apart from the central value. If we sort the 7 pixels of the CWMF's mask, there are only two possible configurations for which the central term is not also the median, i.e. when it is respectively the minimum or the maximum value in the mask.

In the PMF algorithm, the second step yields the maximum M_4 and the minimum m_4 of the four already received pixels (a_1, a_3, a_5 and a_7 in Fig.3.17). If we delay the central pixel a_4 until this step has been computed, it is only necessary to compare it with M_4 and m_4 to know if it is the median or not. With this simple modification, the hardware overhead with respect to the simple PMF implementation is only 1 delay element, 2 max-min blocks and a multiplexer.

The architetcture of the CMF is slightly more complex, due to the timing of the input data. Moreover it is not possible to share part of the structure with the PMF or CWMF.

In the count of used resources, the input memory elements storing the data in the mask are shared with the whole structure, since it represents the actual input block of the system.

The total number of Configurable Logic Blocks (CLB) used for the implementation of these three sub-filters is 207, with an achievable maximum speed of 50 MHz.

3.2.2 The Rational Function

The rational operator has the standard expression found in the previous section for the noise-removing and artifacts-removing operators:

$$y(n) = \Phi_2(n) + \frac{\Phi_1(n) + \Phi_3(n) - 2\Phi_2(n)}{(\Phi_1(n) - \Phi_3(n))^2 + \beta} \tag{3.17}$$

with $\alpha = 1/k$, $\beta = 1/wk$ and w and k as in [41].

As a first approach we decided to use a residue number system (RNS) in order to perform the calculation of both numerator and denominator of the rational operator, because it allows a high parallelism in the operations. A residue number system, though, usually implies redundancy and therefore it needs more hardware. Moreover adders and multipliers in RNS are non-standard structures which require an *ad hoc* hardware to be implemented. On the other hand FPGAs supply built-in operators optimized for the standard binary system, like adders/subtractors and comparators; exploiting these blocks led us to obtain a faster and smaller circuitry.

The features of a system depend very much on the architecture chosen for the implementation. In image processing the amount of calculations to be performed in the time unit is very high even if the image size is small. The system should be able to read and to output data at high rate. To this purpose we decided to implement our rational operator with a pipelined structure, as the median sub-filters. The main drawback of this choice is the large amount of memory elements required at each step. Moreover the FPGA has a relatively small number of built-in memory elements. Therefore the system has to be designed carefully in order to minimize the latency time, or better the number of stages in the pipeline.

As can be seen from eq.3.17 the more demanding arithmetic operations are the square function and the division. In this case we cannot adopt the floating point representation for the input data, as for the reprogrammable filter, since it would require more hardware resources. Therefore another approach had to be found to minimize the number of operations.

The main problem introduced by the square function is the increase in the word width, i.e. in the number of bits in the result, whereas in the RNS all the operations are evaluated with a constant number of bits.In our work the operator is fixed; thus we can exploit our knowledge about the parameters appearing in the function in order to reduce its complexity. In the denominator in eq.3.17 two parameters are used. In particular we can rewrite eq.3.17 as follows:

$$h(\Phi_1(n) - \Phi_3(n))^2 = (\sqrt{h}(\Phi_1(n) - \Phi_3(n)))^2$$

Following the example in [41] these two parameters have been chosen as follows: $k = 6.25$, $h = 0.01$. If we approximate h with a simple sum of powers of 2 the number of bits required by this operation can be reduced. For instance if $h = 2^4 + 2^5$ the range of the term to be squared is limited to 5 bits instead of 8. Moreover the square function can be implemented with a simple shift-and-add multiplier with a precision of 11 bits in the result.

Simulation runs over several images showed that the loss of precision introduced with this approximation is very small, while the evaluation of the denominator has been simplified to 6 additions.

In this design the role of the division is the most critical, since this operation is highly time and size consuming.

The choice of the actual implementation of this operation strongly depends on the constraints on the design and on the precision required by the rational operator and the application. In fact not all the rational operators allow a course approximation in the results if we want them to work effectively, like, as we said, the interpolator.

We developed two different algorithms for the division. The first one is an iterative algorithm derived from [67] which can be implemented both in RNS and in standard binary. This algorithm allows us to obtain a precision in the results of about 1% after 4 iterations. Further tests on images showed that 3 iterations are enough to obtain a good result and this allows a good reduction in terms of size. Nevertheless this algorithm requires few combinational blocks or lookup tables for the selection of the quotient and variable shifts as the new dividend is obtained using the standard formula $y_n = y_{n-1} - q_{n-1} \times x$, where q_{n-1} is the partial

quotient at iteration $n-1$. Moreover the latency time would be greatly increased due to the need of several iterations.

The second algorithm is very simple and is based on successive scalings of both numerator and denominator. Given two numbers n and d, the division $y = n/d$ can be represented as follows:

$$y = \frac{n}{d} = \frac{s^{e_n}\, n_s}{s^{e_d}\, d_s} = s^{(e_n - e_d)}\, \frac{n_s}{d_s} = s^{(e_n - e_d)}\, n_s\, d_s^{-1}$$

where s is a scaling factor, $n_s$and d_s are respectively the scaled numerator and the scaled denominator and e_x is a number such that $s^{e_x} \times x_s$ is the largest number less than x.

In this case we can choose $s = 16$: in this way each scaling can be performed by simply rejecting the 4 less significant bits and the rounding can be easily accomplished summing 1 or 0 depending on the 4 bits rejected.

Since the numerator and the denominator are represented with 10 and 11 bits respectively, after at most 2 scalings by 16 the range is reduced to [0,16], taking into account the rounding effect. A small lookup table supplies the correspondent inverse of the denominator d_s^{-1}. Since the maximum number of scaling can be 2, the difference $e_n - e_d$ can assume values in the interval [-2,2] only, reducing the complexity of the shift.

As the results in the following section show the first algorithm outperforms the second one and allows us to obtain even better results than the theoretical algorithm. The second algorithm, though, is very compact and introduces a latency in the system which is less than 1/4 of the latency of the former. Moreover the required hardware can be easily and effectively implemented exploiting the features of the FPGA.

In Fig.3.20 the block diagram of the resulting circuitry for the rational function is shown. We implemented the system on a medium size FPGA containing 400 CLB.

As can be seen from eq.3.17 the numerator is a very simple linear function; it requires 2 8-bit adder/subtractors and one 9 bit adder. As aforementioned, the design tools for FPGAs allow exploitation of some of the features of these chips. As a comparison with the RNS system, an 8 bit subtractor in standard binary requires 6 CLBs and can be operated

at about 58 MHz; on the other hand a 5 bit adder for the modulo 17 addition requires 7 CLBs and its operating frequency is about 56 MHz. This simple example well justifies the choice of the binary system due to the simplicity of the rational operator.

The denominator is slightly more demanding in terms of hardware because of scaling and multiplication. The number of CLBs required for the denominator is 2.5 times that for the numerator and comprises the flip-flops for the pipeline.

The division itself requires as much hardware as that used for both numerator and denominator, even if the implemented algorithm is relatively simple. The main timing constraints derive from this block, mainly due to the arbitration logic used to choose the appropriate results instead of the traditional shift register. This is done because we do not know *a priori* how many shifts we need at each step; since the number of shifts is relatively small, a simple logic can directly output the result in one clock cycle.

The latency time of the system is 8 clock cycles: 4 are introduced in the evaluation of the denominator, 3 in the evaluation of the division and 1 more to obtain the final sum.

The total number of CLBs used for the rational function is 191, about 47% of the total number at our disposal; the number of flip-flop is 182 and almost 1/3 is used to store Φ_2 during the evaluation of the rational function. The maximum achievable frequency is 41 MHz, therefore this system can be effectively used for real time application with images of 768x625 pixels at a frame rate of 50Hz.

In Fig. 3.21 the layout of the rational operator on the FPGA is reported.

3.2.3 Experimental Results

The implemented system has been tested using images corrupted with i.i.d. noise having the following probability distribution:

$$\nu = (1-\lambda)\mathcal{N}(0,\sigma_n) + \lambda\mathcal{N}(0,\frac{\sigma_n}{\lambda}) \tag{3.18}$$

Three values of λ and several different values for the SNR have been chosen: $\lambda = 0.1$ (very impulsive noise), $\lambda = 0.2$ (mixed Gaussian-impulsive

noise) and $\lambda = 1$ (purely Gaussian noise), with SNR= 3dB, 6dB, 9dB and 15 dB.

As an example, Figs.3.22(a)-(e) are respectively the original clean image, the noisy image with $\lambda = 0.1$, $SNR = 3dB$, the image filtered by theoretical MRHF, the output of the implemented MRHF system with the high precision algorithm for division and the output of the implemented MRHF with the low precision algorithm for division. It has to be noticed that the implemented median filters do not introduce any error, since their output is always exact. It can be clearly seen that the images in Figs. 3.22(c) and 3.22(d) are very close to each other, and both show a good noise attenuation and present sharper edges and smoother flat areas, while Fig.3.22(e) presents a worse quality.

In Fig.3.23 we present the comparisons of results obtained for some images. For completeness sake we reported both the results obtained with the iterative (*Hardware (1)*) and with the simplified (*Hardware (2)*) algorithm for the division. All the data have been normalized to the maximum valued obtained. In all the cases the implemented structure with the iterative algorithm gives even better results than the theoretical one, with a average absolute relative error of about 3.5% for MSE and 2.1% for MAE. This can be explained in two ways basically. The first concerns the coefficients h and k in the denominator. Their values have been chosen following the referenced paper [41], but they are not a result of an optimization process involving the MRHF. The introduced approximation modifies these values, therefore it is possible that they are closer to optimum values than the chosen ones. If we modify h and k with the values obtained with the approximation the results obtained in the theoretical case are better indeed, but they are still worse than those obtained with the implemented system.
The second explanation concerns the rounding errors introduced in the evaluation of the denominator. These errors have the effect of reducing the actual value of the denominator. Therefore it does not affect the numerator so strongly as in the theoretical algorithm. The overall effect is a low-pass filtering which slightly improves the results.
The results obtained with mixed noise show that in this case the simplified algorithm usually outperforms the theoretical operator.

3.3 Conclusion

In this chapter we presented two implemetations of rational filters.

The proposed architectures clearly show that these kind of operators are very suitable for image processing as well as for simple but effective realizations.

Two approaches have been adopted for these implementations: the first is an ASIC design, allowing great flexibility and high operating speed, while keeping a small die size. The second approach is based on FPGA design. In this case an easily reconfigurable cheap product can been obtained at the expense of the overall speed. In any case the system is able to operate up to 50 MHz, which is enough for HDTV applications, as well as video conference or similar applications.

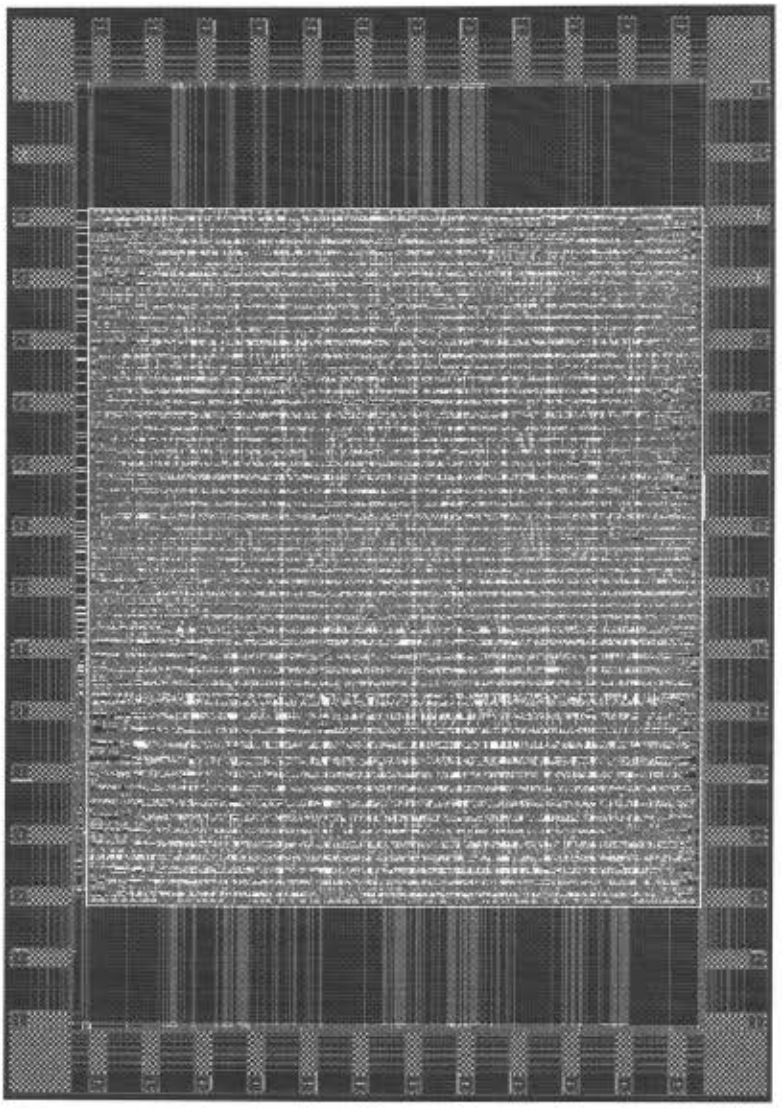

Figure 3.13: Layout of the implemented system

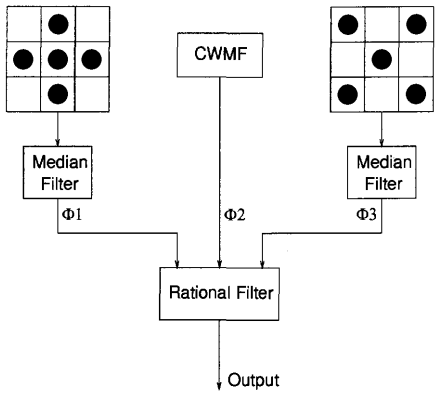

Figure 3.14: Structure of MRHF using two bidirectional median subfilters .

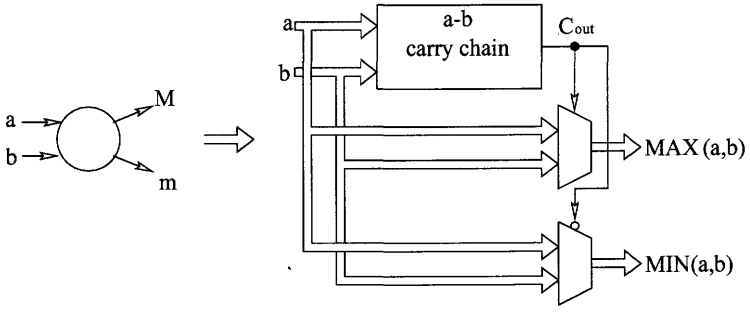

Figure 3.15: Structure of subfilter's building block.

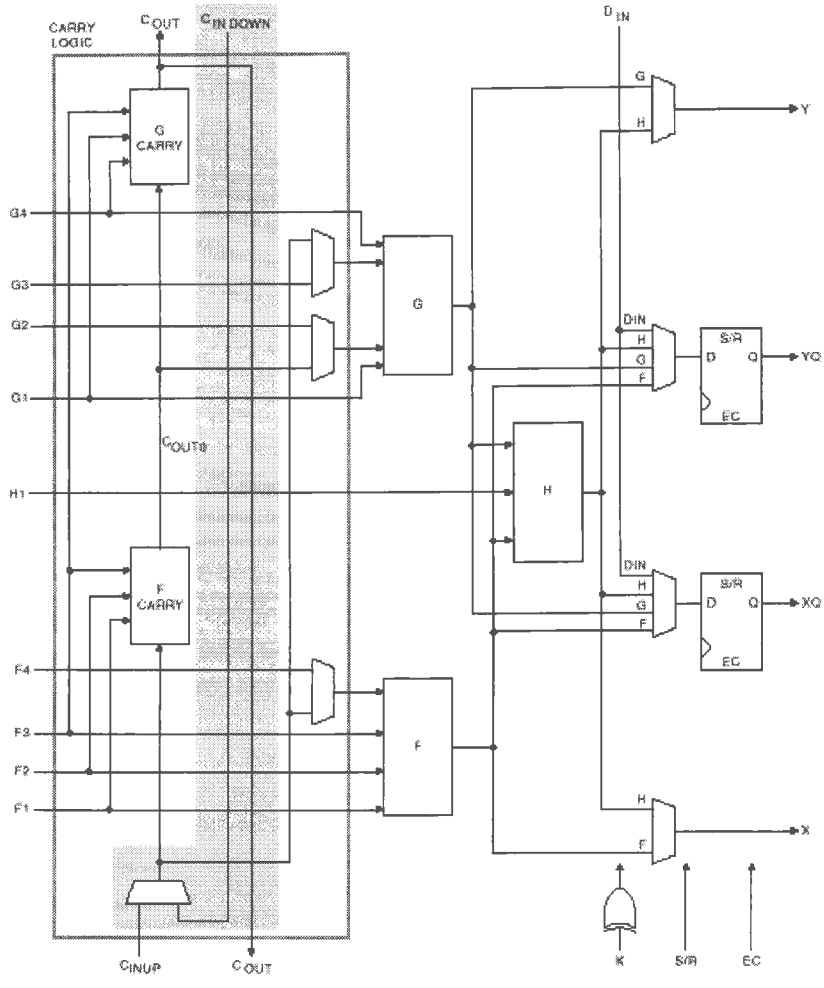

Figure 3.16: Structure Xilinx CLB's carry chain.

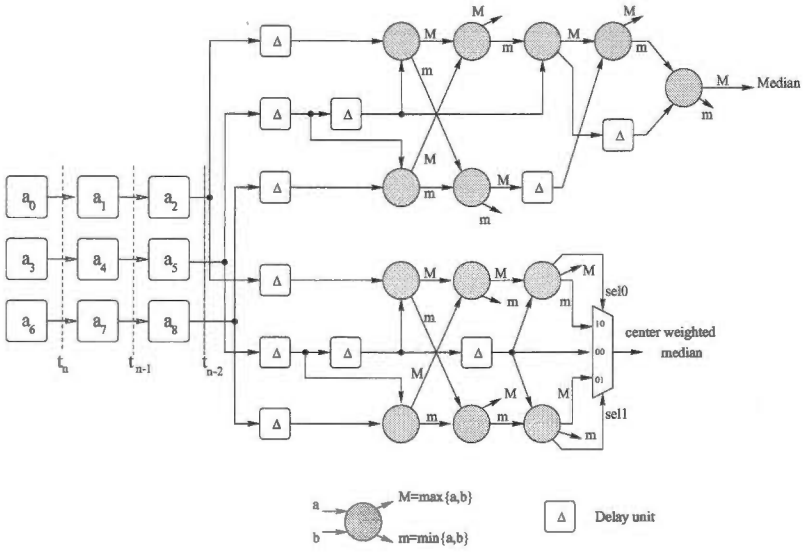

Figure 3.17: Structure of the PMF and CWMF sub-filters.

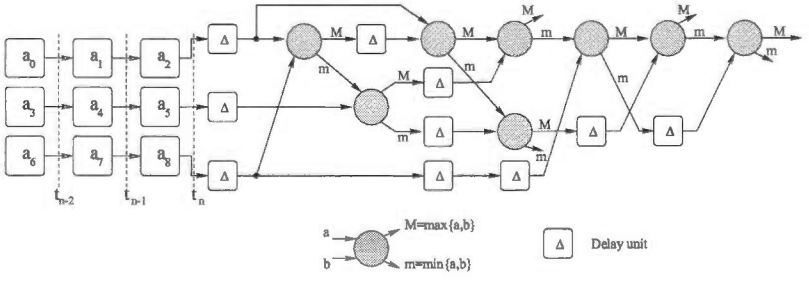

Figure 3.18: Structure of the CMF sub-filter.

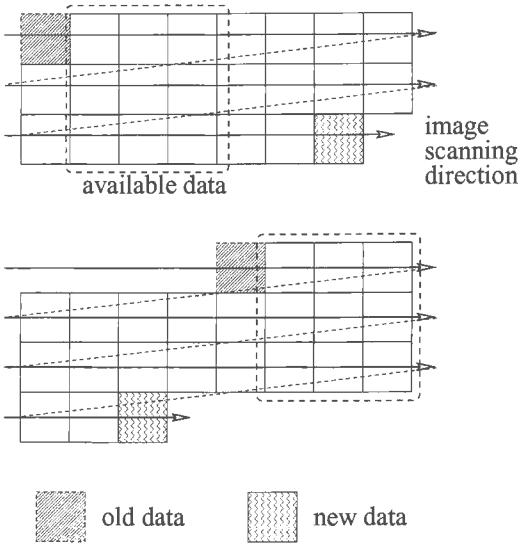

Figure 3.19: Input line memory.

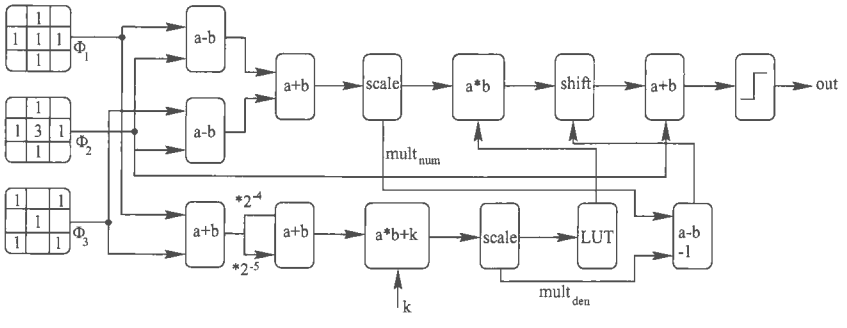

Figure 3.20: Block diagram of the implementation of the rational function..

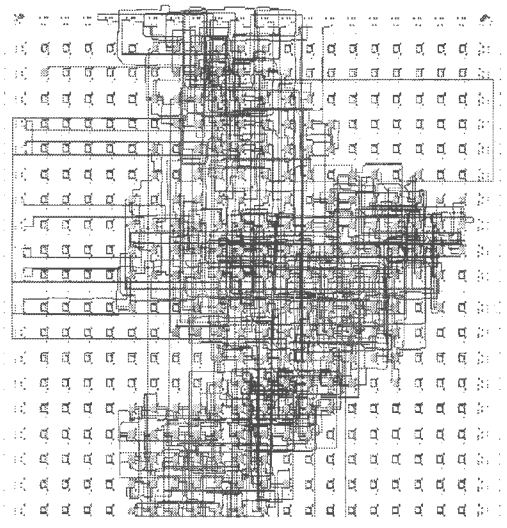

Figure 3.21: Layout on FPGA of the MRHF'rational operator.

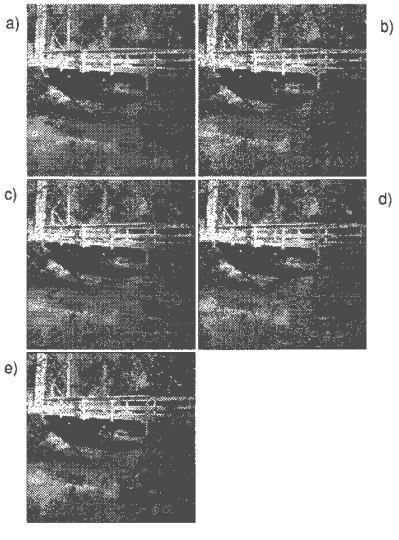

Figure 3.22: Qualitative comparison between implemented and theoretical filter (See text).

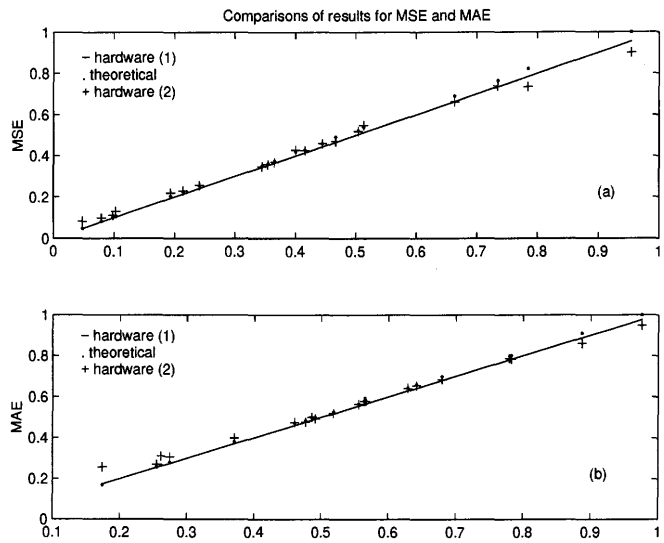

Figure 3.23: Comparisons between implemented and theoretical filter (See text).

Chapter 4

Macromodelling Techniques for Power Estimation

Introduction

With the spreading use and need for portable devices the power dissipation of the electronic circuits has become a critical concern and a fundamental issue in circuit design. Modern microprocessors can exceed 50 Watts in power consumption, cellular phones and laptops need to dissipate very few power, since the available batteries have limited life, and smart power or power-conscious systems are becoming more and more important for every day applications.

Beside practical considerations macromodelling plays a relevant role in the so called intellectual property issue. Usually companies supplying building blocks (e.g. DSP or μC cores) for other applications would like to keep secret the technology beyond their products. At the same time the user should be able to obtain all the information he needs for the blocks he uses in order to optimise the design. In this context macromodelling could solve the problem, giving all the information to the user without showing how the system actually works.

The availability of tools for fast power estimation would allow the

designer to explore several configurations for the system and to find the trade off among area, speed and power. To this end it is important that the estimation phase takes place at the highest level in the design process.

High level of description means that the system is characterised by its functionality or by the functionality of its components. The inner structure of the circuitry is usually unknown because it is the result of the following optimisation and synthesis phase. It is well understood that at this level circuit simulation could be very fast, therefore many different configurations can be analyzed within very short time. However at this level the knowledge about the circuit is poor and consequently the results of simulations will be inaccurate.

On the other side an accurate description of the circuit (e.g. at transistor level) would lead to very good estimates for its features, but at the price of long lasting simulations.

Several techniques have been proposed in the past few years working both at high and low level of description, but the difficulty of representing and taking into account all the phenomena responsible for power consumption leaves an open space for research.

In the following we present a brief overview of the major techniques used for power estimation and our own new recently proposed technique. We'll show that our method is very powerful and can be used in a wide variety of applications.

4.1 Circuit Description Levels

As we said above a circuit can be described at several levels. In order to understand further analysis of proposed estimation algorithms we shortly summarize the used descriptions, their differences and the assumptions they require about the circuit.

In Fig.4.1 the description levels for a simple full-adder circuit are shown.

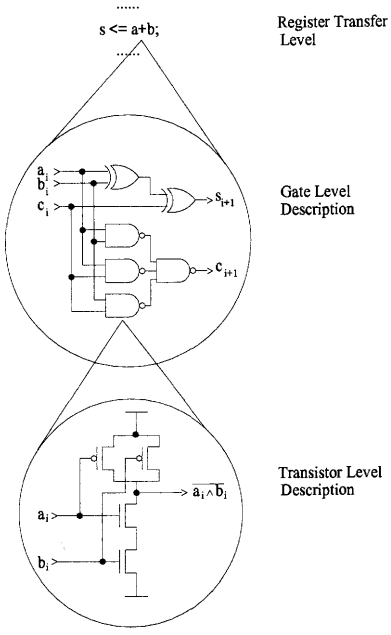

Figure 4.1: Description levels for digital circuits.

4.1.1 Register Transfer Level - RTL

The RT-Level is meant to be the highest description level for digital circuits. At this level the system is described as a set of macro-operators, like adders, multipliers, memories, registers, and so on. For each of these blocks several possible implementations are possible as well as different circuit solution (e.g. Domino logic, standard CMOS). Therefore the knowledge about the system is minimal and restricted to its functionality.

Nevertheless this is the most interesting level for power estimation and power-estimation-based synthesis. In fact the system can be easily and quickly manipulated at this level, allowing the choice of the best composition of the inner blocks.

In this case the synthesis process is usually based on a library of predefined blocks. Each of them has fixed features like speed, area and

power dissipated [130]. Each time a new block is needed a separate characterization phase is accomplished. The goal of the modelling technique is to build a model accurate enough to give good results, but simple so that the simulations can be run quickly.

The intellectual property issue belongs to this class of problems, since the supplier wants just to give a black box from which the user can extract the interesting features.

4.1.2 Gate Level

At this level the circuit is described as a set of interconnected logic gates and memories or, equivalently, with a set of Boolean functions.

The behaviour the system is strictly dependent on the technology used to implement the gates. The silicon foundry supplies the technological library containing all the information about the available gates. Therefore a simulator can take into account all the parameters like delay and dissipated power.

The evaluation of Boolean functions can be executed quickly and it also gives accurate information about the outputs of the system. Nevertheless the execution time increases with the complexity of the gate model. Thus a lot of efforts have been spent trying to simplify these models while keeping as much information as possible. The main target of simplification is the delay model of the gate.

The first and simplest way to avoid this problem is to suppose that each gate outputs its result instantaneously. it is the so called *zero-delay* model. The simulation consists of the evaluation of the logic functions and of the propagation of the results through the system.

Obviously this model cannot be used for sequential circuit and it doesn't consider spurious phenomena at the output like glitches.

A more accurate model assigns a unit time delay to each gate. This is called the *unit-delay* model. It allows the simulation of sequential circuit but again glitches at the output of the gate are not considered. it is to be noticed that the simulatio n is more complex with respect to the case of *zero-delay* model, because the transition time of the outputs varies with the logic depth of the circuitry.

Finally the *general-delay* considers different delays for different gates, therefore it is possible to simulate undesirable effects like glitches.

4.1.3 Transistor Level

The transistor level is considered the lowest level of description. Each gate is represented with the corresponding transistor structure and for each component (e.g. resistor, MOS, BJT, capacitor) a detailed model is given. The simulation is accomplished with tools like Spice and the completion time increases with the size of the circuit. On the other side it is possible to obtain a very deep knowledge about the behaviour of the system.

The study of the circuits at this level is fundamental to understand what are the factors influencing the power dissipation. In fact a lot of work has been done about this topic.

In the following paragraph we present a short description of the most important terms taking part in logic circuit power dissipation.

4.2 Factors Determining Power Dissipation

A distinctive feature of static CMOS circuits is that the power dissipation is mainly caused by the input signal switching, i.e. by the dynamic behaviour of the circuit. Power is consumed by charging the load capacitances at the output node of the gate and it is dissipated when these capacitances are discharged.

If we suppose to have a digital circuit composed by n logic gates we can roughly approximate the dynamically dissipated power with the following expression:

$$P_{avg} = f \frac{V_{DD}^2}{2} \sum_{i=1}^{n} C_{L_i} \eta_i \tag{4.1}$$

In this expression f represents the frequency, V_{DD} is the power supply voltage, C_{L_i} is the load capacitance of the i-th gate and η_i its *transition density*. The later term expresses the number of output's transitions from 0 to 1 and from 1 to 0 in the time unit.

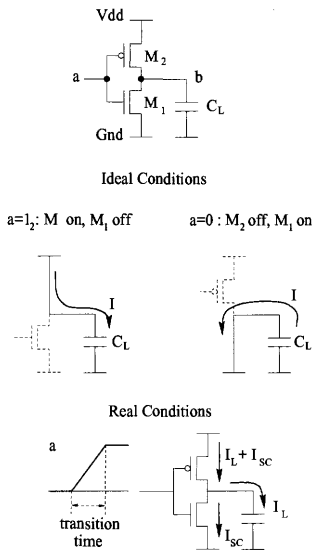

Figure 4.2: Short circuit current in CMOS circuit.

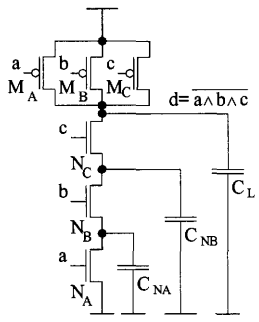

Figure 4.3: Parasitic capacitances in a 3-input NAND.

Eq.4.1 is often used to estimate the power dissipated in a circuit, but it doesn't take into account several other very important factors strictly related one to each other.

4.2.1 Parasitic Capacitances

The term C_{L_i} represents in first approximation the fanout of the gate. In fact this value can vary a lot with the structure of the gate itself. A more accurate model for C_{L_i} should consider the parasitic capacitances of the inner transisto rs. In Fig.4.3 the configuration of a 3-input NAND gate is shown. As we can see from this picture, there are particular configurations of the inputs for which the total load capacitance is different from C_L. As an example, lets suppose we have as input the vector a=0, b=1, c=1. Transistor N_A is off, while N_B and N_C are on. Therefore the current flowing from M_A should charge not only C_L but also the parasitic capacitances C_{NA} and C_{NB} and the actual load becomes $C_L + C_{NA} + C_{NB}$.

4.2.2 Short Circuit Current

Another determinant factor influencing the power dissipation is the so called *short circuit* current. The short circuit current occurs when there is a direct path from power supply to ground [117].

In Fig.4.2 the behaviour of a simple CMOS inverter is presented. We suppose that the power supply voltage satisfies the condition $V_{DD} > V_{TN} + |V_{TP}|$. In the ideal case it never occurs that there is a path from power supply to ground, because the PMOS and the NMOS are never on at the same time. This means that the input signal changes its state instantaneously.

In the real conditions there is a time interval, the *transition time*, when both the transistors conduct, therefore a current flows through the circuit. The width of the transition time depends on the slope of the input signal, which is strictly related to the load of the gate.

4.2.3 Hazard Generation and Propagation

In eq.4.1 we introduced the transition density η for the output signal of a gate. It is obvious that a thorough analysis of the circuit's behaviour is necessary in order to have a good estimate for this parameter.

The evaluation of the Boolean function describing the gate supplies a good baseline for this estimate. However it is not enough by itself due

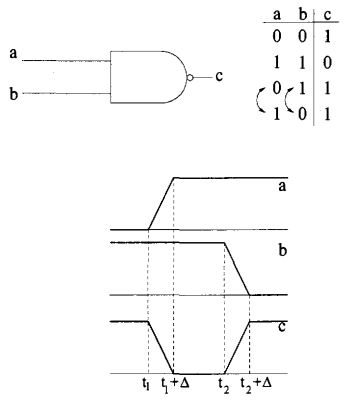

Figure 4.4: Glitch generation due to different delays at the inputs.

to the generation of hazards in the circuit increasing the actual value of the transition density. The power diss ipated due to hazard generation and propagation is almost the 30% of the total power dissipated by the circuit.

Hazards are generated when two or more input signals switch between configurations producing the same gate output value but causing a temporary change to its complementary value.

In Fig.4.4 we present the case of a 2-input NAND gate. As we can see from its truth table the configurations (0,1) and (1,0) yield the same output value 1.

Lets suppose the signal a switches from 0 to 1 at time t_1, while b switches from 1 to 0 at t_2, with $t_2 > t_1$. For the sake of simplicity we suppose the transition time Δ is the same for both signals and that no delay is introduce d by the gate. Before t_1 and after $t_2 + \Delta$ the output value is 1 as from truth table. In the time interval from $t_1 + \Delta$ and t_2 however the output switches from 1 to 0, because the (1,1) configuration is present at the input. Therefore a gl itch is generated, whose width depends on the delay between the two input signals.

The analysis of the hazard propagation is quite difficult, because the real behaviour of the transistors should be considered [116].

The gate presents a sensitivity to the input time misalignment and can be charachterized by a low pass behaviour with respect to the glitches propagation [131]. The main reason for this is that the transistors require a certain amount of time to switch from the off to the on condition. If the delay between the inputs is small enough the output will never change, because the transistors have no time to switch.

Thus it is not said that a hazard generated at the input of a circuit will be fully propagated until the output.

4.2.4 Leakage and Static Power Consumption

Beside the short circuit current two other current factors should be considered, but which can be ignored because they represent just marginal effects with respect to the dynamic and short circuit currents.

The first term is the *leakage* current. It consists of two components: leakage currents through reverse-biased diodes at the transistor drains and sources and subthreshold leakage through the channel when the transistor should be cutoff. This term is technology-dependent and is larger for processes with low threshold voltage. Once the process for the fabrication is chosen the leakage power consumption cannot be changed. However for a well-designed circuit it represents less than 1% of the switchin g power consumption.

The second factor is the static power consumption due to steady state operation. This term depends on the choice of the logic style for the logic cells, therefore it can be completely omitted for CMOS circuits.

4.2.5 Input Statistics

The parameters affecting power dissipation introduced in the previous sections depend on the structure of the circuit and on the physical behaviour of its components.

The statistical properties of the input signals play a fundamental role in power dissipation. The parameter affected by these features is the transition density.

If the input signals have high probability to switch their state then the transition density of a gate will consequently change depending on

its functionality.

A signal can be characterised with four probability functions:

- p_{00} probability of keeping the logic level at 0
- p_{01} probability of transition from 0 to 1
- p_{11} probability of keeping the logic level at 1
- p_{10} probability of transition from 1 to 0

These four probabilities can be reduced to two static probabilities

- $p_0 = p_{00} + p_{10}$ probability of being at the 0 level
- $p_1 = p_{11} + p_{01}$ probability of being at the 1 level

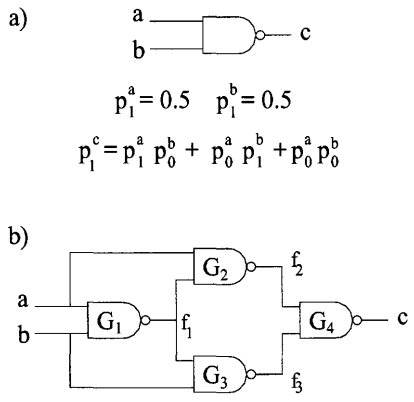

Figure 4.5: Propagation of input's probabilities in a combinational circuit.

The probability of an output signal for a generic gate can be easily obtained using its truth table. As an example lets consider a 2-input NAND gate and suppose that the two inputs have the same probability $p_1 = 0.5$. The probability of the output to be 1 is given by the expression:

$$p_{o_1} = p_0^a \cdot p_0^b + p_1^a \cdot p_0^b + p_0^a \cdot p_1^b = \frac{3}{4}$$

The calculation results only slightly more complex if we want to consider the transition probability of the gates. Lets suppose that for both the input signals it holds: $p_{00} = p_{11} = \frac{1}{6}$ and $p_{01} = p_{10} = \frac{2}{6}$.

Then the probability p_{01} for the output is given by:

$$p_{01} = p_{11}^a \cdot p_{10}^b + p_{10}^a \cdot p_{11}^b + p_{10}^a \cdot p_{10}^b = \frac{2}{9}$$

and the probability p_{10} is:

$$p_{10} = p_{01}^a \cdot p_{01}^b + p_{11}^a \cdot p_{01}^b + p_{01}^a \cdot p_{11}^b = \frac{2}{9}$$

The resulting transition probability is $p_{tr} = p_{01} + p_{10} = \frac{4}{9}$, which is less than the transition probability of the input signals.

This evaluation procedure for η is very attractive, because it is very simple and can be accomplished quickly. However the dependence of the transition density on the input statistics is far more complex than the simple evaluation of a Boolean functi on basically due to correlation. The correlation among inputs can be both spatial and temporal and can affect the circuit behaviour at several levels.

In sequential circuits the state variables are strongly temporarily correlated and this makes power estimation hard to perform.

But even in combinational circuits the correlation can have a deep impact on power consumption. In Fig.4.5 b) a simple circuit composed by four 2-input NAND gates is shown. The gate G_2 receives as inputs the signal a and the output f_1 o f G_1. Since $f_1 = \overline{a \wedge b}$ they are strongly correlated. The same happens for gates G_2 and G_3. Therefore we cannot simply propagate the input probabilities using the NAND Boolean function, because it leads to erroneous results.

At higher level input probabilities can be propagated through macroblocks, like adders and multipliers. In this case we consider the information at the word level, i.e. we consider the statistic parameters of the input words.

The assumption usually made in the simplest case is that the inputs are uncorrelated and uniformly distributed. However studies made on DSP architectures for image processing show that the output bits of such blocks can be brought into relation with the w ord statistics.

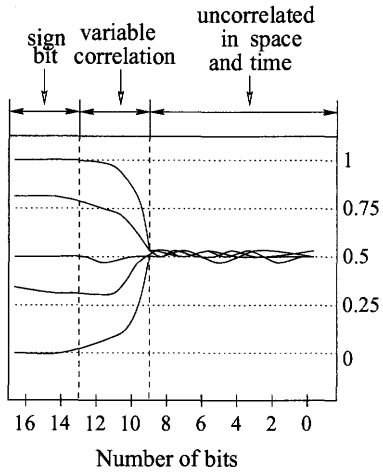

Figure 4.6: Dependance of bit statistics from the word length.

In Fig.4.6 a typical plot for bit transition probability is reported with respect to the word length.

As we can see the output word can be split in three sections corresponding to different correlation between the bit-level and the word-level statistics. From the LSB to a certain position the bits have a transition probability of $\frac{1}{2}$, which is i ndependent on the word statistics. The MSB's behaviour strongly depends on word statistics, while in the middle the correlation gradually increases towards the MSB's.

This observation led to the *dual bit type* and similar models for describing the behaviour of a circuit for power estimation [139].

¿From the previous analysis it clearly results that there are a lot of factors to be considered for power estimation. The method proposed in literature attempt to obtain the best results making some assumptions on the circuit and on its inputs so that the estimation can be simplified. In the following we briefly review the most used techniques, trying to point out their faults and benefits.

We focus our attention on gate-level and RTL techniques, since they are the most studied. The transistor level approach is mainly used

to study the phenomena involved in power dissipation even because it would require too many resources both in time and h ardware to perform power estimation.

4.3 Gate Level Techniques

Before describing the most common algorithms introduced for power estimation, lets clarify the meaning of two fundamental parameters: the spatial and temporal correlations, since, as we said, they play a relevant role in the determination of the accuracy of the proposed algorithms.

By *temporal* correlation we mean the dependence between the current signal value and its previous values, while by *spatial* correlation we mean the dependence due to reconvergent fanout regions. By input data dependence we mean the dependence of the input signal due to their generation source.

In the previous paragraph we introduced some concepts concerning the transition density at a gate node and the probability of the gate inputs. In fact the average number of transitions at the circuit nodes can express the average power consumption of the circuit. Therefore a variety of power estimation methods have been developed mainly trying to obtain a good estimate for these transition densities. Existing approaches to power estimation can be divided into two categories: *simulation-based* methods and *probabilistic* methods.

4.3.1 Simulation-based Methods

Simulation-based methods can model very accurately the spatial and temporal correlations present at the input and internal signals [132]. However they suffer from the significant drawback that they require a trace that accurately matches the operating conditions of the system. When the objective is to characterize a module that can be used in a variety of conditions, simulation-based methods may fail to yield the required precision.

When a trace reflecting the spatial and temporal input correlations is available, the actual switching capacitance can be computed in a trivial

way by simulating the circuit and directly computing the expression:

$$C_{eff} = \sum_k C_k T_k$$

where T_k represents the total number of transitions at node k. Using simulation, this computation is straightforward since the waveforms at each node can be obtained from the input waveforms. However, it may be too time-consuming if the trace is very long.

Methods have been proposed to speed up this computation, without incurring in a significant loss of precision [121]. One such possibility is the *random sampling* [121, 143], where only a subset of the transitions present in the trace is used. More sophisticated methods include sequence compaction and synthesis [124, 123]. Trace approaches aim at accurately reproducing the input correlations with traces that are much smaller than the initial one. Even though the problems of input compaction or the determination of the input sequence yielding the maximum dissipated power have assumed increasing importance, they represent a whole independent research branch and therefore will not be considered in the following.

4.3.2 Probabilistic Methods

Probabilistic methods do not require the existence of a trace and can be used to characterize circuits in a manner which is independent on the inputs presented [144]. However, the most efficient approaches presented up to date allow only for very simplified models of temporal and spatial correlations and, therefore, they limit the accuracy of the estimates. Other more precise approaches are computationally inefficient but they perform better since they use an exact model of the circuit characteristics. To extend the range of applicability of these methods without increasing too much the computational load an approximated model can be used.

The simplest approach is based on the assumption that the primary inputs are uncorrelated in time and space. The user may specify a probability P_1 that a given bit is at 1, which under the used independence assumptions, gives the probability of that bit transition as $2P_1(1 - P_1)$.

This modelling is crude and even an exact computation of P_1 will yield results very far from the actual value of dissipated power, as evaluated in the presence of spatial and temporal correlations.

The probabilistic approaches usually exploit the simplicity of the Boolean function to propagate and evaluate the transition probabilities at each node of the system. However, as we mentioned in the previous section, phenomena like glitches and reconvergent paths don't allow the simple evaluation of these functions to obtain an accurate result, but they require a more complex model of the gate to be incorporated in the function evaluation. In particular the above methods can be classified according to assumed gate delay model in the following categories:

- the *zero-delay* model, where only functional transitions are taken into account:
- the *real-gate delay* model, where both functional and spurious transitions are considered [145].

One method used to evaluate temporal correlations under the general delay model is the *polynomial simulation*, PS [146]. The switching activity of a signal is represented by four probability polynomials corresponding to the signal staying low ($0 \rightarrow 0$) or high ($1 \rightarrow 1$), a rising transition ($0 \rightarrow 1$) or a falling transition ($1 \rightarrow 0$). These probability polynomials are denoted p_x^{00}, p_x^{11}, p_x^{01}, p_x^{10} respectively.

The complexity of the polynomials may be dramatically reduced if the variables are substituted by their probability values. But this leads to an exact solution only if there is no spatial correlation between the variables, i.e. in a tree network. In the p resence of reconvergent paths, an exact computation of the switching activities requires to express each local polynomial group of an internal node at instant t as a function of the primary input nodes and this can be very time-consuming. The PS method proposes to limit the depth in terms of logic levels, when computing the activity, considering the spatial correlation: a parameter l controls the compromise between speed and accuracy. This is based on the idea that spatial correlation between internal signals is more important when reconvergent paths meet within a few logic levels. In the PS method, the variable substitution is applied only to the variables

which are not in a reconvergent path, of if reconvergent path meet after l logic levels. The active variables which converge before l levels are kept.

For a large l, the PS variable substitution approach may keep some small errors in presence of multiple reconvergences. However only single reconvergences are detected by the PS algorithm.

It processes one gate at a time, scanning from the primary inputs to the primary outputs of the circuit. For each gate g_i an ordered list of the possible transition times of its inputs is first obtained. Then possible transitions at the outputs of the gate are derived, taking into account transport delays from each input to the gate outputs.

The key operation of the PS algorithm is the local gate computation. The probability polynomial at the gate output depends on the gate Boolean Function and the probability polynomial of each gate input. This leads to a series of polynomial additions and multiplications, whose total number depends on the local Boolean Function handling. Furthermore the complexity of the polynomial operations depends on the input polynomial size, which is a function of the number of active variables considered.

The number of different techniques proposed to be used for power estimation is very high. Since our own developed method is an RTL method, we just give a superficial and brief description of them.

Other than the PS method most of the used techniques rely on the features of the Binary Decision Diagrams, in a variety of modifications, in order to split the circuit in such a way that reconvergent paths and spurious transitions can be reconducted to Boolean function evaluations [126, 134, 138, 140]. Obviously this generalization leads to an increase of complexity, but in any case usually less than considering complex gate models.

Other interesting techniques exploits concepts from information theory, like the *entropy*, relating them to the behaviour of the circuit [142, 141]. These techniques try to completely avoid the information about the connectivity of the gates using only the probabilistic features of the inputs. For this property they can be placed halfway between pure gate-level techniques and RTL methods. However the methods proposed up until now don't allow a very good accuracy of the results.

4.3.3 Sequential Circuits

Power and switching activity estimation for sequential circuits is significantly more difficult than combinational circuits, because the probability of the circuits being in any of its possible states has to be computed. In the general case a sequential circuit can be split in two parts, one including the combinational circuits (for both next state logic and output logic) and the other constituted by the memory elements [137]. Therefore the inputs of the circuit can be thought as composed by the primary inputs of the system and by the present state variables. The next state of the circuit is uniquely determined by the next state logic starting form this input pair. Therefore the correlation between primary input and state variables is very high.

To compute the average switching activity we require the transition probabilities for the primary input and the static probabilities for the present state [115]. These later quantities are in turn correlated to each other, the correlation depending on the connectivity of the State Transition Graph of the circuit. The solution of the related equation in order to evaluate the static probabilities could be very expensive. Therefore the correlation among the state variables is ignored and their probabilities are usually computed solving the equations for the next state logic.

4.4 RT-Level Techniques

High level power estimation is a key task in current design flows and most of them target RTL estimation. High-level techniques can be split in two main categories: *top-down* and *bottom-up* approaches.

In the top-down approach a circuit is described only as a set of Boolean functions, without any knowledge about the number of gates or nodes or about the internal structure. Information about the circuit's activity is used to optimally decompose these functions and minimize power dissipation [122]. These techniques are useful to carry on the design of completely new blocks.

In contrast bottom-up approaches are very suitable when the design involves the reuse of previously designed blocks, whose low-level

architecture is therefore well known. In this case we can talk about *hard macro*, i.e. each block is considered as a self consistent functional unit with known inner structure. Since these techniques exploit the knowledge of the circuit low-level architecture they can usually achieve a better accuracy than the top-down algorithms.

At the RT level the modelling of spatial and temporal correlations at the module inputs is of critical importance. Ideally at this abstraction level one would compute word-level statistics from functional simulations and use them on library-stored models of power consumption. For an accurate estimate, this model must take into account both the correlations between the bits in a word and the correlation in time between words.

Even if the details of the specific implementation of a module are unknown, for the sake of efficiency a model that abstracts away part of the information for the module should be used. The most commonly used approach to determine the coefficients of the RTL power models is to simulate the module using random inputs and to adjust the model using linear regression. This procedure has several disadvantages:

- the model may be biased due to the input pattern dependence problem associated with simulation-based techniques:
- The model is independent of the topology of the circuit and may become inaccurate if the module input probabilities are not close to uniform: spatial and temporal correlations of the module inputs are not taken into account.

Focusing on *hard-macros* (macros for which a low-level description is available), two classes of characterization-based power models can be considered: *linear regression* and *lookup tables.*

4.4.1 Regression-based Models

Normally regression-based macro-models utilize as parameter the switching information gathered at the unit's boundaries. In principle the more detailed the considered switching information, the higher the accuracy of the model and the longer the model construction time. Keeping in

mind this intuitive trade-off we can consider two regression based models [147]:

- IOSW, a multi-parameter model that represents power consumption as a linear function of the bit-wise input-output activity,
- IOTR, a three parameter model that represents power consumption as a linear function of average input-output statistics.

Both models consists of a vector of fitting coefficients $\mathbf{c}=(c_0, c_1, ..., c_n)$ to be multiplied by the actual values of the n independent variables to obtain the corresponding power estimate.

For IOSW the independent variables are the transition flags of each input and output bit. Hence $n = n_{in} + n_{out}$. IOTR, instead, uses only $n = 3$ independent variables, representing the average input signal probability, P_{in}, the average input transition probability, D_{in}, and the average output transition probability, D_{out}. Both models are characterised by finding the least-square solution of the following over-defined system of linear equations:

$$X \cdot c = P$$

where X is a $N \times (n + 1)$ matrix of values taken by the independent variables during the low-level simulation (N represent the training step), P is a vector of corresponding power values and C is the vector of fitting coefficients. Due to statistical nature of least-squares fitting, the number of training experiments has to be much higher than the number of fitting coefficients. A suggested value $N = 20 \times n$ has been found in literature.

Once the coefficients c has been optimised the evaluation phase is accomplished by running a simulation, gathering the statistic features of the system's inputs constituting the model's inputs, and then evaluating the model associated function. In the simple case of IOTR, this function is:

$$P_{est} = c_1 P_{in} + c_2 D_{in} + c_3 D_{out}$$

.

4.4.2 Table-based Models

The table-based models are usually based on three-dimensional tables. Each entry in the table represents the average power consumption of the unit for given workload conditions. The indexes of the table are the previously mentioned variables P_{in}, D_{in}, D_{out}.

The characterization phase of the table consists of performing several low-level power simulations with input streams representing different workload conditions. For each stream the actual values of P_{in}, D_{in}, D_{out} are computed and used to address the lookup table entry, where the corresponding value of P has been stored.

The space of workload statistics has to be discretized to trade off characterization time for accuracy.

The evaluation phase is straightforward: during RTL/behavioural simulation the input-output statistics for the macro (P_{in}, D_{in}, D_{out}) are collected and used to perform a table lookup. The returned power estimate is the power value stored in the entry whose coordinates are the closest to the actual value.

4.4.3 Sequential Circuits

As for gate-level techniques the design of accurate power macro-models for sequential RTL modules has been usually considered a difficult problem due to the dependence on internal states.

All high level power models for combinational macros rely on the assumption that the power consumption of a CMOS circuit is mainly related to its input activity. This assumption does not hold any longer for sequential macros, where the power consumption corresponding to a given input transition may depend on arbitrarily long previous history, i.e. the internal states of the circuit. State dependence is the main issue that has to be addressed when building a power macro model for a sequential circuit. On the other hand, the effect of the initial state on the average power consumption of a sequential macro decreases when the number of input patterns gets larger. This observation suggests that, as long as average power consumption is the target, state dependence could be neglected. Most of the recent macro-modelling techniques have

been developed for combinational circuits but they have been found very suitable for sequential circuits also, especially if the circuit size and number of inputs are large. In this case performing the characterization phase of the model over very long training sequences the state variables of the circuit can be thought as being stationary and their influence is in average neglectable.

4.4.4 Examples

A typical example of table-based technique is presented in [119]. In this paper a 3 variables model is used, the variables being P_{in}, D_{in}, D_{out}. In order to avoid confusion we define as *input vector* the set of N inputs of the circuit. We also define as *input stream* a sequence of M input vectors.

For the characterization phase first n equispaced values both for P_{in} and D_{in} are selected. For each of these n^2 pairs m different input streams consisting each of k input vectors are generated. The only constraint on the input streams is that the average probabilities and transition densities are equal to the fixed (P_{in}, D_{in}) pair. After the system has been simulated over an input stream the average dissipated power and the average output transition D_{out} are collected and stored. In this way the complete characterization of the table requires $k \times m \times n^2$ simulations, while the table size is $m \times n^2$. In order to obtain a more accurate result during the evaluation phase the authors introduced a two-step interpolation algorithm. Given the input couple (P^*_{in}, D^*_{in}), the nearest neighbours are found. For each of the neighbours a list of m values of D_{out} and P_{est} is explored and the two pairs (D^a_{out}, P^a_{est}) and (D^b_{out}, P^b_{est}) nearest to the measured values (D^*_{out}, P^*_{out}) are used to evaluate the linear interpolation between P^a_{est} and P^b_{est}:

$$\overline{P}_{est} = \frac{P^a_{est} - P^b_{est}}{D^a_{out} - D^b_{out}} D^*_{out} + \frac{P^b_{est} D^a_{out} - P^a_{est} D^b_{out}}{D^a_{out} - D^b_{out}}$$

After this first step four linear interpolated values (one for each neighbour pair (P^*_{in}, D^*_{in})) are obtained. The second step involves a bilinear interpolation algorithm in order to find the final value.

The paper does not report any table with comparative results over benchmark circuits. Relative percentage errors ranging from 6.84% to 8.01 % are said to be obtained. Moreover the method has not been applied to sequential circuits.

A more interesting paper is [118]. In this paper two techniques are presented. The first is an enhanced table-based algorithm, while the second is an analytical method.

The enhanced table-based technique relies on the extension of the number of model's input variables to four in order to improve the accuracy. Beside the canonical P_{in}, D_{in}, D_{out} variable a fourth parameter taking into account the spatial correlation is considered. For instance the variable SC_{in} is defined as follows:

$$SC_{in} = Prob\{x_i \wedge x_j = 1\}$$

where x_i and x_j are the sequences corresponding to input i and j of the system. Practically it is obtained performing the bit-wise *and* operation between the i-th and the j-th sequences. Obviously the spatial correlation of two random variables is not defined in this way, but this variable approximate it the better the longer the inputs number.

The second proposed method tries to avoid the need of a large memory to store the measured results and create the table. The measured values of the four variables are temporarily stored during characterization phase. Once the system has been simulated over all the input patterns, a user-specified function is fitted to the collected data by mean of a least-square algorithm: after this step the collected data are no longer retained. The method would strictly resemble the regression techniques if we choose as function a first order polynomial. The results reported in the paper show that this choice is not very good, with average error comparable with previous table-based methods and with very high maximum errors. In order to improve the accuracy the authors chose a complete second order polynomial in the four variables, with the option to use a cubic polynomial in the critical cases.

In the following tables we report the results proposed in the paper

Circuit	ϵ%	Max ϵ	ϵ%	Max ϵ
c432	5.56	-27.8	3.46	25.75
c499	5.96	46.3	12.03	48.5
c880	6.7	50.6	6.4	53.88
c1355	6.19	33.56	13.06	53.98
c1908	3.85	31.17	4.66	30.39
c2670	7.8	37.53	6.09	46.16
c3540	7.5	44.19	4.66	40.26
c5315	3.48	29.03	3.53	32.63
c6288	9.6	43.15	9.68	54.03
c7552	8.95	-45.79	8.23	46.32

Table 4.1: Comparison between the results obtained with the 4D table and with the quadratic polynomial on combinational benchmark circuits.

for the combinational circuits[1]. The error ϵ is the absolute relative error:

$$\epsilon = \frac{|P_{avg} - P_{est}}{P_{avg}} \times 100$$

where P_{avg} is the actual value of the dissipated power and P_{est} is the model's estimate. The reported parameters are the maximum absolute relative error and mean absolute relative error.

This algorithm has been also tested on sequential ISCAS89 benchmark circuits. In the following table the results are reported:

Even if for the combinational circuits the results are more or less the same as those obtained with the table-based approach, this technique allow a big memory saving. Moreover the results proof that it is very suitable for sequential circuits too.

4.4.5 Remarks

Both regression-based and table-based approaches can be also called *census macro-modelling* techniques, since at every simulation or training

[1]For a more detailed description of the benchmark circuits see the following section.

Circuit	Avg ϵ	Max ϵ	Std. dev
s349	7.66	46.31	7.85
s344	6.48	38.95	7.59
s400	2.65	29.95	2.68
s713	1.65	9.23	1.53
s832	1.39	13.15	1.54
s526	6.2	42.95	7.57
s1494	5.0	34.1	5.09
s1488	3.65	33.51	3.27
s1423	5.29	19.99	4.54
s386	1.49	16.98	1.59
s382	2.56	29.52	3.7
s420	0.94	4.49	0.68
s641	1.71	8.69	1.22
s1196	0.3	3.22	0.43
s510	0.11	0.58	0.1
s953	0.21	2.12	0.29
s298	2.56	9.97	2.09
s1238	0.58	16.5	2.0
s444	2.57	9.99	2.55
s820	1.58	9.6	1.63
s838	0.98	4.04	0.87
s5378	1.11	5.22	0.77
s9234	2.9	21.2	2.56

Table 4.2: Results obtained with the quadratic polynomial for ISCAS89 sequential circuits.

step the activity information has to be collected. There are two problems related to census macro-modelling:

- Input data statistics must be collected for every cycle. The statistics gathering overhead is large and hence it slows down the RTL simulation. As an example, for a 16 x 16 bits multiplier the RTL

simulation requires only one instruction, while statistics gathering requires tens or hundreds of cycles. This shows that census macro-modelling is very costly, especially when the input vector sequence is long.

- Power macro-models are developed by using a training set of input vectors. The training set satisfies certain assumptions such as being pseudo-random data, speech data, image data, etc.. Hence the macro-model is biased, meaning that it produces very good results for the class of data behaving similarly to the training set, otherwise it performs poorly.

Several techniques have been introduced to reduce the aforementioned problems. Two of them are the *sampler macro-modelling* and the *adaptive macro-modelling* [127]. The former uses a simple random sampling technique to reduce the number of cycles during which the data statistics are collected, without significant loss of accuracy. The latter relies on regression analysis combined with gate-level simulation on a small number of cycles to correct the static macro-model estimate and therefore the model can be thought as self-adjusting.

4.5 The Proposed Method

4.5.1 Motivation

¿From the previous sections it appears that a lot of efforts have been spent trying to achieve good results in power estimation without requiring huge computation resources. Focusing our attention on RTL techniques, even though the results obtained with some methods could be considered accurate enough, the methods themselves suffer from two main defects:

1. most of them do not consider correlations at the primary inputs, or if they do, they use simplified models;

2. until now all the presented methods require to run simulations even in the evaluation time. This especially penalizes these techniques, because simulations slow down the design process.

The goal of our technique is to achieve a better accuracy including in the model more features extracted from the primary inputs and, most of all, totally avoid the need of simulations during the evaluation phase.

4.5.2 Input Metrics

One of the most challenging aspects in the construction of an equation-based macromodel is the choice of the model's input parameters, or metrics. To help obtain good estimates, these metrics should capture the features that are primarily responsible for a system's dissipation and output statistics. This section describes and motivates the metrics of our macromodel.

Similar to the approaches in [119, 118], our model uses the average input probability P_{in} and the average input transition density D_{in}. Given an input stream:

$x = ((x_{11}, x_{12}, \ldots, x_{1M}), (x_{21}, x_{22}, \ldots, x_{2M}), \ldots, (x_{N1}, x_{N2}, \ldots, x_{NM}))$

these metrics are defined as:

$$P_{in} = \frac{\sum_{j=1}^{M} \sum_{i=1}^{N} x_{ij}}{MN}, \tag{4.2}$$

$$D_{in} = \frac{\sum_{j=1}^{M} d_j}{M} \tag{4.3}$$

where M is the number of primary inputs to the circuit, N is the number of input vectors in the stream, and d_j is the transition density of the jth input bit. The transition density d_j is defined as the number of $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions per time unit for bit j. P_{in} corresponds to an average probability only if the 0's and 1's at each input are uniformily distributed. Nevertheless, this metric captures the input features effectively even if the input distribution is not uniform.

Input correlation plays a significant role in power dissipation, especially when the circuit is part of a wide datapath. We have therefore decided to incorporate input correlation into the metrics of our model. Specifically, we use two correlation metrics S_{in} and T_{in} to capture spatial and temporal correlations, respectively, of the inputs.

The spatial correlation metric S_{in} is defined as the average of the bitwise XNOR between all possible *channel streams* $x_i = [x_{1i}, x_{2i}, \ldots, x_{Ni}]$

and $x_j = [x_{1j}, x_{2j}, \ldots, x_{Nj}]$ in the stream x:

$$S_{in} = \frac{\sum_{j=1}^{M} \sum_{k=1}^{M} \sum_{i=1}^{N} x_{ij} \oplus x_{ik}}{N \times M \times (M-1)}$$

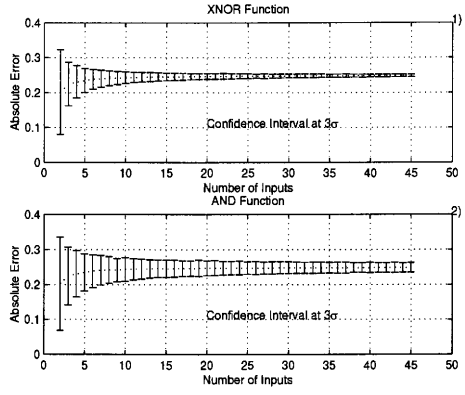

Figure 4.7: Comparison between AND and XNOR functions.

This choice was motivated by the following observation. The metric SC_{in} in [118] was defined as the average of the bit-wise AND between all pairs of input channel streams x_j and x_k. According to this definition, two channel streams x_j and x_k will be highly correlated only if they comprise matching 1's in the two streams. Therefore, the AND operator will miss correlated channel streams that comprise matching 0's. Our choice of XNOR captures correlations of both 1's and 0's, however.

In Fig.4.7 a comparison between the XNOR and AND operators is shown. The curves represent the mean absolute error of the operator's estimate with respect to the actual value of the mean correlation coefficient for the input stream. The x axis reports the number M of primary inputs of the system. As we can see the curves are practically identical, which means that they are perfectly equivalent as correlation estimators. As the number of input increases the mean error also increases, until for $M \simeq 8$ the curves reach a steady value. After this point the error can almost be considered as systematic. For each sample in the curve the

relative confidence interval is plotted. The confidence interval is taken at 3σ. We can see that for a low number of inputs the confidence interval is more or less the same for both the operators. But as the number of inputs increases the confidence interval (i.e. the standard deviation) of the XNOR becomes three times lower than for the AND estimates. This means that especially for large number of primary inputs our operator is more precise.

The temporal correlation metric T_{in} captures input features that are missed by S_{in}. Given a channel stream of length N, we take the convolution of a *window* of length L from the stream. Each term in the convolution represents the number of times that two signals are both simultaneously high. Each term indicates how well the chosen subset is reproduced in the sequence, and therefore it provides an estimate of the signal temporal correlation. The main issue in the definition of T_{in} is the choice of L, since a value that is either too large or too small may result in missing correlation information. In [123] the authors use a similar window technique to estimate the temporal correlation of input streams with arbitrary distributions. Their study shows that a suitable value for L is 10. The final value of T_{in} is the average over all the inputs of the convolution means:

$$T_{in} = \frac{\sum_{j=1}^{M} \sum_{L-1}^{N-L+1} (w_j \otimes x_j)}{N \times M}$$

where w_j is a window of length L in the channel stream x_j.

An important feature of our model is that it does not rely on D_{out}, thus avoiding any time-consuming simulations during the estimation phase. D_{out} provides valuable information about a circuit's dissipation and has been included in all previous macromodels. Tests made on the ISCAS-85 benchmark circuits show that in most cases the two quantities are related extremely well. Even though our macromodel does not use D_{out}, it still achieves remarkable accuracy, superior to that of previous macromodels.

4.5.3 Macromodel Characterization and Evaluation

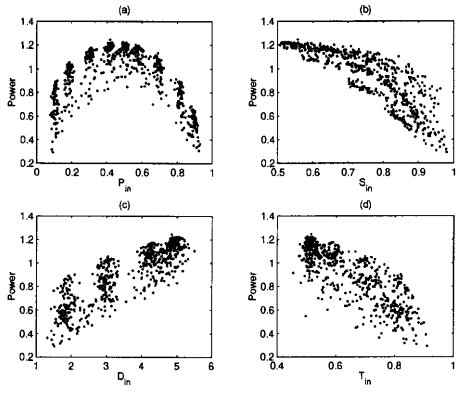

Figure 4.8: Power dissipation of C1908 with respect to the four input metrics of our macromodel.

To avoid the large memory requirements of a large look-up table, our macromodel uses a nonlinear function

$$PD_{avg} = f(P_{in}, D_{in}, S_{in}, T_{in}) \tag{4.4}$$

to estimate the average dissipated power. For the sake of efficiency in the estimation phase, we opted for the use of simple functions.

Figure 4.8 gives the dissipation of the ISCAS-85 circuit C1908 and provides evidence supporting a low-order polynomial dependency of PD_{avg} on the four input metrics of our macromodel. For example, the dependence on P_{in} appears to be quadratic, while that with respect to D_{in} seems almost linear. We therefore used a 3rd degree complete polynomial as a template function for our model. Such a function requires the calculation of 35 coefficients. A similar function can be used to estimate the output statistics $P_{out}, D_{out}, S_{out}$, and T_{out} of each block.

The characterization phase of our model is quite straightforward. Given a circuit, we first specify a value for the metrics P_{in} and S_{in} in the range [0.1, 0.9]. A sequence of inputs streams is subsequently generated using a random number generator. Unless the circuit has very few inputs, it is very difficult to control all four metrics simultaneously. We can nevertheless generate a sufficiently high number of streams to

ensure that results are collected for a reasonable subset of values for D_{in} and T_{in}. To that effect, in our empirical evaluation we used 650 streams, each consisting of 200 input vectors.

If the input patterns are wisely chosen, it is possible to cover most of the range of all the variables. Therefore the model can be considered to a large extent independent from variations of the input statistics, i.e. almost unbiased. This feature is very interesting, because it means that this model can be used over a wide range of applications, where input statistics can be completely different, without requiring a re-characterization to adjust the coefficients.

The next step in the characterization phase of our model is to obtain the circuit's dissipation on each input vector. These values can be computed using any available power estimation approach. Reference values for power dissipation were obtained using the switch-level simulator SLS [107]. The SLS simulator has three working levels:

1. purely logic simulation based on network topology and transistor types, without considering the actual circuit parameters;

2. logic simulation based on actual circuit parameters, like transistor dimensions, interconnection resistances and capacitances, which determine the logic state of the system;

3. logic and timing simulation based on actual parameters as in the preceding level, but delays are also evaluated.

In particular we built our own library including all the gates used in the benchmark circuits. For each of these gates, a transistor level description was used with transistor models obtained from the AMS foundry. Therefore, the SLS simulator working at level 3 took into account very thorough timings for the gates.

We also used the Spice simulators to test our method with a family of ripple-carry adders. The goal of these tests was to demonstrate that a good accuracy can be achieved even if in case of real data. The problem associated with real data is that unknown or not considered effects can affect the dissipated power. Therefore the data can show unpredictable or very complex behaviour than data obtained from higher level simulators.

A very interesting feature of our macro-modelling techniques is that it can be effectively used with a wide variety of simulators, depending on the accuracy we want in the estimates.

In [118], characterization was performed using Monte Carlo simulation with general delay models, thus resulting in very accurate reference values. During the estimation phase, however, zero-delay simulation was used for the sake of efficiency. Thus, the estimation potential of the model was compromised, since the input data were not accurate enough.

Once the reference values have been obtained, a standard nonlinear fitting algorithm can be used to compute the coefficients of the model. The total time required for the characterization of each model from the ISCAS-85 circuits is less than 2 minutes, including simulation and coefficient computation.

The evaluation of the model is really simple and has the advantage that it does not need to perform any simulations during estimation. Once the statistics of the primary inputs are extracted they are fed into the model template function, which outputs the power estimate.

The same template function with the same approach used for power characterization can be exploited to obtain models for the output metrics $P_{out}, D_{out}, S_{out}, T_{out}$. If the system is described as a cascade of macro blocks, the information regarding the input metrics can be propagated throughout the system involving only simple function evaluation. For the design of highly-complex systems with several different pre-defined blocks, our technique can provide fast and accurate estimates, thus enabling designers to explore different block arrangements in real time.

Alternative macromodelling techniques that require even zero-delay simulation during estimation are prohibitively time-demanding, because they must perform a complete simulation each time a block is moved.

4.5.4 Results

This section presents results from the application of our macromodel to several ISCAS-85 and ISCAS-89 benchmark circuits and to a family of ripple-carry adders. ISCAS benchmark circuits [106] are a set of combinational and sequential circuits with a wide range of primary inputs and number of gates. They are used to compare results obtained in cir-

Circuit	G	PI	PO	Max ϵ	Avg ϵ	$\sigma(\epsilon)$
C432	160	36	7	28.7%	3.0%	8.6
C880	383	60	26	10.0%	1.5%	1.8
C1355	546	41	32	19.5%	1.3%	2.5
C1908	880	33	25	12.0%	1.5%	2.3
C3540	1669	50	22	9.2%	1.5%	2.1
C5315	2307	178	123	5.5%	0.9%	0.6

Table 4.3: Accuracy of power estimates.

cuits optimisation, retiming and power estimation, allowing to test the various models under different conditions.

The evaluation phase has been run over 500 different input streams, each consisting of 200 input vectors. For each of the 500 cases the values of power and absolute relative error have been recorded. The error has been calculated with the following expression:

$$\epsilon = \frac{|P_{avg} - P_{est}}{P_{avg}} \times 100$$

It has to be noticed that the characterization and evaluation processes has been performed exploiting the built-in functions of MATLAB. Therefore all the algorithms are well established and commonly used in applications like system optimisation, image processing, etc.. Apart from the building of the digital library necessary to work with SLS, the efforts required to the user to successfully apply our method are quite a few.

Table 4.3 gives the accuracy of the power estimates obtained with our model for several ISCAS-85 circuits. The first two columns give the names of the circuits and their corresponding gate counts. Columns three and four give the number of primary inputs and primary outputs, respectively. The last three columns give the maximum, average, and standard deviation of the absolute relative error ϵ.

All estimates were obtained from the template function within a few seconds. Apart from C432, the maximum error was no more than

Circuit	Metric	Max ϵ	Avg ϵ	$\sigma(\epsilon)$
C432	P_{out}	29.4%	3.0%	11.0
	S_{out}	6.6%	1.3%	1.2
	D_{out}	38.7%	3.9%	13.8
	T_{out}	27.1%	3.5%	9.4
C880	P_{out}	4.6%	7.8%	0.4
	S_{out}	2.3%	0.4%	0.1
	D_{out}	14.0%	2.2%	3.7
	T_{out}	12.7%	1.7%	2.5
C1355	P_{out}	6.8%	0.5%	0.5
	S_{out}	1.1%	0.2%	0.0
	D_{out}	5.0%	0.7%	0.5
	T_{out}	5.3%	0.5%	0.3
C1908	P_{out}	4.7%	0.9%	0.7
	S_{out}	1.9%	0.5%	0.1
	D_{out}	10.1%	1.7%	2.3
	T_{out}	9.1%	1.5%	1.6
C3540	P_{out}	5.4%	1.3%	1.1
	S_{out}	2.2%	0.4%	0.1
	D_{out}	19.7%	2.8%	5.9
	T_{out}	9.0%	1.9%	2.5
C5315	P_{out}	4.2%	0.6%	0.2
	S_{out}	1.0%	0.2%	0.0
	D_{out}	9.7%	1.7%	2.3
	T_{out}	5.3%	1.2%	1.0

Table 4.4: Accuracy of output statistics.

19.5%, while the average error was always less or equal 3%.

For the ISCAS-85 circuits given in Table 4.3, the equation-based approach presented in [118] appears to be less accurate than ours. In that paper, average and worst-case errors were in the ranges 3.5%–13% and 25%–53%, respectively. A direct comparison of the two approaches' accuracy based solely on the published results is impossible, however, because the zero-delay estimates in [118] are compared with reference values obtained using general delay models.

Table 4.4 gives the accuracy of our model for the output characteris-

Circuit	PI	PO	Max ϵ	Avg ϵ	$\sigma(\epsilon)$
1 bit	3	2	15.5%	3.1%	6.6
2 bit	5	3	14.7%	2.9%	7.0
4 bit	9	5	19.8%	5.1%	10.3

Table 4.5: Accuracy of adder power estimates with respect to Spice.

tics of the circuits in our test suite. These results mirror those obtained for power dissipation, showing that our technique could be used effectively to achieve fast and accurate results in the early stages of system design.

In addition to the benchmark circuits, we applied our model to simple ripple-carry adder circuits whose dissipation was estimated separately using Spice. Our simulations were performed using the same criteria for the input vectors as with the benchmark circuits. Our model was applied on three circuits: a 1-bit adder, a 2-bit adder, and a 4-bit adder.

To encompass parameterization, a fifth metric n must be included to the function of our model, which identifies the number of replicas (or number of input bits) in the circuit:

$$PD_{avg} = f(n, P_{in}, D_{in}, S_{in}, T_{in}) \tag{4.5}$$

Table 4.5 gives the accuracy of our macromodel with respect to Spice estimates. The estimates are not as accurate as in the case of SLS estimation. They are nevertheless quite close to Spice and were obtained within a few seconds. Worst-case absolute relative error never exceeded 20%. Average absolute relative error was at most 5.1%. The circuits used in this experiment were particularly glitchy, because XOR gates had been replaced with their NAND equivalent. Further improvements could be achieved by modifying the template function.

Table 4.6 gives the accuracy of the power estimates obtained with our model for several ISCAS-89 circuits. The first two columns give the names of the circuits and their corresponding gate counts. Columns three and four give the number of primary inputs and primary outputs, respectively. The fifth column reports the number of registers in the cir-

Circuit	G	PI	PO	Latch	Max ϵ	Avg ϵ	$\sigma(\epsilon)$
S208	104	10	1	8	6.2%	1.0%	0.7
S298	119	3	6	14	10.6%	1.3%	1.4
S344	160	9	11	15	8.7%	1.5%	1.6
S349	161	9	11	15	14.7%	3.2%	4.9
S382	158	3	6	21	3.5%	0.6%	0.3
S386	159	7	7	6	9.3%	1.7%	3.3
S400	162	3	6	21	4.0%	0.7%	0.3
S420	218	18	1	16	3.4%	0.6%	0.2
S444	181	3	6	21	5.3%	0.7%	0.4
S510	211	19	7	6	11.8%	2.3%	3.9
S526	193	3	6	21	4.3%	0.8%	0.4
S526n	194	3	6	21	3.6%	0.8%	0.4
S641	379	35	24	19	1.9%	0.5%	0.1
S713	393	35	23	19	2.1%	0.5%	0.2
S820	289	18	19	5	11.3%	2.8%	4.8
S832	446	18	19	5	11.1%	2.5%	3.8
S838	288	34	1	32	1.7%	0.4%	0.1
S953	395	16	23	29	5.4%	1.7%	1.5
S1196	529	14	14	18	6.1%	1.1%	0.8
S1238	508	14	14	18	5.3%	1.1%	0.7
S1423	657	17	5	74	5.2%	1.3%	1.0
S1488	653	8	19	6	13.8%	5.2%	3.7
S1494	647	8	19	6	11.3%	4.6%	3.5
S5378	2279	35	49	179	2.7%	0.7%	0.3
S9234	5597	36	39	211	9.7%	3.0%	4.8

Table 4.6: Accuracy of power estimates.

cuit. The last three columns give the maximum, average, and standard deviation of the absolute relative error ϵ for the estimates obtained with our macromodel.

The maximum error was no more than 14.7%. Moreover, for most of the circuits, the average error was less than 3.2%, while for S1488 and S1494 it was around 5%. For each circuit, averages were computed over 500 data points. For each point, data were collected by applying a

uniformly-distributed randomly-generated input stream of length 200.

4.6 Conclusion

In this chapter we presented a review of the most common techniques applied for power estimation and at the end we introduced out own proposed macro-modelling technique.

This technique can be easily used for both combinational and sequential circuits characterization. It yields very accurate estimates with average errors ranging from 3% for combinational circuits to 5% for sequential circuits. Tests made with zero-delay models as well as simulations at transistor level demostrated that this accuracy can be easily achieved even for almost real cases or for very simple models. This feature can be exploit when, for example, only rough estimate or very accurate results are required. In the former case the use of high level simulators allows a quick characterization phase, while in the latter case estimates very close to the real measures can be calculated.

The model has four input variables extracted from the input sequence. This variables take into account transition density, probability and correlations of the inputs in order to improve the accuracy of the results. Avoiding the use of any output metrics allows the technique to be used without the need of simulations during the evaluation phase, thus speeding up the process of power estimation.

A further enhancement of the method consists of the models for the output metrics. For this purpose the same template function used for power modelling is utilized. This extension allows the designer to easily propagate the metrics of the system's primary input throughout its component blocks. Therefore several structures can be quickly evaluated in order to satisfy the design constraints.

Even if the results are very good for almost all the circuits there are a lot of possible improvements to evaluate and problems to be solved. First of all an automatic template selection would allow even better performances and savings in computational resources, since for certain system lower degree polynomials can be selected, while for more complex structures other high order terms can be included. Without limiting

the model to polynomial templates, the application of more complex techniques for system recognition as well as data fitting constitute an interesting field of investigation. Moreover a deeper analysis involving the choice of the input patterns would extend the range of applicability of the models. A major problem concerns the propagation of output metrics through a cascade of macro-blocks. In fact it has to be studied how to merge the information coming from different blocks and how reconvergent paths among macro-blocks can influence the overall performances.

Further improvements will include software modelization, which is of primarily importance in systems where DSPs or μC are used. In fact, in these cases the underlying hardware structure is fixed, with specified performances regarding both speed and power consumption. Creating a model for each instruction or block of instruction, not only an estimate of the power cost of running programs is easily obtainable, but the software itself can be re-engineerized in order to optimise this cost.

Bibliography

[1] N.M. Wigley and G.A. Jullien , *On Modulus Replication for Residue Arithmetic Computations of Complex Inner Products*, IEEE Transactions on Computers, vol. 39, no.8, Aug. 1990, pp.1065–1076

[2] N.M. Wigley and G.A. Jullien, *Large Dynamic Range Computations over Sall Finite Rings*, IEEE Transactions on Computers, vol.43,no.1, Jan. 1994, pp.78–86

[3] T. Zeugmann, *Highly Parallel Computations Modulo a Number Having Only Small Prime Factors*, ICMP, vol.96, 1992, pp.95–114

[4] J.C. Bajard and L.S. Didier and P. Kornerup, *An RNS Montgomery Modular Multiplication Algorithm*, IEEE Transactions on Computers, vol.46, no.7, July 1998, pp.234–239

[5] E.D. Di Claudio and F. Piazza and G. Orlandi, *Fast Combinatorial RNS Processors for DSP Applications*, IEEE Transactions on Computers, vol.44, no.5, May 1995, pp.624–633

[6] G.A. Jullien, *Implementation of Multiplication, Modulo a Prime Number, with Applications to Number Thepretic Transforms*, IEEE Transactions on Computers, vol.29, no.10, Oct. 1980, pp.899–905

[7] M. Karpinski and I. SHparlinski, *Efficient Approximation Algorithms for Sparse Polynomials over Finite Fields*, Technical report TR-94-029, University of Bonn, Aug. 1994

[8] E.V. Krishnamurthy, *Error-free Polynomial Matrix Computations*, Springer-Verlag, New York, 1985

[9] M.A. Soderstrand and W.K. Jenkins and G.A. Jullien and F.J. Taylor, *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*, IEEE Press, New York, 1985

[10] G. Dimauro and S. Impedovo and G. Pirlo, *A New Technoque for Fast Number Comparison in the Residue Number System*, IEEE Transactions on Computers, vol.42, no.5, May 1993, pp.608–612

[11] M.A. Hitz and E. Kaltofen, *Integer Division in Residue Number Systems*, IEEE Transactions on Computers, vol.44, no.8, Aug. 1995, pp.983–989

[12] Mi Lu and J.S. Chiang, *A Novel Division Algorithm for the Residue Number System*, IEEE Transactions on Computers, vol.41, no.8, Aug. 1992, pp.1026–1032

[13] G. Alia and E. Martinelli, *A VLSI Structure for X(mod m) Operation*

[14] F. Barsi, *Mod m Arithmetic in Binary Systems*, Information Processing Letters, vol.40, no.6, Dec. 1991, pp.303–309

[15] R. Zimmermann, *Efficient VLSI Implementation of Modulo* $(2^n \pm 1)$ *Addition and Multiplication*, Proc. of the 14th Symp. on Computer Arithmetic, Adelaide, Australia, April 1999

[16] M. Bhardwaj and T. Srikanthan and C.T. Clarke, *A Reverse Converter for the 4-Moduli Superset* $\{2^n-1, 2^n, 2^n+1, 2^n+1+1\}$, Proc. of the 14th Symp. on Computer Arithmetic, Adelaide, Australia, April 1999

[17] M. Bhardwaj and T. Srikanthan and C.T. Clarke, *VLSI Cost of Arithmetic Parallelism: A Reverse Conversion Perspective*, Proc. of the 14th Symp. on Computer Arithmetic, Adelaide, Australia, April, 1999

[18] A. Bernal and A. Guyot, *Design of a Modular Multiplier Based on Montgomery's Algorithm*

[19] A. Hiasat and H.S. Abdel-Aty-Zohdy, *Design and Implementation of and RNS Division Algorithm*

[20] G.A. Jullien and W. Luo and N.M. Wigley, *High Throughput VLSI DSP Using Replicated Finite Rings*, Journal of VLSI Signal Processing, vol.14, 1996, pp.207–220

[21] N. Burgess, *Scaled and Unscaled Residue Number System to Binary Conversion Techniques Using the Core Function*, Proc. 13th IEEE Symp. on Computer Arithmetic, July 1997, pp.250–257

[22] P.R. Turner, *Fraction-free RNS Algorithms for Solving Linear Systems*, Proc. 13th Int. Conf. on Computer Arithmetic, July 1997, pp.218–224

[23] A.M. Eskicioglu and P.S. Fisher, *Image Quality Measures and their Performance*, IEEECOMM, vol.43, no.12, Dec. 1995, pp.2959–2965

[24] L. Khriji and M. Gabbouj, *Median-Rational Hybrid Filters for Image Restoration*, IEE Electronics Letters, vol.34, no.10, May 1998, pp.977-979,

[25] , L. Khriji and M. Gabbouj, *Median-Rational Hybrid Filters*, Intern. Conf. on Image Processing ICIP'98, Oct., 1998

[26] G. Ramponi, *The Rational Filter for Image Smoothing*, IEEESPL, vol.3, no.3, March 1996, pp.63–65

[27] J. Lee and V.J. Mathews, *A Fast Recursove Least Square Adaptive Second-Order Volterra Filter and Its Performance Analysis*, IEEESP, vol.41, no.3, March 1993, pp.1087–1102

[28] H. Leung and S. Haykin, *Detection and Estimation Using and Adaptive Rational Function Filter*, IEEESP, vol.42, no.12, Dec. 1994, pp.3366–3376

[29] C. Dick and F. Harris, *FPGA Interpolators Using Polynomial Filters*, Proc. of the 8th Int. COnf. on Signal Processing Applications and Technology, Sept. 1998

[30] G. Ramponi and G.L. Sicuranza, *Image Sharpening Using a Polynomial Operator*, Proc. 11th European Conference on Circuit Theory and Design, September,1993

[31] S. Kröner, G. Ramponi, *Design Constraints for Polynomial and Rational FIlters*, IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing, NSIP-99, Antalya, Turkey, June 1999

[32] , G. Ramponi, *The Rational Filter for Image Smoothing*, IEEE Signal Processing Letters,vol.3, no.3, March 1996, pp.63–65

[33] , S. Marsi and S. Carrato and G. Ramponi, *VLSI Implementation of a Nonlinear Image Interpolation Filter*, IEEE Trans. on Consumer Electronics, vol.42, no. 3, August 1996, pp.721–728

[34] , R. Castagno and G. Ramponi, *A Rational Filter for the Removal of Blocking Artifacts in Image Sequences Coded at Low Bitrate*, Proc. 8th European Signal Processing Conf., September 1996

[35] R. Castagno, S. Marsi, G. Ramponi, *A SImple Algorithm for the Reduction of Blocking Artifacts in Images and its Implementation*, IEEE Trans. on Consumer Electronics, Aug. 1998, vol.44, no.3.

[36] S. Carrato, G. Ramponi, S. Marsi, *A Simple Edge-Sensitive Image Interpolation Filter* Proc. Third IEEE Int. Conf. on Image Processing, ICIP-96, Lausanne, CH, Sept. 1996

[37] G. Ramponi, *Image Processing Using Rational Functions*, Proc. COST-254 Workshop, February 1997

[38] , G. Ramponi and C. Moloney, *Smoothing Speckled Images Using an Adaptive Rational Operator*, IEEESPL, vol. 4, no. 3, March 1997

[39] F. Cocchia and S. Carrato and G. Ramponi, *Design and Real-Time Implementation of a 3-D Rational Filter for Edge Preverving Smoothing*, IEEE Transactions on ComputersE, vol. 43, no. 4, November 1997, pp.1291–1300

[40] G. Ramponi and S. Carrato, *Interpolation of the DC Component of Coded Images Using a Rational Filter*, Proc. 4th Int. Conf. on Image Processing, October 1997

[41] G. Ramponi and A. Polesel, *A Rational Unsharp Masking Technique*, Journal of Electronic Imaging, vol.7, no.2, April 1998, pp. 333–338

[42] G. Ramponi and P. Fontanot, *Enhancing document images with quadratic filters*, Signal Processing, vol.33, no.1, July 1993, pp. 23–34

[43] G. Bernacchia, S. Marsi, *A VLSI Implementation of a Configurable Rational Filter*", Proc. of ICCE '97, Los Angeles, California, June 1998.

[44] G. Bernacchia, S. Marsi, *A VLSI Implementation of a Configurable Rational Filter*, IEEE Trans. on Consumer Electronics, Vol.44, No.3, pp. 1076-1085, Aug. 1998. (extended paper)

[45] L. Krhiji, G. Bernacchia, M. Gabbouj, G. Sicuranza, *Hardware Implementation of the Median-Rational Hybrid Filters for Colour Images*,IEEE-EURASIP Workshop on Nonlinear SIgnal and Image Processing, Antalya, Turkey, June 1999.

[46] L. Krhiji, M. Gabbouj, *Median-Rational Hybrid Filters for Image Restoration*, IEEE Signal Processing Letters, June 1999 (to be published)

[47] L. Krhiji, M. Gabbouj, *Vector Median-Rational Hybrid Filters*, Electronic Letters, vol.34, no.10, May 1998, pp.977-979.

[48] L. Khriji, G. Bernacchia, M. Gabbouj, G. Sicuranza*Hardware Implementation of the Median-Rational Hybrid Filters*, Int. Conf. on Electronics, Circuits and Systems, Paphos, Greece, Sept. 1999.

[49] G. Bernacchia, L. Krhiji, M. Gabbouj, G. Sicuranza, *A Dedicated Hardware System for the Median-Rational Hybrid Filters*,Proc. Sixth IEEE Int. Conf. on Image Processing, ICIP-99, Oct. 25-28, 1999, Kobe, Japan.

[50] G. Bernacchia, L. Krhiji, M. Gabbouj, G. Sicuranza, *Programmable Hardware Implementation for the Median-Rational Hybrid Filters*,Proc. Sixth IEEE Int. Conf. on Image Processing, ICIP-99, Oct. 25-28, 1999, Kobe, Japan.

[51] G. Bernacchia, M. C. Papefhtymiou, *Analytical Macromodeling for High-Level Power Estimation*, International Conference on Computer Aided Design, San José, CA, Nov. 1999

[52] E. Mumolo and A. Carini, *Recursive Volterra Filters with Stability Monitoring*, Proc. 8th European Signal Processing Conf., Sept. 1996

[53] T.S. Panicker and V.J. Mathews and G.L. Sicuranza, *Parallel-cascade adaptive Volterra filters*, Proc. 8th European Signal Processing Conf., Sept. 1996

[54] G. Ramponi and P. Fontanot, *Enhancing document images with quadratic filters*, Signal Processing, vol. 33, no. 1, July 1993, pp.23–34

[55] H.T. Kung, *Why Systolic Architectures ?*, CMP, vol. 3, no.3, Jan. 1992, pp. 637–44

[56] P. Kornerup, *A Systolic Linear Array Multiplier for a Class of Right-Shift Algorithms*, IEEE Transactions on Computers, vol. 43, no. 8, Aug. 1994, pp.892–898

[57] F. Zhou and P. Kornerup, *Bit Serial Structure for Full Search, Block Matching Algorithm*, Proc. of the IS&T, Symposium on Electronic Imaging, Jan. 1996

[58] S.M. Nowick and M.B. Josephs and C.H. van Berkel, *Special Issue on Asynchronous Circuits and Systems*, IEEE Proceedings, vol. 87, no.2, Feb. 1999

[59] A.J. Martin, *Asynchronous Datapaths and the Design of an Asynchronous Adder*

[60] G.E. Sobelman and K. Frant, *CMOS Circuit Design of Threshold Gates with Hysteresis*

[61] M. Renaudin and B. El Hassan and A. Guyot, *A New Asynchronous Pipeline Scheme: Application to the Design of a Self-Timed Ring Divider*, IEEESSC, vol.31, no.7, July 1996, pp.1001–1013

[62] C.H. Chang, *Performance Optimization of Pipeline Circuits with Latches and Wave Pipelining*, Ph.D. dissertation at the University of Michigan, 1996

[63] S.M. Nowick and K.Y. Yun and P.A. Beerel and A.E. Dooply, *Speculative Completion for the Design of High-Performance Asynchronous Dynamic Adders*, Proc. of the 3rd IEEE Int. Symp. on Advanced Research in Asynchronous Circuits and Systems, Async-97, Eindhoven, Netherlands, 1997

[64] K.Y. Yun and P.A. Beerel and J. Arceo, *High-Performance Asynchronous Pipeline Circuits*

[65] J.D. Garside, *A CMOS VLSI Implementation of an Asynchronous ALU*, Proc. of the VII Banff Workshop:Asynchronous Hardware Design Canada, Aug. 1993

[66] N.C. Paver and P. Day and S.B. Furber and J.D. Garside and J.V. Woods, *Register Locking in an Asynchronous Microprocessor*, Proc. of the 1992 Int. Conf. on Computer Design: VLSI in COmputer & Processors Oct. 1992, pp.351–355

[67] Mi Lu, Jen-Shinu Chiang, "A Novel Division Algorithm for the Residue Number System", IEEE Transactions on Computers, vol.41, no.8, Aug 92.

[68] U.V. Cummings and A.M. Lines and A.J. Martin, *An Asynchronous Pipelined Lattice Sructure Filter*, Proc. Int. SYmp. on

Advanced Research in Asynchronous Circuits and Systems, Nov. 1994, pp. 126–133

[69] W.F. Richardson, *Architectural Considerations in a Self-Timed Processor Design*, Ph.D. Thesis at the University of Utah, March 1996

[70] P.C. Mathias and L.M. Patniak, *Systolic Evaluation of Polynomial Expressions* IEEE Transactions on Computers, vol.39, no.5, May 1990, pp. 653–665

[71] I. Sutherland, *Micropipelines*, Communications of the ACM, vol.32, no.6, 1989, pp.720-738,

[72] A. Avizienis, *Signed digit number representation for a fast parallel arithmetic*, IRE Trans. Electron. Comput., EC-10, Sept. 1961, pp.389-400,

[73] A. Vandemeulebroecke and E. Vanzieleghem and T. Denayer and P.G.A.Jespers, *A New Carry-Free Division Algorithm and its Application to a Single-Chip 1024-bit RSA Processor*, IEEE Journal of Solid-State Circuits, vol.25, no.3, June 1990

[74] K. Hwang, *Computer Arithmetic. Principles, Architecture and Design*, John Wiley & Sons, New York, 1979

[75] A.M. Nielsen, *Number Systems and Digit Serial Arithmetic*, Ph.D. Thesis at the Dept. of Mathematics and COmputer Science, Odense University, Denmark, Aug. 1997

[76] A.M. Nielsen and P. Kornerup, *MSB-First Digit Serial Arithmetic*, Journal of Universal Computer Science, vol.1, no.7, 1995

[77] P. Kornerup and D.W. Matula, *An Algorithm for Redundant Binary Bit-Pipelined Rational Arithmetic*, IEEE Transactions on Computers, vol.39, no.8, Aug. 1990, pp.1106–1115

[78] E.B. Saff and R.S. Vargas, *Padè and Rational Approximation. Theory and Applications*, Academic Press, New York,1977

[79] A. Kurosh, *Higher Algebra*, Mir Publishers, Moscow

[80] D. Manocha, *MultiPolynomial Resultant Algorithms*, JSC, vol.15, 1993, pp.99–122

[81] G. Viry, *Factorization of Multivariate Polynomials with Coefficients in F_p*, JSC, vol.15, 1993, pp.371–391

[82] C.G. Ponder, *Parallel Multiplication and Powering of Polynomials*, JSC, vol.11, 1991, pp.307–320

[83] E.R. Hansen and M.L. Patrick and R.L.C. Wang, *Polynomial Evaluation with Scaling*, ACM, vol.16, no.1, March 1990, pp.86–93

[84] L. Kronsjo, *Algorithms: Their Complexity and Efficiency*, John WIley and Sons, Chichester, 1979

[85] V. Kantabura, *Designing Optimum Carry-Skip Adders*, Proc. of the 10th Int. SYmp. on COmputer Arithmetic, Grenoble, France, 1991

[86] S. Turrini, *Optimal Group Distribution in Carry-Skip Adders*, Western Research Laboratory Research Report no. 89/2, 1989

[87] R.P. Brent and H.T. Kung, *A Regular Layout for Parallel Adders*, IEEE Transactions on Computers, vol.31, no.3, March 1982, pp.260–264

[88] S.E. McQuillan and J.V. McCanny, *VLSI Module for High-Performance Multiply, Square Root and Divide*, IEE Proceedings, vol.139, no.6, Nov. 1994, pp.505–510

[89] M. Kishinevsky and L. Lavagno and P. Vanbekbergen, *The Systematic Design of Asynchronous Circuits*, Tutorial presented at the Int. Conf. on Computer Aided Design, ICCAD'95, 1995

[90] F. Bergadano and N.H. Bshouty and S. Varricchio, *Learning Multivariate Polynomials from Substitution and Equivalence Queries*, ECCC Technical Report TR96-008, Jan. 1996

[91] I. Shparlinski, *On Polynomial Representation of Boolean Function Related to Some Number Theoretic Problems*, ECCC Technical Report TR98-054, 1998

[92] A. Chistov and M. Karpinski, *Algorithms for Interpolation of Sparse Rational Functions Using Polynomial Number of Evaluations*

[93] A. Chistov and G. Ivanyos and M. Karpinski, *Polynomial Time Algorithms for Modules over Finite Dimensional Algebras*, May 1997

[94] K. Werther, *An Interpolation Algorithm for Sparse Polynomials over Z_M*, Aug. 1994

[95] V. Shoup, *A New Polynomial Factorization Algorithm and its Implementation*, JSC, vol.20, 1995, pp.363–397

[96] V. Pan, *Simple Multivariate Polynomial Multiplication*, JSC, vol.18, 1994, pp.183–186

[97] J. Ganz, *Evaluation of Polynomials Using the Structure of the Coefficients*, SIAM Journal of Computation, vol.24, no.3, June 1995, pp.473–483

[98] L.A. Montalvo and K.K. Parhi and A. Guyot, *New Svoboda-Tung Division*

[99] N. Mikami and M. Kobayashi and Y. Yokoyama, *Roundoff-Error Reduction for Evaluation of a Function by Polynomial Approximation with Error Feedback in Fixed-Point Arithmetic*, IEEESP, vol.41, no.5, May 1995, pp.1953–1955

[100] Gang Li and M. Gevers, *Roundoff Noise Minimization Using Delta-Operator Realizations*, IEEESP, vol.41, no.2, Febr. 1993, pp.629–637

[101] L.W. Chang, *Roundoff Error Problem of the Systolic Array for DFT*, IEEESP, vol.41, no.1, Jan. 1993, pp.395–398

[102] T. Bose, *Asymptotic Stability of Two-Dimensional Digital Filters Under Quantization*, IEEESP, vol.42, no.5, May 1994, pp.1172–1177

[103] C.Y. Hsu J.C. Yao, *Comparative Performance of Fast Cosine Transform with Fixed-Point Roundoff Error Analysis*, IEEESP, vol.42, no.5, May 1994, pp.1256–1259

[104] K. Chang and W.G. Bliss, *Finite Word-Length Effects of Pipelined Recursive Digital Filters*, IEEESP, vol.42, no.8, Aug. 1994, pp.1983–1995

[105] G.L. Litvinov, *Approximate Construction of Rational Approximations and the Effect of Error Autocorrection. Applications*, Technical Report, University of Oslo, May 1993

[106] Saeyang Yang, *Logic Synthesis and Optimization Benchmarks User Guide. Version 3.0*, MCNC Technical Report, Jan. 1991

[107] A.C. de Graaf and A.J. van Genderen, *SLS: Switch Level Simulator User's Manual*, Delft University of Technology, Oct. 1987

[108] C. Gloster, Jr., *Dynamic Scan Testing: Investigating A New Paradigm*, MCNC Technical Report TR-93-06 April 1993

[109] W.H. Press and S.A. Teuklosky and W.T. Vetterling and B.P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, Jan. 1993

[110] H. Lappalainen, *Using an MDL-Based Cost Function with Neural Networks*, Proc. of the Int. Conf. On Neural Networks IJCNN'98, May 1998, pp.2384–2389

[111] L. Cao and A. Mees and K. Judd and G. Froyland, *Identification of Variable Relations from Multivariate Chaotic Time Series Data*, Proc. of the 1997 Int. Symp. on Nonlinear Theory and its Applications, vol.2, Dec. 1997, pp.1305–1308

[112] L. Cao and A. Mees and K. Judd and G. Froyland, *Determining the Embedding Dimensions of Input-Output Time-Series Data*, Int. J. of Bifur. Chaos Appl. Sci. Engrg., 1999, in press

[113] J. Rissanen, *Modeling by Shortest Data Description*, Automatica, vol.14, 1978, pp.465–4471

[114] J. Rissanen, *Stochastic Complexity in Statistical Inquiry*, World Scientific, Series in Computer Science Vol. 15

[115] J. Monteiro and S. Devadas, *Techniques for the Power Estimation of Sequential Logic Circuits under User-Specified Input Sequences and Programs*, Symposium on Low Power Design, 1995, pp.33–38

[116] M. Favalli and L. Benini, *Analysis of Glitch Power Dissipation in CMOS ICs*, Symposium on Low Power Design, 1995, pp.123–128

[117] S. Turgis and N. Azemard and D. Auvergne, *Explicit Evaluation of Short Circuit Power Dissipation for CMOS Logic Structures*, Symposium on Low Power Design, 1995, pp.129–134

[118] S. Gupta and F. N. Najm, *Analytical Model for High Level Power Modeling of Combinational and Sequential Circuits*

[119] M. Barocci and L. Benini and A. Bogliolo and B. Riccò and G. De Micheli, *Lookup Table Power Macro-models for Behavioral Library Components*

[120] A.M. Hill and S.M. Kang, *Determining Accuracy Bounds for Simulation-Based Switching Activity Estimation*, Symposium on Low Power Design, Oct. 1995, pp.215–220

[121] C.S. Ding and C.T. Hsieh and M. Pedram, *Improving Sampling Efficiency for System Level Power Estimation*, Proc. Int. Symp. on Low Power Electronics and Design, 1998, pp.115–117

[122] S.B.K. Vrudhula, H.-Y. Xie, *Techniques for Cmos Power Estimation and Logic Synthesis for Low Power, Proc. Int. Workshop Low Power Design, Apr. 1994.*

[123] N. Dragone and R. Zafalon and C. Guardiani and C. Silvano, Power Invariant Vector Compaction Based on Bit Clustering and Temporal Partitioning, *Proc. Int. Symp. on Low Power Electronics and Design, 1998, pp.118–120*

[124] , A. Macii and E. Macii and M. Poncino and R. Scarsi, Stream Synthesis for Efficient Power Simulation Based on Spectral Transform, *Proc. Int. Symp. on Low Power Electronics and Design, 1998, pp.30–35*

[125] D. A. Kirkpatrick and A. L. Sangiovanni-Vincentelli, Digital Sensitivity: Predicting Signal Interaction Using Functional Analysis, *Proc. Int. Symp. on Low Power Electronics and Design, 1998, pp.536–541*

[126] A. Narayan and J. Jain and M. Fujita and A. Sangiovanni-Vincentelli, Partitioned ROBDDS - A Compact, Canonical and Efficiently Manipulable Representation for Boolean Functions, *Proc. Int. Symp. on Low Power Electronics and Design, 1998, pp.547–553*

[127] C.T. Hsieh and Q. Wu and C.S. Ding and M. Pedram, Statistical Sampling and Regression Analysis for RT-Level Power Evaluation, *Proc. Int. Symp. on Low Power Electronics and Design, 1998, pp.583–588*

[128] H. Yalcin and J.P. Hayes and K.A. Sakallah, An Approximate Timing Analysis Method for Datapath Circuits, *Proc. IEEE-ACM Design Automation Conference, June 1999, pp.114–118*

[129] A. Raghunathan and S. Dey and N.K. Jha, Register-Transfer Level Estimation Techniques for Switching Activity and Power Consumption, *Proc. IEEE-ACM Int. Conf. on Computer-Aided Design, Nov. 1996, pp. 158–165*

[130] A. Raghunathan and S. Dey and N.K. Jha, Glitch Analysis and Reduction in Register Transfer-Level Power Optimization, *Proc. 33rd ACM/IEEE Design Automation Conference, June, 1996, pp. 331–336*

[131] Y.J. Lim and K.I. Son and H.J. Park and M. Soma, A Statistical Approach to the Estimation of Delay-Dependent Switching Activities in CMOS Combinational Circuits, *Proc. 33rd ACM/IEEE Design Automation Conference, June 1996, pp.445–450*

[132] M.G. Xakellis and F.N. Najm, Statistical Estimation of the Switching Activity in Digital Circuits, *Proc. 31rd ACM/IEEE Design Automation Conference, June, 1994, pp. 728–733*

[133] S. Gupta and F. N. Najm, Power Macromodeling for High Level Power Estimation, *Proc. Design Automation Conference, June 1997, pp. 365–370*

[134] A. Bogliolo and B. Riccò and L. Benini and G. De Micheli, Accurate Logic-Level Power Estimation, *Symposium on Low Power Design, Oct. 1995, pp. 40–43*

[135] A. Bogliolo and L. Benini, Node Sampling: A Robust RTL Power Modeling Approach, *Proc. IEEE-ACM Int. Conf. on Computer-Aided Design, Nov. 1998, pp. 461–467*

[136] T. Sato and Y. Ootaguro and M. Nagamatsu and H. Tago, Evaluation Architecture-Level Power Estimation for CMOS RISC Processors, *Symposium on Low Power Design, Oct. 1995, pp. 44–45*

[137] S. Devadas and B. Lin and C.Y. Tsui and M. Pedram, Exact and Approximate Methods of Switching Activity Estimation in Sequential Logic Circuits, *Proc. Int. Workshop on Low Power Design, 1994, pp. 117–122*

[138] P.H. Schneider and U. Schlichtmann, Decomposition of Boolena Functions for Low Power Based on a New Power Estimation Technique, *Proc. Int. Workshop on Low Power Design, 1994, pp. 123–128*

[139] E.D. Kyriakis-Bitzaros and S. Nikolaidis and A. Tatsaki, Accurate Calculation of Bit-Level Transition Activity Using Word-Level Statistics and Entropy Function, *Proc. Int. Conf. on Computer-Aided Design, Nov. 1998, pp. 607–610*

[140] B. Kapoor, Improving the Accuracy of Circuit Activity Measurement, *Proc. Int. Workshop on Low Power Design, 1994, pp. 111–1116*

[141] D. Marculescu and R. Marculescu and M. Pedram, Information Theoretic Measures of Energy Consumption at Register Transfer Level, *Symposium on Low Power Design, Oct. 1995, pp.81–86*

[142] F. N. Najm, Towards a High-Level Estimation Capability, *Symposium on Low Power Design, Oct. 1995, pp. 87–92*

[143] I. Hamzaoglu and J.H. Patel, Test Set Compaction Algorithms for Combinational Circuits, *Proc. IEEE-ACM Int. Conf. on Computer-Aided Design, Nov. 1998, pp. 283–289*

[144] A.T. Freitas and A.L. Oliveira and J.C. Monteiro and H.C. Neto, Exact Power Estimation Using Word Level Transition Probabilities, *PATMOS99, Oct. 1999*

[145] S. Theoharis and G. Theodoridis and N. Zervas and C. Goutis, Accurate and Fast Power Estimation of Large Combinational Circuits, *PATMOS99, Oct. 1999*

[146] R. Ferreira and A.M. Trullemans, An Efficient Switching Activity Estimation for Low Power Resynthesis under General Delay, *PATMOS99, Oct. 1999*

[147] A. Bogliolo and E. Macii and V. Mihalovici and M. Poncino, Combinational Characterization-Based Power Macro-Models for Sequential Macros, *PATMOS99, Oct. 1999*

[148] H. Eriksson and P. Larsson-Edefors and A. Edman, Using Clock Division to Reduce Power Consumption, *PATMOS99, Oct. 1999*

[149] Y. Tsividis and P. Antognetti, Design of MOS VLSI Circuits for Telecommunications, *Prentice Hall, Inc.*

[150] R.X. Gu and K.M. Sharaf and M.I. Elmasry, High-Performance Digital VLSI Circuit Design, *Kluwer Academic Publishers, Boston, 1996*

[151] *R. Zimmermann and H. Kaeslin,* Cell-Based Multilevel Carry-Increment Adders with Minimal AT- and PT-Products, *IEEE Trans. on Very Large Scale Integration (VLSI) Systems, submitted July 1995*

[152] *H. Kunz and R. Zimmermann* , High-Performance Adder Circuit Generators on Parameterized Structural VHDL, *ETH Technical Report 96/7, Aug. 1996*

[153] *R. Payne,* Self-Timed FPGA Systems, *Proc. 5th Int. Workshop on Field Programmable Logic and Applications, Sept. 1995*

[154] *B. Lang,* Self Arbitrating Elements for Modelling Systolic Dataflow in Field Programmable Gate Arrays, *Proc. Workshop Anwenderprogrammierbare Schaltungen, June 1994*

[155] The Role of Distributed Arithmetic in FPGA-based Signal Processing, *Xilinx Application Note, ref: bhttp://www.xilinx.com/apps/arith.html*

List of Tables

List of Figures