Project and Development of Hardware Accelerators for Fast Computing in Multimedia Processing

Cappetta Carmine

UNIVERSITY OF SALERNO

DEPARTMENT OF INDUSTRIAL ENGINEERING

Ph.D. Course in Industrial Engineering
Curriculum in Electronic Engineering - XXXI Cycle

PROJECT AND DEVELOPMENT OF HARDWARE ACCELERATORS FOR FAST COMPUTING IN MULTIMEDIA PROCESSING

Supervisor
Prof. Gian Domenico Licciardo

Ph.D. student
Carmine Cappetta

Scientific Referees
Prof. Nicola Petra
Prof. Maurizio Martina

Ph.D. Course Coordinator
Prof. Ernesto Reverchon

Ma rimasi nascosto,
Passarono a stanarmi,
Tornai a muovermi
Rimanendo al buio.
Mi insegnarono a vedere,
A riconoscere la tecnica,
Poi anche l'arte.
Mi mostrarono la cima,
Capovolsero la prospettiva,
Ero ancora acerbo.
Ora punto solo a migliorare
Provando a preferire
Che la ragione sia altrui.

Acknowledgments

The development of this work and the personal growth that come with it has been possible thank to many people.

First, I would like to thank Prof. Gian Domenico Licciardo for the help and the chances he gave to me, and, moreover, his suggestions and encouragement. A special thank goes also to Luigi, who not only helped me, but also put up with me, which is no easy task. With him I cannot forget to thank also Prof. Alfredo Rubino, Rosalba and Antonio for the kindness and their time. I would also like to thank Prof. Salvatore Bellone, because without his teaching I would not be here and I would not have chosen a PhD course; moreover, I would have been a completely different engineer and person.

Huge thanks goes also to Giuseppe and Thomas for the period at STMicroelectronics. Together with their group, they have not only accepted me, but made me feel comfortable and taught me how to move in a different environment from the university one. Last year changed a lot my perspectives and my way of thinking.

I would not be the man I am without my family, hence a particular thank goes to my mother, Helga, who is often too caring towards me even if I could be detached or annoying, my father, Donato, who somehow manages to find always the right words to take out the best out of me, and my sister, Gina, who helped me in so many ways she could not imagine. With them there is also a great supporting cast made up by my grandparents, my uncles and cousins.

A lot of people say that friends are the family you choose, so I have to thank also: Aika, Alessia, Alessio, Alfonso, Carmen, Dario, Ece, Emanuele, Emilio, Flora, Francesco, Giovanni, Luca, Maria Alfonsa, Marianna, Mattia, Mehmet, Michele, Monica, Nicola, Raffaele, Rino, Roberto, Rosa, Sarah, Serena, Tommaso, Ugo and Valerio that always supported me during these months. I wish to particularly thank five of my closest friends among the others. Carmine, who is my friend since we were 3 years old helped me in so many ways during these years, together with his family, and with so much patience that he is like a brother to me (Cosimo, Sara, Mattia and Alessio I will not forget you and I hope nothing but the best for you). Carlo, who despite his occupations never forgets to give me a call or to ask me how I feel, even if he notices a slight change of my mood (with him comes an army of other friends: Elvira, Giuseppe, Giulio, Orlando and Maria Virginia); Andrea, who is not my psychologist, but really helps in easing my mind, which is one of the most amazing stuff a person could do for a guy who thinks too much even on small things like me; Manuela and her family (Filomena and Giovanni) who helped me throughout all my Milan days and not only, making me feel accepted and giving me some great memories; Gianmarco who is not only able to calm me down, but is also a great person to talk to.

If anyone of you is reading that right now I would you to know that I will be very happy to share with you a cup of coffee and a chat at any time, thus let us keep in touch so that I could thank you more in the future: you will not be forgotten.

List of publications

Licciardo, G.D., Cappetta, C., Di Benedetto, L., Rubino, A., "Design of an offset-tolerant voltage sense amplifier bit-line sensing circuit for SRAM memories", *Electronics Letters*, 2016.

Licciardo, G.D., Cappetta, C., Di Benedetto, L., Vigliar, M., "Weighted Partitioning for Fast Multiplierless Multiple-Constant Convolution Circuit", *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2017.

Licciardo, G.D., Cappetta, C., Di Benedetto, L., "FPGA optimization of convolution-based 2D filtering processor for image processing", *2016 8th Computer Science and Electronic Engineering Conference, CEEC 2016 - Conference Proceedings*, 2017.

Licciardo, G.D., Cappetta, C., Di Benedetto, L., "Design and FPGA implementation of a real-time processor for the HDR conversion of images and videos", *2016 8th Computer Science and Electronic Engineering Conference, CEEC 2016 - Conference Proceedings*, 2017.

Licciardo, G.D., Cappetta, C., Di Benedetto, L., "Application specific image processor for the extension of the dynamic range of images with multiple resolutions", *2016 IEEE International Conference on Electronics, Circuits and Systems, ICECS 2016*, 2017.

Licciardo, G.D., Cappetta, C., Di Benedetto, L., "Design of a Convolutional Two-Dimensional Filter in FPGA for Image Processing Applications", *Computers*, 2017.

Licciardo, G.D., Cappetta, C., Di Benedetto, L., "Dynamic range enhancement for medical image processing", *Proceedings - 2017 7th International Workshop on Advances in Sensors and Interfaces, IWASI 2017*, 2017.

Cappetta, C., Licciardo, G.D., Di Benedetto, L., "Optimal design of a Gabor filter for medical imaging applications", *Proceedings - 2017 7th International Workshop on Advances in Sensors and Interfaces, IWASI 2017*, 2017.

Cappetta, C., Licciardo, G.D., Di Benedetto, L., "An FPGA oprimization of a multiple resolution architecture for LDR to HDR image conversion", *ISSCS 2017 - International Symposium on Signals, Circuits and Systems*, 2017.

Cappetta, C., Licciardo, G.D., Di Benedetto, L., "Hardware accelerator using Gabor filters for image recognition applications", *ISSCS 2017 - International Symposium on Signals, Circuits and Systems*, 2017.

Cappetta, C., Licciardo, G.D., Di Benedetto, L., "Hardware architecture for 2D Gaussian filtering of HD images on resource constrained platforms", *ISSCS 2017 - International Symposium on Signals, Circuits and Systems*, 2017.

Licciardo, G.D., Cappetta, C., Di Benedetto, L., Rubino, A., Liguori, R., “Multiplier-Less Stream Processor for 2D Filtering in Visual Search Applications”, *IEEE Transactions on Circuits and Systems for Video Technology*, 2018.

Licciardo, G.D., Cappetta, C., Di Benedetto, L., “Design of a Gabor Filter HW Accelerator for Applications in Medical Imaging”, *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 2018.

Licciardo, G.D., Cappetta, C., Di Benedetto, L., “Design Criteria for Real-time Processing of HW Gabor Filters in Visual Search”, *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2018.

Summary

Contents

Figures Index

Contents

Tables Index

Contents

Abstract

The aim of the present thesis work is to show the developed researches and the related results on the project and development of Very Large Scale Integration (VLSI) electronic circuits and in particular on Hardware (HW) accelerators and dedicated arithmetic units which could be used in resource and time constrained applications, with particular attention to image and video contents ones. The range of applications in this field stems from image filtering and enhancement to object recognition and Neural Networks (NN), to list only some of the most demanding ones. The aim of the present work is to create a set of HW accelerators tools capable of being integrated in a single Digital Signal Processor (DSP) to carry out the needed pre-filtering operations on the considered input data.

The present work will develop the HW simplifications needed to implement the well-known algorithms in HW friendly fashion, in order to obtain a resource optimization which results in better trade-offs between area and power consumption, maximum operating frequency, resource utilization and portability.

The following thesis will be organized as follows:

- the importance of digital image processing and its developments are explained in the introduction with particular attention to HW applications;
- a new floating point (FP) multiplier for dedicated applications have is shown, capable of simplifying the HW complexity of FP Multiply and Accumulate (MAC) used in several filter applications;
- a HW accelerator structure for Low Dynamic Range (LDR) images enhancement capable of working with several image resolutions up to 4K UHDTV format is presented;
- a design of an offset tolerant sensing circuit for Static Random Access Memory (SRAM) circuits is developed to address the integration related problems for these kind of circuits;
- a HW accelerator using Gabor filters is presented to address pre-filtering issues in image retrieval problems, while reducing the computational burden of the overall operation via careful simplifications and approximations;
- the previously developed instruments and acquired knowledge are used to develop part of a NN, in order to obtain a speed up during pre-filtering operations, which represents the real bottleneck of the overall architecture.

The developed activities took place in Lab I7 of Microelectronics of the Department of Industrial Engineering in Università degli Studi di Salerno and, for the last part, in STMicroelectronics facility of Castelletto, Cornaredo (MI). The design, simulation, measurement and setup were personally performed and followed.

This work is a part of a project developed in cooperation with STMicroelectronics.

Introduction

In the last decades more and more applications arose in the field of digital image processing, due to the development and the diffusion of new data processing algorithms, handheld devices and automatic systems. These developments caused a huge growth in the amount of data related to image and video content to be processed, which in turn resulted in many cases in several constraints in the developed applications, in particular, due to the increase in the resolution formats and in the information carried in the image (e.g. the extension of the dynamic range). This obviously represents a problem for hard real-time applications in which the system has to produce the result in a well defined and reduced amount of time, in order to satisfy the requirements needed by the other parts of the system to carry on the computation. This chapter sketches a brief recap of the main results related to image digital processing.

i.1 Background

From a mathematical standpoint an image could be seen as a function of two variables [Gonzalez C., Woods R. E., 2007] *f(x,y)*, being *x* and *y* the two spatial coordinates describing the image space and determining an output which could be a single value representing for example the gray level of the image in that point, or a vector, taking into account the different components composing a model for the definition of the image (e.g. the three emulsions sensible for red, green and blue spectral components for analog images and the RGB [Red Green Blue] color model for digital images). In the case of analog images we would have a continuous value, while in the case of digital images the result will be a discrete and limited set of values. Moreover, any digital image has to be obviously composed by a finite number of elements, while the reproduction of an analog image is related only to the light wavelength, the photographic film and it, of course, a continuous function. The elements composing the digital image are called picture elements or *pixels*.

One of the most important features of digital image processing is that, changing only the dedicated sensors, it is possible to extend the processing algorithms to images obtained using not only sensors working in the visible band of the electromagnetic (EM) spectrum, but in every bandwidth, and, moreover, not only to the light spectrum, but also to acoustic waves. An example using multi wavelength imaging is reported in Figure i.1.

Table i.1 *Class of used wavelengths in Digital Image Processing, frequencies and applications.*

Class	Frequency [*Hz*]	Applications
Gamma Ray	$> 5 \cdot 10^{19}$	Medical Diagnostics, Astronomy
X-Ray	$3.4 \cdot 10^{16}$ - $5 \cdot 10^{19}$	Medical Diagnostics, Industrial Diagnostics
Ultra Violet	$7.9 \cdot 10^{14}$ - $3.4 \cdot 10^{16}$	Lithography, Biological Imaging, Astronomy
Visible	$3.9 \cdot 10^{14}$ - $7.9 \cdot 10^{14}$	Used in all other reported applications
Infrared	$3 \cdot 10^{11}$ – $3.9 \cdot 10^{14}$	Satellite Imaging
Microwave	10^{9} - $3 \cdot 10^{11}$	Astronomy, Geographic Area Definition
Radio	$< 10^{9}$	Medical Diagnostics, Astronomy
Ultrasound	$2 \cdot 10^{4}$ - $2 \cdot 10^{8}$	Medical Diagnostics, Artefacts Diagnostics

Applications about digital image and video content date back to 1960s when the contemporary developments of computers and high-level programming languages made it possible the first processing on digital images. However, it was only thanks to the miniaturization of the components due to the introduction of VLSI techniques and Hardware Description Languages (HDLs) in the 1980s that these applications became of huge commercial interest, making this one of the most important research fields. Another turning point in image processing diffusion was represented by the introduction of handheld devices, such as digital cameras, smartphones,

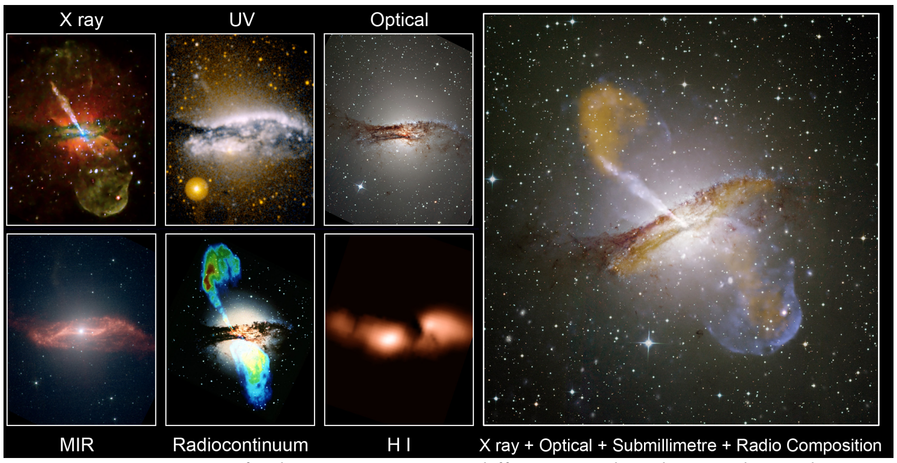


Figure i.1 *Image of galaxy Centaurus A at different wavelengths, together with a composition of the same images [http://www.atnf.csiro.au/people/lop009/multiwave.html].*

which together with Internet-of-Things (IoT) applications highlighted the need for high frequency, low power and reduced area applications. The target performances could be achieved via different trade-offs and approximations, feasible using software (SW) implementations, HW ones or both. In particular, low level tasks, as contrast enhancement, simple changes in the image settings and simple computations, could be carried out in real-time by a SW system, having the benefit of being more easy to program, more flexible and being a general purpose system; obviously the drawback in these systems are related to the fact that they are not optimized for any task, which is represents particularly a problem in the case of specific high level tasks, like in features extraction, image recognition and in general all the tasks having higher computational complexity. For these reasons, nowadays several demanding applications demand for at least a HW accelerator stage before carrying the rest of the computations in SW. In fact, the different specific tasks require different approaches, in particular for high level tasks.

i.2 Sampling, Quantization and Resolution

Image sensors usually output continuous voltages, which need to be converted in order to be fed to digital screens, digital image processors and other devices working on discrete signals. The conversion is done obtaining discrete values for both spatial coordinates and amplitude of the input image; the former task is called *sampling*, while the latter is called *quantization*. The sampling in space is mostly determined by the sensor which acquires the image, since it is inherent to mechanical limits, lenses manufacturing, dark currents and similar issues. The maximum sampling rate is established by

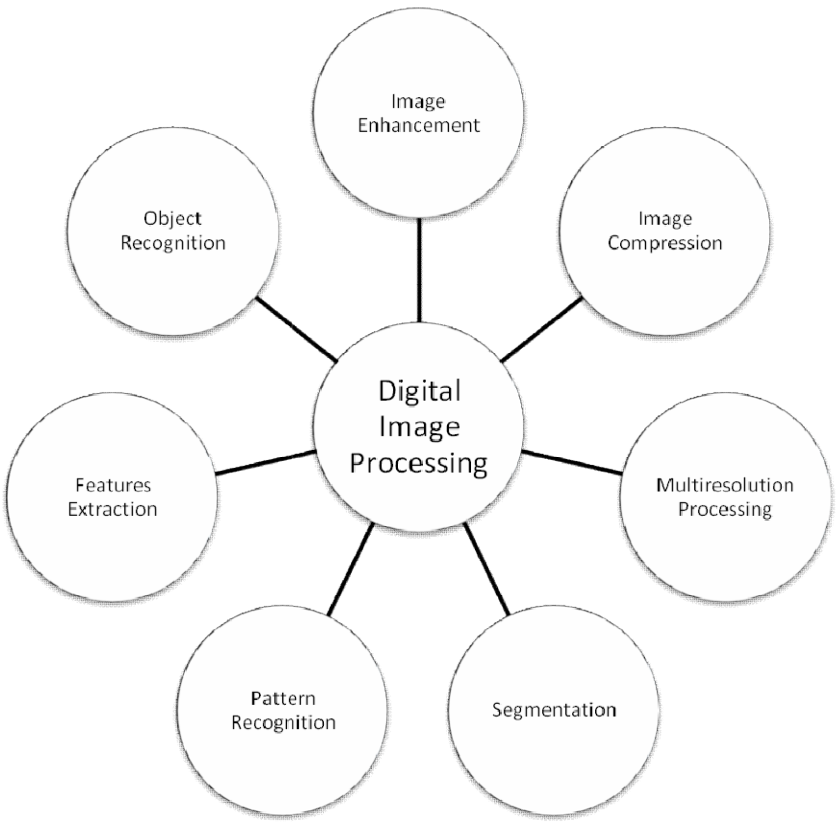


Figure i.2 *Principal applications in digital image processing.*

the density of sensing elements in the considered image sensor. Quantization is instead obtained by digitalizing the amplitude of the incoming signals; it can be obtained digitalizing the image linearly, as in usual Low Dynamic Range (LDR) images, or not linearly, as it happens in High Dynamic Range (HDR) photography. The final result of these processes is a matrix of numeric values representing the image, in which each value is called pixel. The number of matrices to represent a single image depends on the image coding: in the case of gray scale images it is possible to use a single matrix to represent the image, while if dealing with RGB images three matrices for the three channels are needed, for example. Figure i.3 shows the results obtained quantizing Lena image with a different number of quantization levels.

Another important concept in digital image processing is *resolution* which gives the minimum quantity of detail the image can carry, e.g. how close two lines in a picture could be being visibly resolved. The higher the resolution, the better it is possible to distinguish the details in the image.

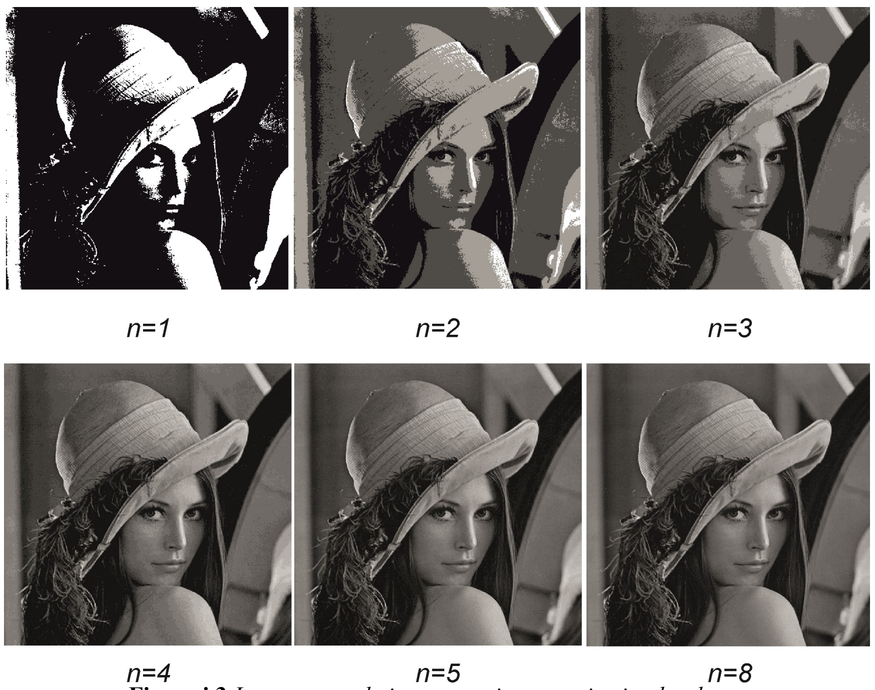


Figure i.3 *Lena grayscale image varying quantization levels, n.*

Resolution of the obtained images tends to increase when using better image sensors. Standard formats from a minimum of 352x240 pixels (Video CD) to a maximum of 10*k* x 7*k* pixels (IMAX HD) are used, depending on the task of interest.

i.3 Image Enhancement

Image enahnacement algorithms change a lot according to the particular application they have to be fit. For this reason, the different problems have to be dealt using different solutions, also from an HW standpoint. The chosen methods could change depending on the wavelength at which the image was taken, the particular field of application (medical, photographic, astronomy, etc.), the parts of the images we are interested to enhance (colors, edges, particular patterns) and more. Due to the number of applications to consider, general purpose processing is left to SW implementations, while HW ones focus on particular issues and are hence specific purpose oriented to carry out the most complex operations from a computational standpoint. Due to that, a huge attention is dedicated to HW accelerator architectures, which tend to enhance the performances of computationally heavy operations in particular algorithms, at the general end of enhancing their performances.

The computations could be derived in *spatial* or *frequency domain*, according to considerations regarding the complexity of the operation in

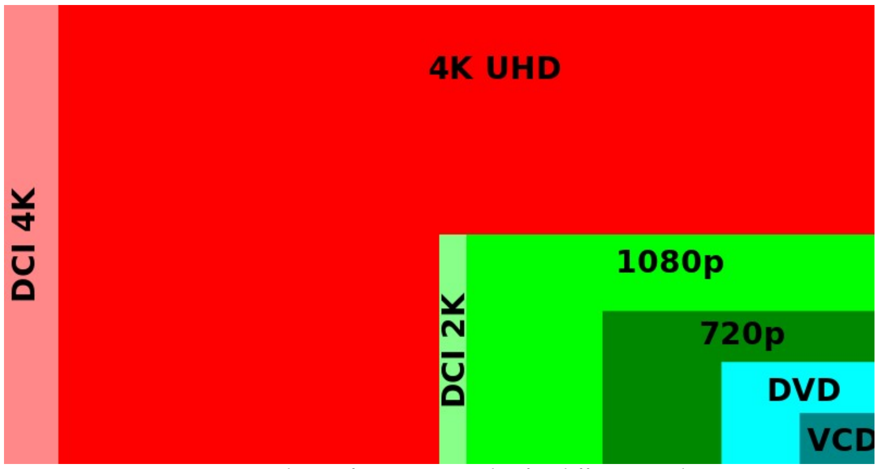


Figure i.4 *Resolution formats examples for different applications [https://it.wikipedia.org/wiki/File:Digital_video_resolutions_(VCD_to_4K).svg].*

Usually, spatial domain is preferred when possible and cost-effective, in order to avoid the introduction of transformation and anti-transformation techniques. Different techniques in both domains are now briefly explained, to gain an insight on the various problems that could arise in image processing applications.

i.3.1 Spatial Domain Techniques

Spatial domain processing describes techniques made up by several operations carried out on a group of neighboring pixels that could be mathematically described as:

$$g(x, y) = T\left[f(x, y)\right] \qquad \text{(i.1)}$$

being *f(x,y)* and *g(x,y)* the original and the processed images, while T[·] represents the particular operator applied on the original image. While it seems from the above formulation that operations could be performed only over single images, it is possible that operators are applied over several images, in case it is necessary to interpolate over the a set of images pixelwise. However the processing can be done on single pixels, the operations are usually carried out on tiles made up of neighboring pixels which could be squares, rectangles or circular approximations.

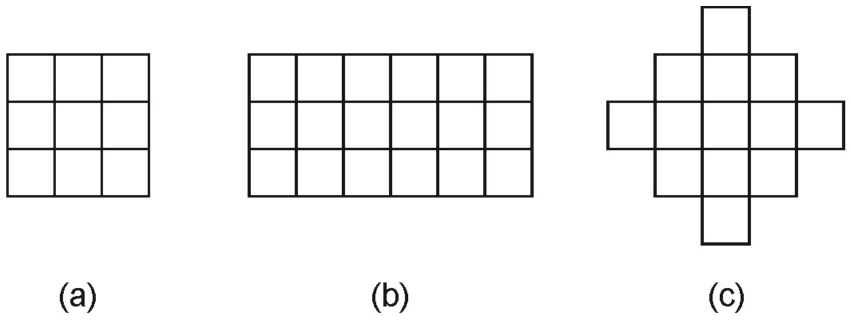


Figure i.5 *Examples of (a) square, (b) rectangular and (c) circular tile patterns in spatial domain image processing.*

Simple or complex operations could be performed according to the aim of the particular enhancement task, the masks used in the processing and the applied operator $T[\cdot]$. These kind of operations are also known as spatial *filtering operations* and the mask could also be referred to as *kernels*. It is worth to notice that even if we are approaching the problem as a 2D (Two-Dimensional) one, the same considerations apply to 3D (Three-Dimensional) domains and even to multidimensional clusters of data, more in general. Another important thing to highlight is that the size of the particular mask used in the processing could enhance or worsen the overall filtering results, depending on the particular mask used, the level of detail to be achieved by the processing, the timing constraints of the particular application and the resources to be used.

Examples of image transformations are represented by:

- Image negative;
- Log transformations;
- Gamma corrections;
- Piece-wise functions transformations;
- Image subtraction;
- Averaging;
- Thresholding;
- Histogram processing.

In linear spatial filtering applications the operations are carried out via multiplications and subsequent additions of the partial products obtained. This kind of operation is known as convolution and is carried out using Multiply and Accumulate (MAC) units. MAC operation represents one of the most computationally intensive in image processing, also because of the fact that it is an operation that has to be repeated several times during the processing; multiplications are still one of the most time consuming operations, depending on the kernel coefficients, image representation and the particular implemented architecture. Hence, in general, the response and thus the result associated to a particular pixel could be written as

$$g(x,y)=\sum_{m=-a}^{a}\sum_{n=-b}^{b}k(m,n)\cdot f(x+m,y+n) \qquad \text{(i.2)}$$

for a generic kernel having dimensions (*m*,*n*). The result of the above operation gives the spatial filtering for the considered tile of the image and it is then conventionally associated to the central pixel of the tile. Moreover, in order to avoid interpolations and further operations, tiles having a well-defined central pixel are usually considered (i.e. tiles having odd dimensions). Two examples of filter masks are reported in Figure i.6.

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

(a)

1	0	-1
2	0	-2
1	0	-1

(b)

Figure i.6 *Examples of (a) average and (b) horizontal Sobel filters.*

In this context great sharpening filters assume great importance. Different sharpening filters are used in various applications and have got great importance due to their key role in object recognition algorithms, among the others. Due to the role they will have in the further discussions, a brief examination of this class of filters is conducted.

i.3.1.1 Sharpening Filters

Sharpening filters are specifically used to highlight details in an image or to enhance blurred details in image and video multimedia contents. Several studies have been conducted on how to describe sharpening operation from a mathematical standpoint.

Several methods have been proposed using first and second order derivatives. It is important to notice that in the digital domain derivatives are defined in terms of differences and the first derivative operation has to respect the following properties:

– it must be zero among segments having pixels of constant values;
– it must be non-zero among segments having pixels of different values.

Following the same considerations, the second derivative operation has to respect the following properties:

– it must be zero in areas having pixels of constant values;
– it must be non-zero when the first derivative among the considered segments of pixels is non constant.

Moreover, even if the second derivative is computationally more intensive to calculate than the first derivative, it is often preferred to the latter one due to its better performances in term of fine detail recovery and for its higher sensitivity to step responses.

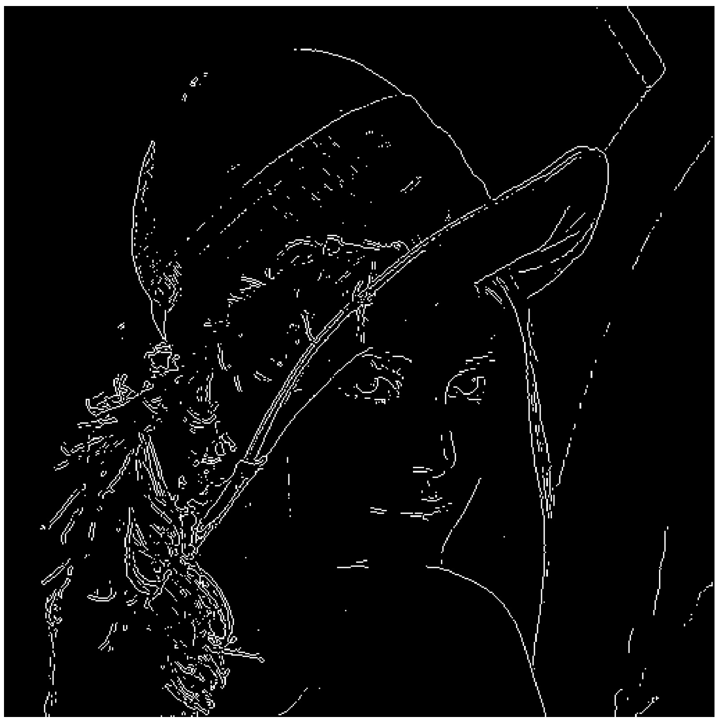


Figure i.7 *Examples of first derivative filtering using Sobel filters.*

The simplest isotropic second derivative two-variable function is the Laplacian as demonstrated in [Rosenfeld A., Kak A.C., 1982]

$$\nabla^2 f = \frac{\partial^2 f}{\partial^2 x} + \frac{\partial^2 f}{\partial^2 y} \qquad \text{(i.3)}$$

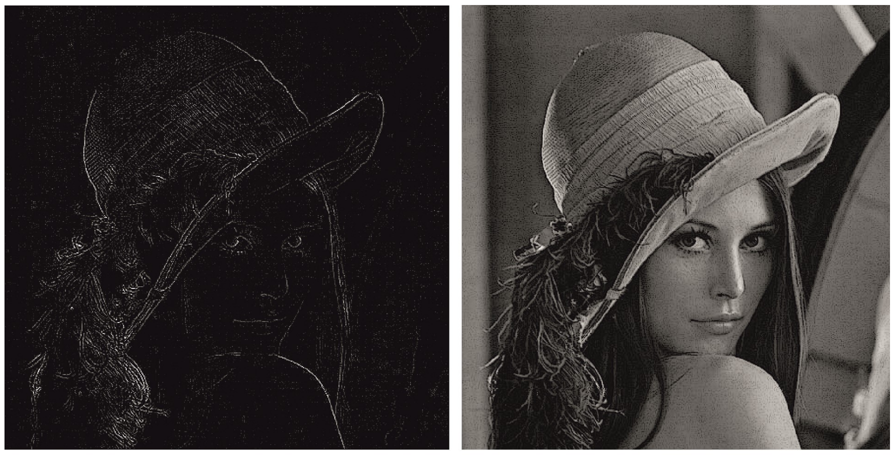


Figure i.8 *(a) Example of second derivative filtering using Laplacian filters for detail enhancement and (b) sharpening of original Lena image.*

Furthermore, in [Witkin, A. P., 1983], [Koenderink J.J., 1984] and [Lindeberg T., 1994] demonstrated that the descriptive equation of linear scale space is the one describing linear diffusion. The equation is a Partial Differential Equation (PDE) that could be written as

$$\frac{\partial \varphi}{\partial s} = c \cdot \nabla^2 \varphi \qquad \text{(i.4)}$$

satisfying a maximum principle according to which the amplitude of local maximums tends to decrease while decreasing the resolution, while local minimums show the opposite behavior. Furthermore, it is possible to demonstrate that normalizing the Laplacian it is possible to obtain scale invariance, which is a desired behavior in image recognition and feature extraction algorithms. In fact, as stated in [Mikolajczyk, K., 2002], [Mikolajczyk K., Schmid C., 2005] the use of the normalized Laplacian produces more stable function if compared to the Hessian function or the gradient. Obviously, due to the operators involved in the calculations, it is possible to state that the so built filters will give higher responses in the case of abrupt discontinuities among the values of neighboring pixels. The Laplacian function is mainly used to retrieve image details and enhance high frequency components (e.g. corners or edges) at the end of subsequently enhancing the original image for further use.

To conclude the discussion above spatial domain filters, it has to be noticed that also non-linear spatial filters have great importance in image processing, even if they are usually computationally more intensive than linear ones. They are often used in applications aiming to reduce noise and in image recovery.

i.3.1 ***Frequency Domain Techniques***

From Fourier signal theory it is known that it is possible to represent a periodic signal via a sum of periodic sine and cosine signals having different frequencies and amplitudes called Fourier Series (FS). However, Fourier theory also gives a mean to represent non-periodic signals surrounding a finite area, using the so called Fourier Transform (FT). The most important feature of FT and FS is that ideally it is possible to develop all the wanted operations in the frequency domain and then transform back the obtained results in the original domain having correct results and no information loss. In digital domain the Fast Fourier Transform (FFT) algorithm is of particular use, since it represents a fast way to calculate the Discrete Fourier Transform (DFT), which is the equivalent of the FT for discrete signals. The modern technique was developed in the 1965 by Cooley and Turkey [Cooley, James W. and John W. Tukey, 1965], but different formulations of FFT algorithms arose since then, due not only to exact algorithms, but also to several approximate algorithms, which result in faster calculations while maintaining the accuracy error under a certain limit.

2D DFT is defined as

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux/M+vy/N)} \tag{i.5}$$

while its inverse is given by

$$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(ux/M+vy/N)} \tag{i.6}$$

Any FT could be separated into a spectrum magnitude component and into a phase one given by

$$|F(u,v)| = \sqrt{\mathrm{Re}^2\{F(u,v)\} + \mathrm{Im}\{F(u,v)\}} \tag{i.7}$$

$$\theta(u,v) = \tan^{-1}\left\{\frac{\mathrm{Im}[F(u,v)]}{\mathrm{Re}[F(u,v)]}\right\} \tag{i.8}$$

In a frequency plot the DC component of the image concentrates in the origin of the plot, while the AC components tend to distribute farther from the origin with the growth of frequency considered. High frequency components, such as corners and edges, tend to be located in the upper right corner of such a plot.

Several properties make the frequency domain more suitable for image processing operations in particular cases. For example, one of the most important properties of the frequency domain is that the convolution of signals is substituted with the multiplication between the signals, thus simplifying this calculation stage. However, in several applications, working on small tiles of the input images it is still convenient to work in the spatial domain for the complexity related to the operations of transformation and anti-transformation and for the pre and post-processing operations related to the FT and Inverse Fourier Transform (IFT), which does not compensate for the simplification of the convolution operation. All the previously seen filtering applications could be also implemented in the frequency domain and each filter could be seen as a low pass, high pass or band pass filter.

In the following all the reported filters are presented both in spatial and frequency terms, in order to give the reader a better insight.

Chapter I
Bachet's Multipliers

I.1 Introduction

In recent years the need of elaborating both image and video high-quality media contents gave rise to an intense interest in the research activity, principally meant to the improvement of filters and MAC units. Usually, the first requirement to be taken into account in high-quality elaboration is the speed of these units, which is to say, the highest frame rate per second (fps) achievable by the system. Most of the times the chosen solution is to brutally allocate large number of arithmetic operators, which could cause in several cases, the slackening of the overall circuit. Other solutions in the modern literature recur to serialization and folding techniques techniques [G. D. Licciardo, A. D'Arienzo, A. Rubino, 2015], [Parhi K. K., 2007] or try to kick the problem intervening on the complexity of the algorithms. The former solution usually results in a significant reduction in performaces, while the latter approach, together with a careful design of the MAC units and of the overall architecture, remains the best way of achieving a good trade-off among Power, Performances and Area (PPA) requirements.

Some solutions are also based on the complete substitution of the multiplier units, while recurring to fast adders and shifters [Paul B. C., Fujita S., Okajima M., 2009], [Meher P.K., 2009] to achieve the same results for the particular operand codings, Canonical Signed Digit (CSD) and Modified Booth (MB), primarily [Hewlitt R. M., Swartzlantler E. S., 2000], [Tsoumanis K., Axelos N., Moshopoulos N., Zervakis G., Pekmestzi K., 2016]. It has to be highlighted that the filters simplification results particularly effective when at least one of the operands involved in the calculations can be reduced to a certain finite set of pre-calculated values, which is the case of pre-defined kernel filters. Distributed Arithmetic (DA) [Peled A., Liu B., 1974] techniques could be applied in these cases to obtain a partition of multiplication into simpler shift-and-add operations. Of course, in this kind of solutions the use of memories to store the pre-calculated coefficients to compute the final result is fundamental. However, it would be possible to reduce the overall memory usage by using Multiple Constants Multiplications (MCM) techniques [Voronenko Y., Püschel M., 2007.], [Aksoy L., Flores P., Monteiro J., 2014], which allow the DA use also in place of usual MB and CSD [Berkeman A., Owall V., M. Torkelson, 2000]. Nevertheless, DA techniques can lead to eccessive increment of mapped physical resources and, hence, a careful trade-off between its bit-serial operation and the partial sums parallelism needs to be found.

The candidate explored the possibility of development of a new partitioning scheme based on DA techniques to be used in high accuracy FP32 filtering applications having one integer input, based on the substitution of FP32 multipliers with FI adder chains without loss of accuracy. The obtained architecture is fit for multiple-constant filtering applications, working with FP and having a set of pre-calculated kernel coefficients and a finite range for the integer input values.

I.2 Mathematical Background

Claude Gaspard Bachet de Mèziriac was a well-known French mathematician from XVII, whose studies was mainly focused around arithmetic and number theory. Of particular importance was his "*Problem plaisants*" which was a collection of arithmetical tricks and questions. One of the problems the famous mathematician posed in this book is about integer partitions, stated as [O'Shea E., 2008]:

"What is the least number of pound weights that can be used on a scale pan to weigh any integral number of pounds from 1 to 40 inclusive, if the weights can be placed in either of the scale pans?"

Even if the problem was proposed for a single range of integer values, it is generalizable for any given range of integer values, given the theory and the theorem developed during the years and reported in what follows. The problem was called Bachet's Weights Problem after and several results on of number theory were derived from it.

The solution of the original problem is found using no more than four weights, namely {1, 3, 9, 27}, given that one could only add or subtract the different values to obtain the resultant integer. Table I.1 reports the different coefficients combination to obtain all the values in the original range. From the obtained results, it seems pretty obvious to deduce that is possible to obtain all the integer values in the range $\left[1; \frac{1}{2}(3^{n+1}-1)\right]$ using only the first n powers of 3. Such decomposition could be useful for different fields, e.g. to obtain the optimal cuts of coins and currency for economics [Tesler L. G., 1995], where it is however not used in favor of the easier and more intuitive decimal system [Van Hove L., 2001]. More important for us is the fact that it could be used in the simplification of the circuitry devoted to filtering apparatus [Vigliar M., Licciardo G. D., 2013].

During the years several demonstrations have been devoted to partitions and in particular to generalize Bachet's problem and theory to any kind of integer number range. The interest for this problem has been reignited in the beginning of the past century by one of the greatest mathematicians in

Table I.1 *Coefficients combinations to compose the numbers in the original Bachet's problem interval.*

Val.	Coeff.	Val.	Coeff.	Val.	Coeff.	Val.	Coeff.
1	1	11	9+3-1	21	27-9+3	31	27+3+1
2	3-1	12	9+3	22	27-9+3+1	32	27+9-3-1
3	3	13	9+3+1	23	27-3-1	33	27+9-3
4	3+1	14	27-9-3-1	24	27-3	34	27+9-3+1
5	9-3-1	15	27-9-3	25	27-3+1	35	27+9-1
6	9-3	16	27-9-3+1	26	27-1	36	27+9
7	9-3+1	17	27-9-1	27	27	37	27+9+1
8	9-1	18	27-9	28	27+1	38	27+9+3-1
9	9	19	27-9+1	29	27+3-1	39	27+9+3
10	9+1	20	27-9+3-1	30	27+3	40	27+9+3+1

number theory of the past century, G. H. Hardy, in one of the most famous and utilized textes in that field of study since then [Hardy G. H., Wright E. M., 2008]. These results were then reconsidered and extended mainly thanks to two papers in the last twenty years [Park S. K., 1998], [Rødseth Ø. J., 2006]. From the obtained results it is possible to demonstrate that, defining a set

$$W_r := \{3^0, 3^1, 3^2, \ldots, 3^{n-1}, R\} \tag{I.1}$$

where $R = r - (3^0 + 3^1 + 3^2 + \ldots + 3^{n-1})$, it is possible to write every integer in the [0, *r*] set as a superposition of the terms in W_r, each multiplied for a coefficient chosen in the group $C := \{-1,0,1\}$. The choice of the coefficients is however not unique, since it is possible to use only additions and use the set $C := \{0,1,2\}$ obtaining the same results. To sum up the various theorems obtained around this topic it is possible to state that

Theorem I.1 Defined as partition of a positive integer r the ordered sequence of positive integers that sum to $r = \lambda_0 + \lambda_1 + \lambda_2 + \ldots + \lambda_n$ where $\lambda_0 < \lambda_1 < \lambda_2 < \ldots < \lambda_n$ and the set $W_r := \{\lambda_0, \lambda_1, \lambda_2, \ldots, \lambda_n\}$ it can be demonstrated that:

1. every integer $0 \le q \le r$ can be written as:

$$q = \sum_{i=0}^{n} C_i \lambda_i \tag{I.2}$$

2. there does not exist another partition of *r* satisfying 1. with fewer parts than *n*+1.

Corollary II.1 Every partition of *r* is composed by exactly $n+1 = \lfloor \log_3(2r) \rfloor + 1$ parts.

Table II.2 shows the application of the generalized partitioning method. An example of the proposed decomposition using an integer range [0, 255] and hence six parts, could be given, e.g. for input “42” which can be rewritten as 42=(0)1+(-1)3+(-1)9+(-1)27+(+1)81+(0)134, namely the set of values from Table I will be {0,-1,-1,-1,+1,0}.

The above results can be applied to the development of MAC operations between a generic vector of coefficients A_k and a vector of integer limited inputs x_k:

$$y = \sum_{k=0}^{K-1} A_k x_k \tag{I.3}$$

and substituting (I.2) in (I.3), it results

Table I.2 *Coefficients combinations to compose the numbers in the generalized Bachet's problem for any integer interval.*

Values	Partition				
	λ_0	λ_1	λ_2	...	λ_n
1	1	0	0	...	0
2	-1	1	0	...	0
3	0	1	0	...	0
4	1	1	0	...	0
5	-1	-1	1	...	0
...	...	...	...	...	...
q	C_0	C_1	C_2	...	C_n
...	...	...	...	...	...
n	1	1	1	...	1

$$y = \sum_{k=0}^{K-1} \sum_{i=0}^{n} A_k C_{ki} \lambda_i \qquad (.4)$$

Considering that for several multimedia applications the parameter n is small and that all the integers in the range could be obtained with a linear superposition of the parts, the previously shown results could be used to write all the multiplications involved in the MAC operations and thus in filtering operations, recurring to a partition of pre-calculated terms, obtained multiplying a priori the product of A_k and the parts obtained for the partition range. In usual DA techniques the term x_k is decomposed as $x_k = \sum_{i=0}^{s-1} b_{ki} 2^i$, where $b_{ki} \in \{0,1\}$ represents the sign digit, in a way similar to the C_i coefficient in the Bachet's case. It is possible to rewrite (I.4) in the DA case in form of a partition as:

$$y = \sum_{k=0}^{K-1} \sum_{i=0}^{s-1} A_k b_{ki} 2^i \qquad (I.5)$$

were s is the number of bits of the binary representation of the input. Due to this parameter, it is possible to state that, although, the use of b_{ki} in place of C_{ki} contributes to reduce the number of logic elements used to implement (I.5), the actual values of s are significantly higher than n used in (I.4). Therefore, using Bachet's partition theory it is possible to strongly reduce the number of operators to implement the inner products in equation (I.4).

I.3 Architectural Design

I.3.1 Bachet's Multipliers

In all the considered implementations the incoming pixels are acquired in raster scan mode, directly from an image source, such like an image sensor. The method derived from the above reported theory has been used to implement the filtering scheme reported in Figure I.1 for the calculation of the convolution $G * I$ between a FP32 kernel, G, and an input unsigned integer vector, I, coded using m bits. The obtained architecture for a single multiplier is shown in Figure I.2. The decomposition of G and the parts involved in it are defined offline and from that it is possible to decompose each element of I in n+1 parts, according to Table I.2. The pre-multiplied coefficients obtained multiplying the parts and the kernels are stored in dual port Read Only Memories (ROMs) in a Look-Up Table (LUT) fashion, while another ROM dedicated to the C_i coefficients, coded on 2 bits each, stores the different combinations used to select the sign of the operands to

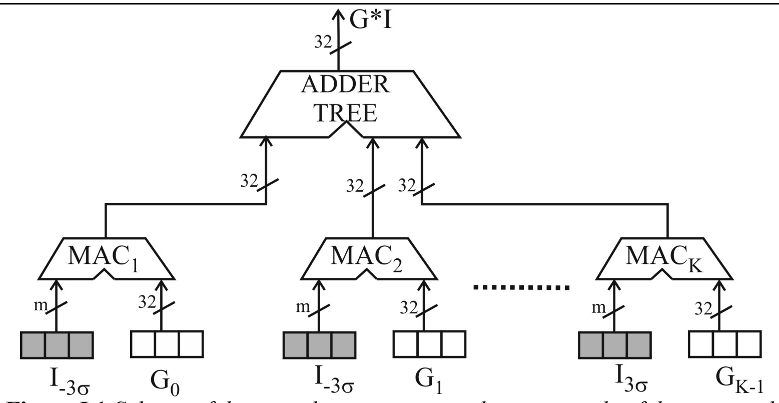


Figure I.1 *Scheme of the convolution circuit used as case study of the proposed decomposition method.*

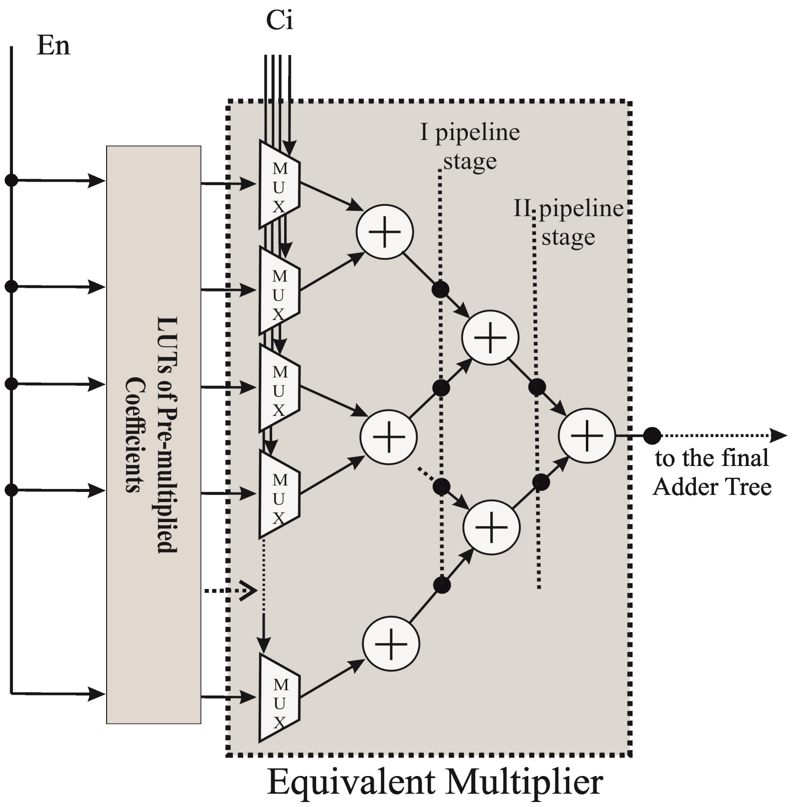


Figure I.2 *Scheme of the proposed multiplier, deployed in a Gaussian convolution.*

compose the final result. This further needed memory is composed by $(n+1)2^{m+1}$ bits. In the proposed architecture the multipliers are substituted using n adders, as shown in the dashed box of Figure I.2. To optimize the structure, the adders are distributed along a $\lceil \log_2(n+1) \rceil$ depth tree. Even if in principle FP adders should have been used, it is possible to derive a sort of custom coding of the partial results which allows us to reduce the overall operation complexity while not altering the multiplication final result. The peculiar used coding is derived starting from the standard IEEE754 FP32 coding [ANSI/IEEE754 - 1985, 1985] and increasing all the exponents of the pre-computed coefficients to the greatest one, contemporarily increasing the number of bits to code the significands accordingly, to avoid truncations during the operations and take into account all the shifted codes.

All the derived implementations have been targeted to both Field Programmable Gate Arrays (FPGAs), using a Xilinx Virtex 7 XC7V2000tflg1925-1, as part of the proFPGA DUO ASIC prototyping board [Xilinx, 2015], and std_cells, using TSMC CMOS 90nm libraries. The obtained data for a single multiplier are reported in Table II.3 and Table II.4.

Table I.3 *Synthesis of the proposed multiplier in comparison with recent FPGA oriented designs.*

	Bachet*	FPGA Conventional FP32*	[Arish S., 2015]
Target Platform	Virtex 7	Virtex 7	Virtex 4
LUTs	475	612	1545
Memory [*bytes*]	904	52	--
Worst Path Delay [*ns*]	2.456	8.235	4.77
Power** [*W*]	0.909	1.113	--

*Three stage pipeline
**Normalized at 100MHz

Table I.4 *Synthesis of the proposed multiplier in comparison with recent std_cells oriented designs.*

	Std_cells		
	Bachet*	[Basiri M. A., 2014]	[Själander M.**, 2009]
Target Platform	CMOS 90nm	CMOS 45nm	CMOS 65nm
Area [μm^2]	21269*** (5961/15308)	63682	52000
Worst Path Delay [*ns*]	2.61	2.18	2.5
Power**** [*mW*]	2.41	0.37	9.36
Area-Delay-Power [$\mu m^2 \cdot ns \cdot mW$]	$3.75 \cdot 10^4$	$5.16 \cdot 10^4$	$1.21 \cdot 10^6$

*Three stage pipeline
**Modified-Booth multiplier
***in parenthesis logic/memories
****Normalized at 100MHz

When possible IPs provided by Xilinx have been used, to obtain a fair comparison; to the same aim, both architectures are configured with a 3-stage pipeline. In the std_cell implementation Carry-Save Adders (CSA) have been used and no optimized conventional multiplier has been found for comparison in the same technology. The work in [Arish S., Sharma R.K., 2014], for FPGA, and [Basiri M. A., Sk. N. M., 2014], [Själander M., Larsson-Edefors P., 2009] for std_cells have been found for comparison.

From the comparisons in Table I.3 and Table I.4 it is possible to state that the FPGA implementation is the most advantageous one, thanks to the possibility of using hard macros to implement the ROM structures. Due to these reasons, the proposed implementation is capable of achieving a speed up of 335% if compared to a conventional FP32 multiplier, since using slow/slow corner simulations for both of the structures we obtain 2.456 *ns* and 8.235 *ns*, respectively. Moreover, the physical resources needed to implement the proposed architecture are 30% less than in the case of the design proposed by [Arish S., Sharma R.K., 2014] and also the worst path delay results almost halved.

For the std_cell implementation it is possible to state that it shows competitive performances against the two designs proposed by [Basiri M. A., Sk. N. M., 2014] and [Själander M., Larsson-Edefors P., 2009], using a two stage pipeline. Moreover, even if the comparison designs are implemented on shrunk technologies, we obtain similar worst path delays, while having a huge reduction in terms of the occupied area. Considering that the implemented design uses LUTs implemented memories and not optimized structure, the obtained results are encouraging towards an ASIC implementation of it. All the above considerations contribute to the have a Area-Delay-Power (ADP) product of $3.75\times10^{4}\,\mu m^{2}\cdot ns\cdot mW$, which is better than the ADP products of the structures in the comparison.

I.3.2 1D Filtering with Bachet MAC Units

The method could be applied to every particular kernel, but it has been targeted to a Gaussian filter for two main reasons: its large diffusion in multimedia elaboration and its computational complexity, which makes it one of the trickiest separable filters to work with. The kernel of a 1D Gaussian filter could be written in as

$$G(x,\sigma) = Ae^{-\frac{x^2}{2\sigma^2}} \tag{I.6}$$

Thanks to the separability property it is possible to decompose the 2D Gaussian filter in two 1D consecutive Gaussian filters. In that case, the implementation in the space/time domain is typically preferred to the frequency conversion, since usually the filters dimensions are not that big to justify for the FFT/IFFT conversion circuits.

Establishing the input range, given by m, as m=8, which is a common choice for representation of Chroma and Luma components in image processing, it is possible to state from the previously developed theory, that I could be partitioned using the first $n+1=\lfloor \log_3 2(2^8-1)\rfloor+1=6$ parts from Table I.2. The convolution operation could then be rewritten as

$$G(x,\sigma)*I(x)=\sum_{j=0}^{K-1}G_jI_{j-\frac{K-1}{2}}=\sum_{j=0}^{K-1}G_j\sum_{i=0}^{n}C_i\lambda_i=\sum_{j=0}^{K-1}\sum_{i=0}^{n}(G_j\lambda_i)C_i=\sum_{j=0}^{K-1}P_j \tag{I.7}$$

All the inner products, $G_j\,\lambda_i$, could be computed for each value of the input I in the given range [0,255] for every different kernel coefficient. This can be done due to the fact that G_j remains constant once σ and K have been defined, which can be done knowing that the minimum kernel length to obtain good accuracy of the results could be established in $K=6\sigma+1$ and that it translates in the storing of only $3\sigma+1$ values, due to the Gaussian symmetry. The input value is used to access the LUTs and choose the needed coefficients from them; the coefficients LUTs are in principle sharable among all the multipliers. The collected outputs coming from the ROMs are sent to a bank of multiplexer (MUX) to select the correct input for the adders. From what stated before, the other ROMs have depth $3\sigma+1$ and provide the terms that must be added toward the final results. Moreover, since it is not desirable to implement an exponent handling after every operation, we considered the greatest and the smallest coefficients resulting from this pre-computation, which are $G_0\lambda_n = \frac{\lambda_n}{\sqrt{2\pi\sigma}}$ and $G_{K-1}\lambda_0 = \frac{\lambda_0}{\sqrt{2\pi\sigma}} e^{-\frac{9\sigma^2}{2\sigma^2}} = \frac{e^{-4.5}}{\sqrt{2\pi\sigma}}$ respectively and given their ratio decided to increase the codelength of significands is increased of

$$\left\lceil \log_2\left(\frac{G_0\lambda_n}{G_{K-1}\lambda_0}\right)\right\rceil = \left\lceil \log_2\left(\lambda_n e^{4.5}\right)\right\rceil = \left\lceil \log_2(\lambda_n) + 6.5\right\rceil \tag{I.8}$$

to make it possible to carry out the intermediate operations using fixed point coding, which makes the implementation easier. This kind of considerations is general and could obviously be applied to any kind of different considered filter. For a single multiplier in that particular case, the above considerations would cause a FI coding on

$$l = \left\lceil 23 + \log_2(\lambda_n) + 6.5\right\rceil = \left\lceil \log_2(134) + 29.5\right\rceil = 37 bits \tag{I.9}$$

Another HW friendly simplification has been introduced to reduce the effect of this operand enlargement on the ROMs dimensions, omitting the exponent in the partial results and introducing it back only in the final one, in order to normalize the obtained data to the standard FP32 format. An exampling of the used coding is given in Table I.5 where it has been employed on the smallest coefficient with Uint-8 inputs and $\sigma = 4$.

The amount of required memory to implement the proposed solution is reported in Table I.6, where also a comparison with a canonical radix-2 DA implementation is given.

Table I.5 *Custom coding applied on the smallest pre-multiplied coefficient with Uint-8 inputs and $\sigma = 4$.*

Smallest Coefficient	r	λ_n	*FP32 coding of 1.1x10^{-3}*
1.1x10^{-3}	256	134	*0 01110101* ***00100000010110111100000***
			Modified coding of 1.1x10^{-3}
			0000000000001001000000101101111100000

Table I.6 *Required memory as a function of the input range.*

	$C_i+\lambda_n$ (C_i) [Kbits]	
Input length [bits]	Bachet	radix-2 DA
8	5.958 (3.072)	5.896 (2.048)
9	10.535 (7.168)	8.937 (4.608)
10	17.703 (14.336)	15.050 (10.240)
...	...	...
m	$\lambda_n=(n+1)[(3\sigma+1)l]$ $C_i=(n+1)2^{m+1}$	$\lambda_n=m\ [(3\sigma+1)l]$ $C_i=m\ 2^m$

The graph in Figure I.3 reports the required number of additions as a function of the input length, *m*. It is worth to notice that for $m>4$ the proposed solution is always better than the canonical DA in terms of required addition operations, while the required memory becomes significantly higher only for $m>9$, which is to say, when the lower number of parts becomes to be compensated by the number of bits required for C_i.

Of course, the advantages of the proposed solution are evident when a large number of multipliers is required. For the first example of implementation given next, where $m=8$ and $\sigma=4$, 25 MAC structures are required to implement a full-parallel circuit. Then, using the proposed decomposition is possible to save 50 adders from the case of a radix-2 DA. The shareability between all the MACs of the memory structures, the proposed solution proves itself better than a canonical radix-2 DA to implement filters with fixed kernel dimensions.

Table I.7 reports the results for the implementation of a 1D Gaussian filtering circuit based on the proposed multiplier architecture. The filter is made up by 25 multipliers and an output adder tree to perform the final addition of the partial results. In this case, the FPGA implementation achieves a 824% speed-up, as the worst path delay passes from 16.986 *ns* to 2.981 *ns*. At the same time the LUTs count is decreased by 44.21%.

On the other hand, in std_cell implementation we achieve an area reduction of 19.52% and a speed-up of 11.93%. The implemented design has in this case a higher power dissipation, mainly due to the non optimized memory structures, which show high dynamic power dissipation. It is worth to notice that in the proposed implementation all the ROM memories have to be read on the same clock edge from the multipliers; even if on FPGA this does not require particular attention, this does not apply for the ASIC implementation. In fact, the latter requires a custom implementation of very small ROMs in order to show good performances, as shown in [Paul B. C., Fujita S., Okajima M., 2009].

Finally, it is important to highlight that the amount of memory required for such a design would not represent a real problem for image and video

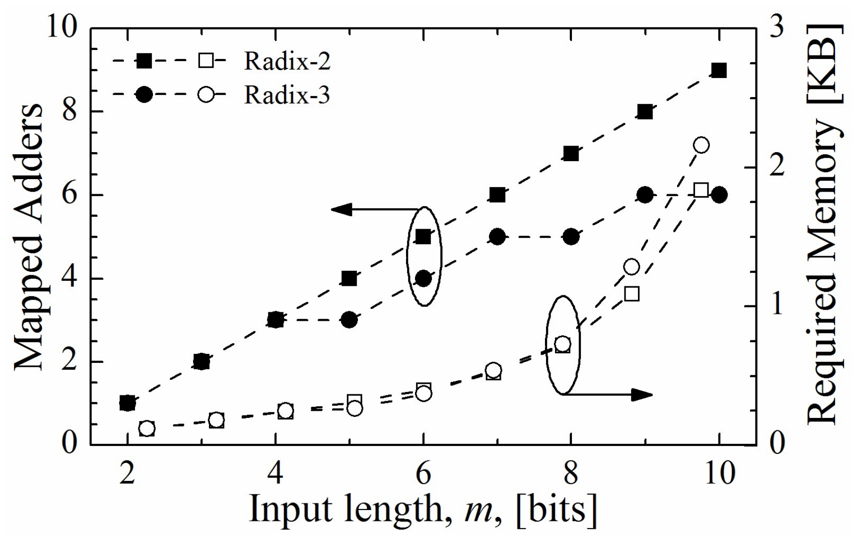


Figure I.3 *Comparison between required resources of the proposed partitioning method and the radix-2 DA for 1D filtering, with l=37 and (3σ+1)=13; the number of adders is given for a single multiplier.*

processing applications, since in these cases buffer memories in the order of Mbits are usually required, making the amount of memory required for the storage of the coefficients often negligible.

Table I.7 *Synthesis results of the 1D Gaussian Convolution Circuit.*

	FPGA		Std_cells	
	Bachet	Conv. FP32	Bachet	Conv. FP32
Platform	Virtex 7	Virtex 7	90nm	90nm
LUTs/Area [μm^2]	8654	15512	275339	342127
Mem. [*bytes*]	904	--	904	--
Delay[*ns*]	2.981	16.986	3.99	4.531
Power* [*W*]	2.317	4.495	$28.011\cdot10^{-3}$	$16.907\cdot10^{-3}$

*Normalized at 100MHz

I.3.3 2D Filtering with Bachet MAC Units

The achieved results on the multiplier unit and the 1D filter have then been used to implement a 2D non separable generic filter for which equation (I.7) becomes

$$\begin{aligned} O(x_0,y_0) &= \sum_{h=0}^{K-1}\sum_{j=0}^{K-1} F(h,j) I\left(x_0+h-\frac{K-1}{2}, y_0+j-\frac{K-1}{2}\right) = \\ &= \sum_{h=0}^{K-1}\sum_{j=0}^{K-1} F_{h,j} \sum_{i=0}^{n} C_i\left(x_0+h-\frac{K-1}{2}, y_0+j-\frac{K-1}{2}\right)\lambda_i = \\ &= \sum_{h=0}^{K-1}\sum_{j=0}^{JK-1}\sum_{i=0}^{n} (F_{h,j}\lambda_i) C_i\left(x_0+h-\frac{K-1}{2}, y_0+j-\frac{K-1}{2}\right) \end{aligned} \qquad \text{(I.10)}$$

and starting from this form it is possible to derive a convolutional architecture capable of directly performing the operation in 2D. In that case, since it is not possible to implement a Serial Input Parallel Output (SIPO) structure, and hence we have to reproduce the modules generating the C_i coefficients. For this reason, we limited the dimension of the tile to 3x3, which however is a really common size in several multimedia applications, like Visual Search (VS). The method could be improved using further techniques to improve the memories behavior or recurring to folding and reuse of the structures in case the tiles dimensions become too big, in case of necessity. In this case, the coding has to be further modified to take into account the 2D nature of the filtering. In fact, repeating the same calculations as before we obtain

$$G(x,y,\sigma) = Ae^{-\frac{x^2+y^2}{2\sigma^2}} \qquad \text{(I.11)}$$

which, given the same considerations previously developed, results in

$$\left\lceil \log_2\left(\frac{G_0\lambda_n}{G_{K-1}\lambda_0}\right)\right\rceil = \left\lceil \log_2\left(\lambda_n e^9\right)\right\rceil = \left\lceil \log_2(\lambda_n) + 14\right\rceil \qquad \text{(I.12)}$$

$$l = \left\lceil 23 + \log_2(\lambda_n) + 14\right\rceil = \left\lceil \log_2(134) + 37\right\rceil = 44 bits \qquad \text{(I.13)}$$

A block scheme of the proposed implementation is shown in Figure I.4, where it is possible to notice how the system could be divided into two main sub-systems:

- the memory module, which codes the input data;
- the filtering module, which carries out the calculations related to the considered tile.

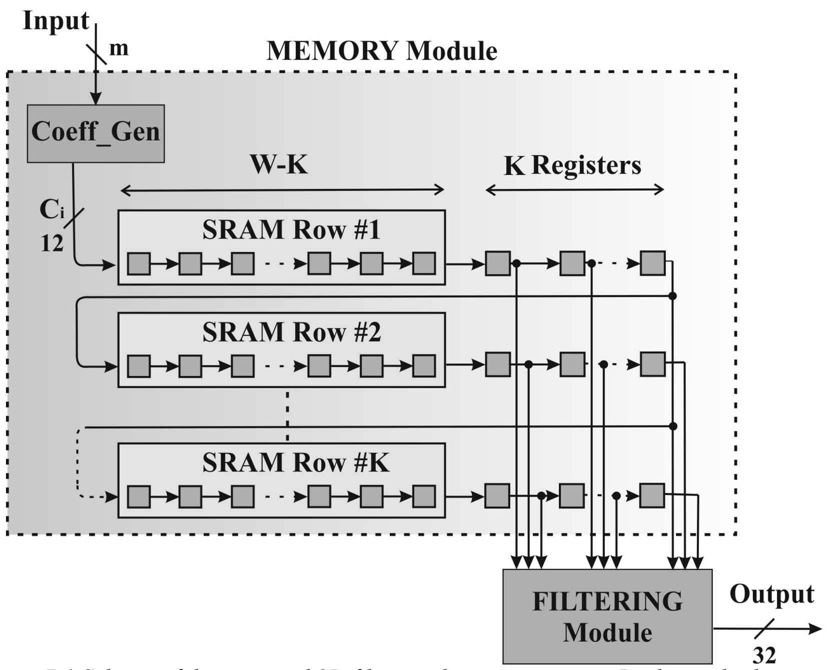


Figure I.4 *Scheme of the proposed 2D filter implementation using Bachet multipliers units.*

The incoming pixels are sent to the conversion module, where the coefficients for each one of them are derived from the coefficients LUT. The coefficients are then sent to a buffering structure capable of storing a number of coefficient vectors equal to the width of the considered image, W, multiplied for the vertical dimension of the kernel; for the considered case it translates in a 640×3 strucuture (W=640), where each location store l_C=12 bits. The structure is a SIPO folded like a stripe buffer, giving as output all the coefficients of the considered tile at the same time. It is important to note that such a structure aligns automatically the new incoming pixels, since after the filtering operations all the values of the SIPO are shifted by one position; hence, the architecture does not need any further circuit to reshuffle or reorder the values. A straightforward implementation of the memory module would in principle be done using registers, but would not represent an optimal solution given the amount of physical resources to be allocated. Considering the proposed case, it is possible to state that for a VGA image (W=640) and a kernel dimension K=25, the needed resources would be 640×25×12 bits =188 Kbits. Due to this kind of considerations it would be better to implement the proposed structure using Static Random Access Memories (SRAMs) to emulate the SIPO structure in a buffer fashion. It also has to be highlighted that due to these choices it is possible to implement the structure via dual-port SRAMs, allowing the reading and the writing of the

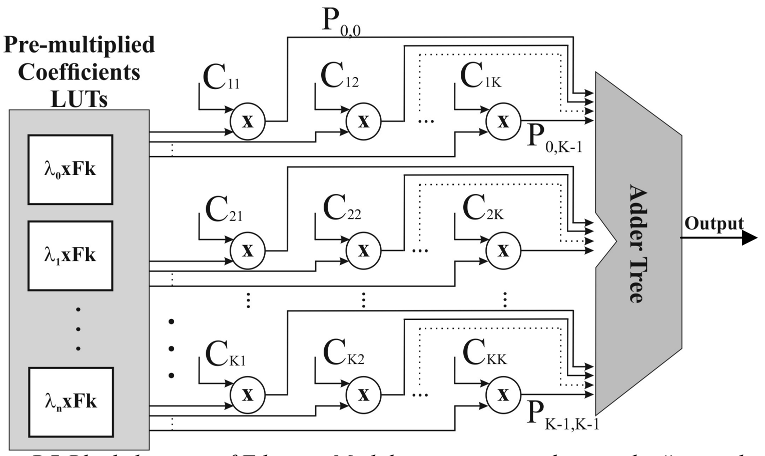


Figure I.5 *Block diagram of Filtering Module, representing the way the "equivalent" multipliers are interconnected.*

data during the same clock cycle. The dimensions of any of the SRAM structures is $l_C \times (W-K)$, while the parallel reading of the $K \times K$ tile of interest is obtained using K registers to complete each SRAM row.

Figure I.5 presents a block diagram representation of the filtering module, with the interconnections of the various units and the Bachet's multipliers block. As in the previous case, the LUTs are not only storing the K coefficients, but also their 2's complement to be selected when the coefficient $C_i=-1$. As it has been done in the previous case, the length of the operands have been increased and a custom coding has been developed for the particular case; starting from the same consideration we obtained, establishing F^{min} and F^{max} as the minimum and the maximum values of the used kernel a codelength

$$l_s = \left\lceil 23 + \log_2\left(\frac{F^{\max}\lambda_n}{F^{\min}\lambda_0}\right)\right\rceil \tag{I.14}$$

where 23 bits is the length of the standard FP32 significand. The normalization stage is introduced at the end of the overall computation to normalize the output in a standard FP32 format just one time and avoid intermediate normalizations.

To have further insights on the developed structure and have comparison to the related literature, the filter processor has been implemented with a 2D symmetric Gaussian kernel

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{4\sigma^2}} \tag{I.14}$$

A square kernel with dimension K=3 has been implemented, since it is the minimum usable dimension for VS applications. The range of input values has been chosen as r=256, using Uint-8, hence l_c=12 bits also in this case. Due to these considerations it is possible to write

$$G^{\max} = G(0,0,\sigma) = \frac{1}{2\pi\sigma^2} \qquad \text{(I.15)}$$

$$G^{\min} = G(3,3,\sigma) = \frac{1}{2\pi\sigma^2} e^{-9} \qquad \text{(I.16)}$$

and from that results a length of the significands

$$l_s = \left\lceil 23 + \log_2\left(\lambda_n e^9\right) \right\rceil = 44 bits \qquad \text{(I.17)}$$

The curve reported in Figure I.6 generalizes the results for different

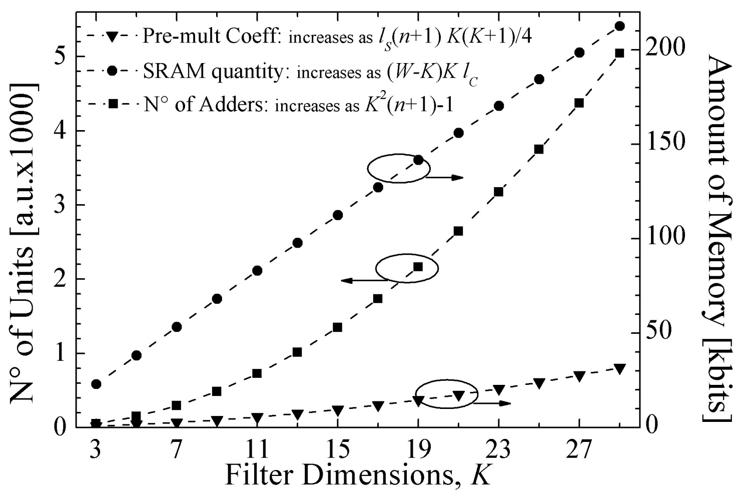


Figure I.6 *Required resources of the 2D convolution-based filter as a function of its dimensions, when m=8 bits and W=640 pixels.*

dimensions of the kernel tile.

Using the same platforms of the previous implementation we obtained the results reported in Table I.7.

A fair comparison evaluation has been achieved using Xilinx IPs for arithmetic structures and configuring the different implementations using a three stage pipeline. The total requirement in terms of memorized coefficients is of 904 *bytes*, while it is clear the advantage of the proposed structure on a FPGA platform precisely because of the use of hard macros for the memory implementation. The critical path delay shows a speed-up of 371% with respect to a conventional FP32 multiplier and the physical resource usage is 38.6% less than in the latter case.

Table I.7 *Synthesis results of the 2D Gaussian Convolution Circuit.*

	FPGA		Std_cells	
	Bachet	Conv. FP32	Bachet	Conv. FP32
Technology	Virtex 7	Virtex 7	90nm	90nm
LUTs/Area[μm^2]	4750	7732	294217	386768
Mem. [*bytes*]	904	--	904	--
Worst Path Delay*[*ns*]	4.700	17.432	4.426	8.717
Power** [*W/mW*]	0.684	0.907	13.980	13.594

*1 pixel filtered per clock cycle
**Normalized at 100MHz

In std_cell the considerations are slightly different; in fact, while the proposed structure achieves a 94.8% speed-up and a 24% area reduction, the power dissipation is higher than in the conventional case, mainly due to the dynamic power dissipated in the memory structures. It is worth to highlight that even in this case the amount of memory required does not represent an actual problem in real multimedia applications, since it is negligible compared to the other memory structures involved. Table I.8 shows a comparison with the existent literature.

Table I.8 *Comparison among the obtained results and the ones found in literature.*

	FPGA		Std_cells	
	Bachet*	[Huang F.C., 2012]	Bachet	[Cobello F., 2015]
Technology	CMOS 90nm	CMOS 180nm	Virtex 7	Spartan 6
Area /LUTs [μm^2]	294217	N.A.	4750	5052
Memory [*bytes*]	276 kb	224 kb	904 b	--
Worst Path Delay*[*ns*]	4.426	10	4.7	10
Power** [*mW/W*]	13.980	N.A.	0.684	N.A.

*scaled to 3 octaves, 6 scales **Normalized at 100MHz

Raw comparisons could be carried out scaling our design to the same dimensions of [Huang F.C., Huang S.Y., Ker J. W., Chen Y. C., 2012], which implements a 2D Gaussian filter to develop Scale Invariant Feature Transform (SIFT) filtering stage, in a three octaves, six scales implementation. In this case, even if the calculations are developed on 24 bits FI coding, the greater quantity of memory required in the proposed design is compensated by the lower number of adders and multipliers. The design in [Cobello F., Leon J., Iano Y., Arthur R., 2015] instead presents a 16 bits implementation on a Xilinx Spartan 6, having the same granularity of the used Virtex 7. Even in the FP16 case, the proposed solution reveals advantageous both in worst delay path delay time and area occupation, proving the solution is really fit for bounded integer inputs implementations.

I.3.4 Memory Reduced Bachet's Multipliers

Another implementation has been developed to overcome the problems related to the memory occupation in memory constrained applications. The

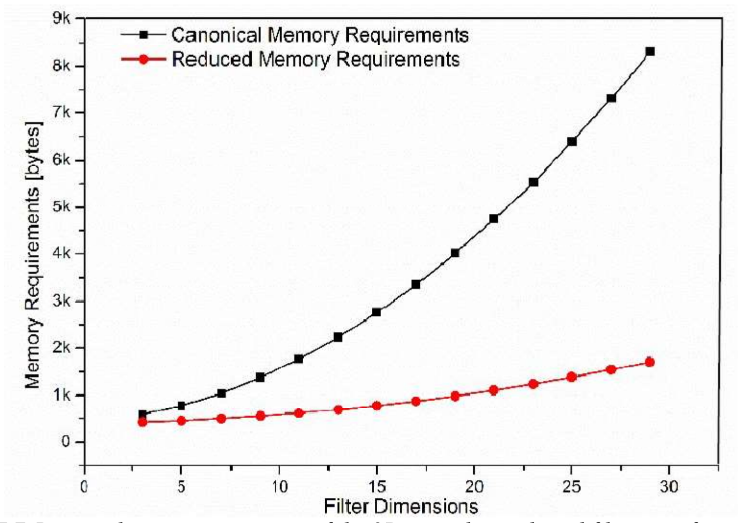


Figure I.7 *Required memory resources of the 2D convolution-based filter as a function of its dimensions, when m=8 bits.*

amount of memory to be stored increasing the considered tile dimensions are reported in Figure I.6. However, Figure I.7 shows the results obtained using the proposed architecture for memory reduced implementations.

The amount of memory to be used is related to the input range to be considered and it is only due to the memory storing the parts of the Bachet decomposition, which undergoes to a huge increase with the increase of the filter size. An analytical expression for this increase in the number of bits is shown in equation I.18

$$M_{bits} = 2l_s(n+1)\sum_{i=1}^{\lceil K/2 \rceil} i \qquad \text{(I.18)}$$

It could be seen that due to the memory increase it is possible to obtain unacceptable results for particular constrained implementations. In this kind of implementations a method of deriving the other coefficients starting from the coefficients $F_{h,j}\,\lambda_0$ of the coefficients related to the first parts and theirs

2's complement counterparts. The other coefficients are obtained thorugh shifting and adding operations starting from these, since a left bit shifting operation corresponds to a multiplication by a factor 2. The obtained results and the correspondent operations are reported in Table I.9 in the case of a 8 bits input for all the needed coefficients.

Table I.9 *Detailed shift and add operations for an 8 bits input design.*

	Operation	Number of Shifts	Number of Additions
λ_0=1	--	0	0
λ_1=3	2+1	1	1
λ_2=9	2^3+1	3	1
λ_3=27	2^5-2^2-1	7	2
λ_4=81	2^6+2^4+1	9	2
λ_5=134	2^7+2^2+2	10	2
$-\lambda_0$=-1	--	0	0
$-\lambda_1$=-3	-2-1	1	1
$-\lambda_2$=-9	-2^3-1	3	1
$-\lambda_3$=-27	-2^5+2^2+1	7	2

This approach reduces the amount of embedded ROMs to be implemented on the targeted FPGA at the cost of allocating more logic and algebraic elements.

From that kind of implementation the results of Table I.10 are obtained. The implementation is been targeted only to FPGA since memory constrains are problematic particularly in that type of implementations.

Table I.10 *Synthesis results of the 2D Gaussian convolution circuit using the reduced memory implementation.*

	FPGA
	Bachet Reduced Memory
Technology	XC7V
LUTs	7760
Memory [*bytes*]	417
Delay[*ns*]	4.720
Power* [*mW/W*]	0.728

*Normalized at 100MHz

Comparing the results for this implementation to the ones in Table I.7 it is possible to state that the proposed reduced memory implementation has got approximately the same delay path of the original Bachet's implementation, while consuming 63.3% more area and 6.4% more power.

The memory usage is however reduced by approximately 28.4% and the performances would be even better in the case of taking into account larger filter sizes.

Table I.11 shows the fps performances on various image resolutions for the various implemented architectures.

Table I.11 *Performances of the reported designs in terms of fps varying the resolution of the frames.*

	Bachet	*Bachet Red. Mem.*	*Conv. FP32*	*MB*	*Prop. FP16*	*Cabello*
VGA (640x480)	692	689	186	370	471	325
Full-HD (1920x1080)	102	102	27	54	69	48
4K UHDTV (3840x20160)	25	25	6	13	17	12

I.4 Conclusions

Several ways of implementing an easy, compact and low power consuming circuit devoted to the calculation of multiplications and MAC operations and using a decomposition of the operands in smaller factors has been implemented. The method well applies for limited range operands in the development of FP32 calculations. The partitioning method allows implementing the circuitry for convolution operators, which are typically employed in filters, without multipliers, encoders and auxiliary circuitry. These are completely substituted by simplified adders and memory structures for storing pre-multiplied coefficients. Moreover, a simplified implementation has been proposed, for memory constrained applications. The proposed solutions obtain state-of-the-art performances. The solutions are well suited for the application of multi-constant multiplication techniques, in order to further simplify the circuital topology.

Chapter II
LDR2HDR Filters HW Design

II.1 Introduction

Starting from the previous work on filtering structures, their implementations and the acquired knowledge in FP coding, we tried to improve a system first developed by the laboratory group [G. D. Licciardo, A. D'Arienzo, A. Rubino, 2015], making it more feasible to work on different image dimensions and thus making it possible to show LDR images on HDR screens without having to resize them when a change in the resolution occurs.

During the last years HDR applications become more and more popular due to the possibility of obtaining a more faithful representation of human vision in image and video applications. The main problem in fidelity in multimedia issues is related to the limited range to represent pixel values and correctly display them on a screen. The definition of dynamic range is the ratio between the darkest and the brightest points of a scene, which could be determined in digital processing as

$$DR = \frac{M\_pix}{m\_pix} \tag{II.1}$$

where *M_pix* represents the maximum pixel value of the considered image while *m_pix* represents the minimum pixel value of the same. DR is usually measured also using *stops*. A stop represents a variation of the brightness of a factor 2. From this simple definition it is possible to state that a traditional LDR camera is capable of 8 f-stops, since the data are coded on 8 bits. On the contrary the human eye is capable of discriminating up to more than 20 f-stops. This difference between the two values leads to the huge perspective differences between a multimedia content and the eye viewed scene. To give an insight on the importance of such a matter, it could be reminded that a classic film camera achieves 15.5 stops, meaning it was far more faithful in representing the light of a scene having high values of dynamic range. Moreover, special developed films and devices where capable of achieving up to 26.6 f-stops [Wyckoff C.W., 1969]. For all the reasons previously stated, a lot of researchers began to try to expand the dynamic range of digital content since the beginning of the ‘90s. The research became more and more important in the last decade, thanks to the development of filming techniques, digital cameras and screens aiming to use higher definitions and to give the viewer a more realistic experience.

Some of the problems related to that task are:

- the high amount of data to store to correctly represent the high dynamic content;
- the costs of HDR screens and in general of HDR devices is still high if compared to other technologies;
- the methods to show a LDR image on a HDR device.

The above described issues translate in different approaches and solutions to achieve cheaper, more efficient and standardized HDR devices. In particular, in the case of capturing devices, multi sensor systems have been developed and used to achieve the reconstruction of the final image while avoiding artifacts and halo effects. However, the research is far from reaching a turning point, since it is not easy to realize consistent and reliable cameras using multiple sensors for high resolution images. Nevertheless, several prototype have been developed, achieving performances of 20 f-stops and 30 fps, but they are far from getting to the market. Furthermore, the need for data storage became a major issue, due to the fact that a single frame at 4K-Ultra High Definition TV (UHDTV) needs 95 *Mbytes*, meaning 167 *Gbytes* for any minute of video at 30 fps [Chalmers A., Debattista K., 2017]. Also, a lot of effort is now put on finding new data compression methods, dedicated to HDR processing and to the development of quality metrics, due to the fact that there is no uniform and well-defined metric or a standardized format to evaluate and display HDR content. However, either FI with large fractional parts or FP solutions are used to expand the lighting range representation.

In our research, we focused on the problem of developing a HW design capable of generating a HDR result starting from a single LDR frame. This is needed step, in order to develop HDR screen technologies capable of showing also old LDR content, not needing multiple exposure images of the same frame, which is usually not available. This problem is the inverse of the one of mapping HDR images on LDR frames, which represent a well-known problem, solved thanks to different transformation techniques [Reinhard E., Ward G., Pattanaik S., Debevec P, 2006], [Myszkowski K., Mantiuk R., Krawczyk G., 2008], [Banterle F., Artusi A., Debattista K., Chalmers A., 2011] and known as Tone Mapping (TM). The opposite problem, inverse Tone Mapping (iTM), has been raised only in the recent years [Masia B., Agustin S., Fleming R. W., Sorkine O., Gutierrez D., 2009], [Banterle F., Chalmers A., Scopigno R., 2013]. The problem arose due to the higher availability of HDR reproduction devices compared to HDR acquisition systems, which make it necessary the development of techniques to show the common LDR contents on HDR reproduction devices. Several algorithms working on different exposure images and combining them was first proposed [Masia B., Agustin S., Fleming R. W., Sorkine O., Gutierrez D., 2009], [Debevec P. E., Malik J., 1997]. Later, also expansion techniques starting from a single LDR frame and using an expansion map has been proposed and developed; these techniques could be differentiated in local [Banterle F., Ledda P., Debattista K., Chalmers A., 2008], [Rempel A. G., Trentacoste M., Seetzen H., Young H. D., Heidrich W., Whitehead L., Ward G., 2007] and global operators techniques [Masia B., Agustin S., Fleming R. W., Sorkine O., Gutierrez D., 2009]. Even if global methods gives good results from a perspective point of view [Akyüz A. O., Fleming R., Riecke B. E., Reinhard E., Bülthoff H. H., 2007], analytical measurements and perceptual tests have been carried out show that when possible local methodology is preferable. The better results come at the cost of a higher computational complexity [Banterle F., Ledda P., Debattista K., Bloj M., Artusi A., Chalmers A., 2009], with the consequent difficulty of obtaining real-time performances. In fact, unless of recurring to several algorithm simplifications [Rempel A. G., Trentacoste M., Seetzen H., Young H. D., Heidrich W., Whitehead L., Ward G., 2007] or using hard computing Central Processing Unit/Graphic Processing Units (CPU/GPU) combined systems [Kovaleski R. P., Oliveira M. M., 2009], the latencies of the SW implementations make these systems not fit to meet real-time performances unless using low resolution images. In contrast it is possible to develop a HW Application Specific Image Processor (ASIP) capable of carrying out the calculations related to these algorithms up to 4K-UHDTV resolution, achieving almost real-time performances on FPGA and real-time performances in std_cell implementation.

Before going on with the description of the implemented architecture and used techniques, a brief review of color spaces and coding techniques for pixels is provided, at the end of developing an unambiguous and clear description of the ideas involved in the following description.

II.1.1 Color Spaces and Encodings

A color space, also known as *gamut*, is a set of colors representing all the color that a certain screen is able to display or, in general, the set a certain device is able to reproduce. A lot of different color spaces have been created for various applications and to improve the understanding of how to obtain better color displaying on devices. However, all the considered color spaces span a smaller space than the Lab color space, representing the range of colors a human eye can see. A lot of color spaces exists, for different purposes and different applications; nevertheless, the most used ones and the two here examined are RGB and Y’UV, which represent two device independent models. An example of RGB and Y’UV components for the same image is given in Figure II.1.

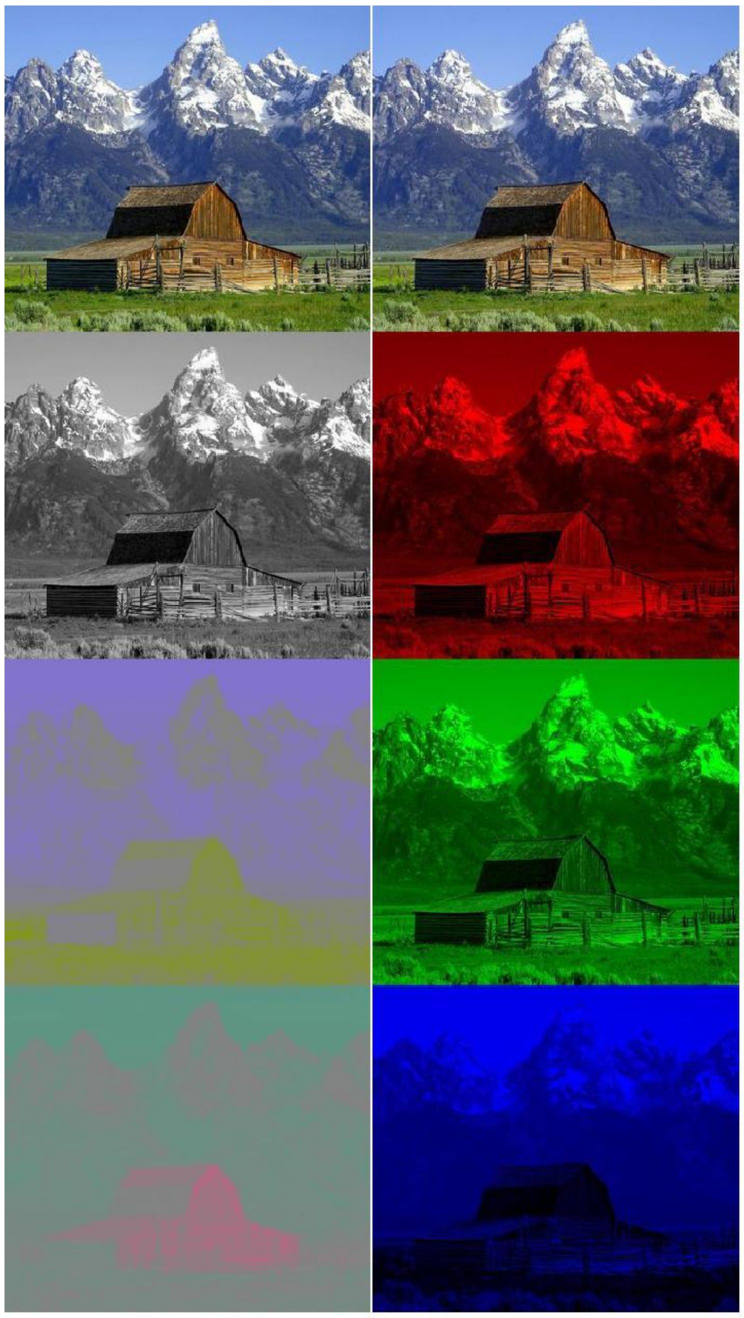


Figure II.1 *Y'UV and RGB components example [http://resizing.info/p2s.html].*

II.1.1.1 RGB Color Space

An RGB color space is any kind of color space defined on additive operations among these three components. The color space will be composed by all colors that could be made up using red, green and blue components.
RGB systems are often used in computer graphics and in general nowadays consumer's displays, since the color space provides a good representation of the human eye perception. However, high end devices often work on color spaces having larger spans if compared to the standard RGB (sRGB) one. This color space is based on three channels for red, green and blue components, each of them coded on 8 bits. That is why alternative RGB color spaces, such as Adobe RGB, have been developed color spaces covering larger spans. However, even if sRGB is still the standard color space for most of the consumer devices, new models capable of spanning larger color spaces are acquiring more and more importance. Among them one of the main alternatives to RGB color space is represented by Y'UV color space.

II.1.1.2 Y'UV Encoding

Y'UV encoding definition has its roots in the analog transmission of color images for the display on black and white devices. It was necessary to have a signal suitable for black and white televisions that could be used also on color screens in combination with additional signals for the chrominance. The signal chosen for such a task was the luminance component (also known as Luma), while the additive signals were chosen as ultraviolet (UV) because they are color difference signals, which allows us to modify the image using pixels shifts without altering the brightness of the image.

One of the main advantages of Y'UV systems is related to the lower bandwidth needed. In fact, the human eye is more sensible to changes in brightness than it is to changes in spatial sensitivity to colors. For this reason it is possible to allocate less bandwidth for the UV or chrominance (Chroma) components. This results in a signal compression due to the subsampling, which is usually done halving the Chroma horizontal resolution (4:2:2 subsampling) or both the horizontal and vertical resolution (4:2:0 subsampling), while very few devices still use the uncompressed (4:4:4) solution. However, Y'UV is not an absolute color space, but represents a way of encoding RGB information in a lossy way. Nevertheless the brightness range allowed by an RGB coding is far lower than the one allowed by a Y'UV coding. Due to that, Y'UV coding is usually preferred for HDR applications in which the range of the luminance of the frame represents the main issue.

Matrix transformation between RGB and Y'UV encoding are reported in equations II.2 and II.3.

$$\begin{bmatrix} Y' \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51499 & -0.10001 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{II.2}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.13983 \\ 1 & -0.39465 & -0.58060 \\ 1 & 2.03211 & 0 \end{bmatrix} \begin{bmatrix} Y' \\ U \\ V \end{bmatrix} \tag{II.3}$$

In the following a Y'UV encoding is considered for the development of the proposed HDR system.

II.2 Algorithmic Considerations

For what previously stated, we choose a method capable of obtaining a HDR image starting from a single frame, using a local operator and working on Y'UV encoded data. In particular, the last choice gives us the possibility of performing only the expansion of the Luma component, which is the most important for the human eye perception, while maintaining the Chroma components in their compressed states, saving bandwidth and memory, while obtaining HDR compatible results from the computation.

The general behavior of the proposed algorithm could be explained in two phases:

- the LDR Luma image undergoes an intermediate expansion using a sigmoid function;
- a second selective expansion is performed, using a further filtering stage, to obtain a further enhancement for darker and lighter regions of the intermediate frame.

The input Luma components of the LDR original image, $L_D(x,y)$, are taken as inputs and used to calculate the incremental average, $L_a^{inc}(x,y)$, using the sigmoid function derived from inverting the Tone Mapping operator. Analytically, we obtain

$$L_M(x,y) = \frac{1}{L_a^{inc}(x,y)} \frac{L_D(x,y)}{1 - L_D(x,y) + \lambda} \tag{II.4}$$

In the above formula λ represents a normalization term used to eliminate the singularity in the above equation. It is important to notice that several similar algorithms reported in literature [Huo Y., Yang F., Brost V., 2013], [Banterle F., Ledda P., Debattista K., Chalmers A., 2008] calculate the average Luma value over all the input image before proceeding to the

LDR2HDR conversion; however, the main purpose of the proposed design is to perform an on-the-fly elaboration of the input frame, in order to achieve hard real-time performances. Due to that and to the unavailability of obtaining the entire frame to process at the beginning of the computation, we decided to calculate the moving average online during the process, continuously updating the value at each incoming pixel, following the raster scan order of the incoming pixels, which could be acquired by a memory device or directly by a sensor.

After that step we implement in HW a process similar to the dodging and burning technique in photographic processing

$$L_H(x,y) = L_M(x,y)\frac{L_D(x,y)}{L_R(x,y)} \tag{II.5}$$

where the L_R term in the second factor of the above equation is calculated as

$$\begin{cases} L_R(x,y) = \dfrac{1}{k(x,y)} \displaystyle\sum_{m=-M_x}^{M_x} \sum_{l=-M_y}^{M_y} \left[G_2(m,l,\sigma_2) G_1(\Delta L_D(m,l),\sigma_1) L_D(x-m,y-l) \right] \\ \Delta L_D(m,l) = L_D(x-m,y-l) - L_D(x,y) \end{cases} \tag{II.6}$$

In the equation (II.6) L_R is calculated using an edge bilateral filter as shown in [Durand F., Dorsey J., 2002]. Furthermore, the other functions involved in equation (II.6), G_1 and G_2, are a one-dimensional (1D) gaussian function, G_1, and a 2D gaussian function, G_2, defining also the $k(x,y)$ are defined as

$$\begin{cases} k(x,y) = \displaystyle\sum_{m=-M_x}^{M_x} \sum_{l=-M_y}^{M_y} \left[G_2(m,l,\sigma_2) G_1(\Delta L_D(m,l),\sigma_1) \right] \\ G_1(\Delta L_D,\sigma_1) = A_1 e^{-\frac{\Delta L_D^2}{2\sigma_1^2}} \\ G_2(m,l,\sigma_2) = A_2 e^{-\frac{m^2+l^2}{2\sigma_2^2}} \end{cases} \tag{II.7}$$

where A_1 and A_2 are real constants, l and m are relative integer numbers, $\{l,m\} \in \mathbb{Z}$ and M_x and M_y are the horizontal and vertical dimensions of the filter.

The above equations have to be simplified to obtain a HW friendly design, since a straightforward implementation would require a huge amount of MAC units. A piece-wise approximation of the above equations for the bilateral filter is provided in [Durand F., Dorsey J., 2002] in the frequency domain. On the contrary, the proposed design focuses on a spatial filtering technique in order to reduce the latency related to the transformation and antitransformation operarations, while achieving lower area requirements. The proposed modification exploits the separability of the multidimensional gaussian kernels, allowing us to write

$$L_R(x,y) = \frac{A_2}{k(x,y)} \sum_{m=-M}^{M} e^{-\frac{m^2}{2\sigma_2^2}} F(x,y,m) \tag{II.8}$$

and

$$F(x,y,m) = \sum_{l=-M}^{M} e^{-\frac{l^2}{2\sigma_2^2}} L_D(x-m,y-l) G_1(\Delta L_D(m,l),\sigma_1) \tag{II.9}$$

These simplifications holds for the $k(x,y)$ equation in (II.7). Exploiting these simplifications it is possible to achieve the wanted computational complexity reduction of the previously reported equations. The complexity of the overall operation passes from $O(M^2)$ to $O(M)$ due to the separability of the gaussian kernels; the tile processing is divided in two smaller elaborations, first over the rows and then over the columns. This will cause the implementation of an additional structure, as described in the following, but results in a general improvement of the proposed design, nevertheless.

II.3 Numerical Considerations

A gaussian function is defined on an infinite domain, while on the contrary a gaussian digital filter is of course defined using a finite number of values. It is possible to obtain a good approximation of a gaussian filter in HW considering only the first 3σ of the gaussian as an approximation of the real function and truncating the rest of it, considering the standard deviation of the gaussian function, σ. Considering 3σ of the 1D gaussian function it is possible to take into account approximately the 99.73%, meaning that only the 0.27% of the space spanned by the original gaussian is not taken into account. This choice allows to state that the excluded values are two order of magnitudes smaller than the median value of the filter, guaranteeing a good accuracy in the developed computations. It is important to highlight that different approximations could be made in order to obtain different speed/area trade-offs, using lower M values, hence sacrificing the accuracy of the design.

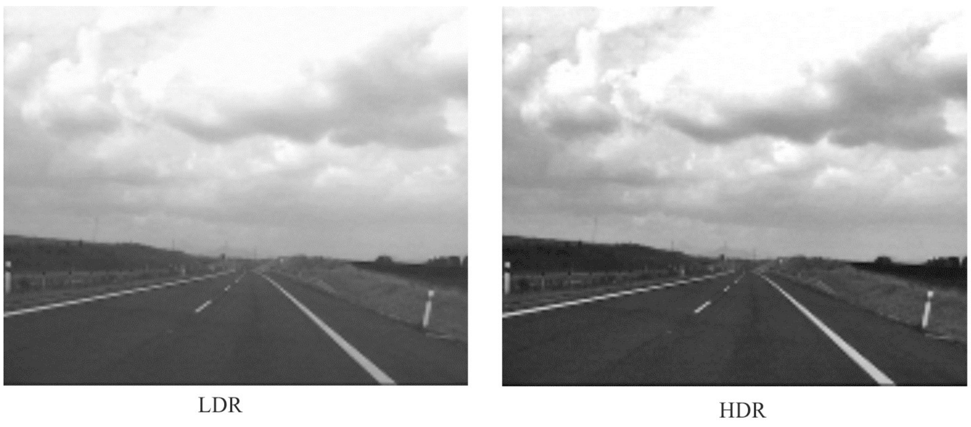
Wanting to develop a comparison with the work in [Durand F., Dorsey J., 2002], we established the values of σ_1 and σ_2 according to that. In particular $\sigma_1=16$ and $\sigma_2=3$ were considered, obtaining M=48 and thus considering a

Table II.1 *Algorithm performances compared with [Huo Y., Yang F., Brost V., 2013].*

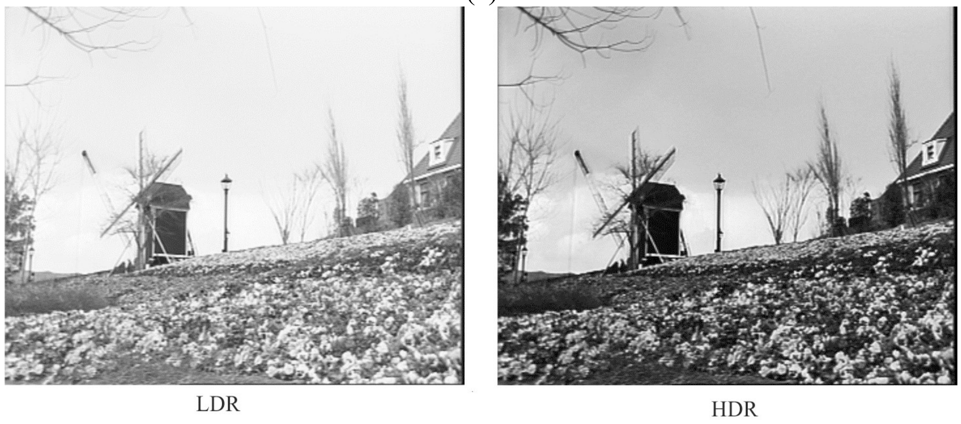
Images	Dynamic range [a.u.] Ref. [Huo,2013]	Dynamic range [a.u.] Current Work	PSNR [dB]	SSIM
bridge_close	$1.13\text{x}10^4$	$1.13\text{x}10^4$	24.6	97.8%
bridge_far	$2.41\text{x}10^3$	$2.05\text{x}10^3$	26.0	96.6%
city	$5.92\text{x}10^4$	$6.49\text{x}10^4$	22.0	94.3%
football	$3.64\text{x}10^5$	$3.63\text{x}10^5$	49.2	99.9%
garden	$2.05\text{x}10^7$	$2.12\text{x}10^7$	35.5	97.9%
highway	$1.89\text{x}10^3$	$1.58\text{x}10^3$	21.0	94.0%
mill	$1.85\text{x}10^6$	$1.86\text{x}10^6$	36.4	98.0%
Prato	$1.69\text{x}10^7$	$1.76\text{x}10^7$	41.9	96.5%
tulips	$5.41\text{x}10^2$	$5.49\text{x}10^2$	25.9	90.8%

kernel having dimensions $(2M+1)$ $(2M+1)=97\times97$ pixels. A comparison between the considered method and the one reported in [Huo Y., Yang F., Brost V., 2013] is reported in Table II.1, in terms of Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM). The set has been chosen because the considered images has different exposure levels, making the set a good one to test for the various lighting conditions and gradients and make the overall observation independent from the particular exposure conditions. From the results in Table II.2 is possible to notice that the differences in dynamic range between the two implementations are almost negligible, while also the PSNR and SSIM values provide good results. In particular, SSIM shows an average value of 96.2%, proving the good accuracy of the proposed method, despite of the approximations performed to achieve a HW friendly design, capable of meeting acceptable area and timing requirements. It is also possible to further notice that the PSNR values are lower for frames having values of Luma decreasing from the top to the bottom of the frame (e.g. "highway" frame), while an opposite result could be highlighted in the images having Luma values increasing from top to bottom (e.g. "football" or "Prato" frames). Examples of the proposed elaboration are shown in Figure II.2 for the "highway" and "mill" frames.

Then we conducted several tests to correctly choose the values of the couple (σ_1, σ_2). In these tests σ_1 values in the range [0.1; 0.5] and values in the range [4; 32] were considered. A new metric was defined, in order to obtain numerical information on the goodness of the proposed method and provide a fair comparison between the various couples of values to be tested. The new defined metric is derived from three parameters indicated as P_{loss}, P_{ret} and P_{inv}, which are the probability of contrast loss, retention and



(a)



(b)

Figure II.2 *Application of the proposed expansion method to (a) "highway" (PSNR = 21.0 dB) and (b) "mill" (PSNR = 36.4 dB) frames.*

inversion, respectively, calculated comparing the HDR final result to the LDR original frame. The chosen score function is so defined

$$S = P_{ret} - P_{loss} - 2P_{inv} \qquad \text{(II.10)}$$

where the factor 2 for the term P_{inv} is related to the higher severity of that kind of error. The results of the conducted tests are reported in Table II.2. From the reported data it is possible to state that the best score function is obtained using $(\sigma_1, \sigma_2)=(0.2, 32)$; nevertheless, the implementation of such a design would require the partial storage of at least 32 rows of data, which would be far too much in the case of high resolution frames, and will also cause a major increase in the number of arithmetic units, resulting in unacceptable performances both in terms of area occupation and power consumption. Due to these considerations, we choose the set of parameters

Table II.2 *Average quality metrics on the considered image set.*

σ_1	σ_2	P_{loss}	P_{ret}	P_{inv}	$S=P_{ret}- P_{loss}-2P_{inv}$
0.1	4	0.0252	0.0453	0.044	0.0112
	8	0.0247	0.0463	0.0045	0.0126
	16	0.0244	0.0468	0.0045	0.0134
	32	0.0243	0.0469	0.0045	0.0136
0.2	4	0.0257	0.0482	0.0050	0.0125
	8	0.0253	0.0501	0.0051	0.0147
	16	0.0247	0.0514	0.0050	0.0167
	32	0.0241	0.0524	0.0050	0.0184
0.3	4	0.0269	0.0488	0.0056	0.0106
	8	0.0265	0.0509	0.0057	0.0130
	16	0.0260	0.0524	0.0056	0.0151
	32	0.0247	0.0493	0.0054	0.0137
0.4	4	0.0282	0.0479	0.0062	0.0073
	8	0.0279	0.0504	0.0063	0.0100
	16	0.0273	0.0522	0.0062	0.0125
	32	0.0261	0.0538	0.0060	0.0158
0.5	4	0.0290	0.0474	0.0066	0.0051
	8	0.0287	0.0499	0.0067	0.0077
	16	0.0283	0.0517	0.0065	0.0104
	32	0.0269	0.0537	0.0063	0.0142

$(\sigma_1, \sigma_2)=(0.2, 4)$, in order to obtain a better trade-off in ADP performances, truncating the gaussian monolateral dimension to $M_x=3\sigma_2$ and $M_y=\sigma_2$, choosing to deal with an asymmetric gaussian filter of smaller dimensions. The dimensions of the considered filter are then 25×9.

The elaboration is conducted on the Luma component of the LDR frame coded in Y'UV. A careful study to reduce the number of bits used to code the partial results and the final HDR Luma component has been conducted. First it has to be notice that all the quantities involved in the computation are non negative since the input LDR pixel is represented only using positive values in an integer range [0:255], while the coefficients of the gaussian kernels are always represented by real positive numbers. From these considerations it is possible to state that the sign controls on the intermediate results could be avoided, while it could be juxtaposed in the final FP standard compliant result. The original Luma LDR component is coded using 8 bits per pixel, while for the final coding we choose FP32 standard. Another consideration was developed starting from the extreme values of the terms in the above equations; in particular, we evaluated the minimum values of the quantities $\sum_{m=-M}^{M} e^{-\frac{m^2}{2\sigma_2^2}} F(x,y,m)$ and $k(x,y)$, which are $6.4\cdot10^{-6}$ and 4.2, respectively, while their maximum values are $1.9\cdot10^{4}$ and 74.3. The final range of the exponential part of the FP number is defined by these amounts and it is possible to state that the range to be spanned by the exponent is [-17; 14], meaning that it is possible to code the exponent using 6 bits, spanning a range [-31; 32] and saving two bits with respect with the standard 8 bits exponent defined in IEEE-754 standard. These simplifications lead in turn to a great reduction of the overall datapath involved in the calculation of the L_R factor and overall reduce the amount of memory to be used for the storage of the intermediate results.

II.4 Architectural Considerations

The proposed design provides a HW friendly implementation of an iTM algorithm made up by:

- an incremental average calculation unit for the Luma components;
- a bilateral filter based on partial serialization and stripe buffers;
- small buffers dedicated to the storage of Chroma components.

The overall detailed structure is presented in Figure II.3, in which it is possible to highlight two main structures which develop the main operations:

- a module devoted to the calculation of the L_R via the computation of its numerator and denominator, N_R and D_R;
- a module devoted to the computation of L_M and L_H starting from the partial results of the L_R module and the module computing the moving average.

The input pixels are acquired in raster scan mode, while the developed real-time architecture allows us to avoid the use of buffer structures or caching apparatus external to the acquisition device. After each pixel arrives to the processing unit in Y'UV uncompressed format, it is possible to separate the Luma component from the Chroma ones, to provide the former to the elaboration structure and the latter to a Chroma dedicated synchronization buffer, needed to construct the final HDR image. Then the Luma components are processed to derive the incremental average L_a^{inc} and the L_M and L_H terms to finally compute the expanded HDR Luma result.

Figure II.4 schematizes the operation principle of the filtering stage; the computation of $F(x,y,m)$ starts after the acquisition of the first $(2M_x+1)$ pixels from the acquisition device, to store them into a stripe buffer structure of dimensions $W\cdot(2M_Y+1)$, being W the width of the considered frame, while $(2M_x+1)$ and $(2M_Y+1)$ are the horizontal and vertical dimensions of the considered filter. It is worth to underline that the stripe buffer is needed to store the partial results before sending them to the vertical filter.

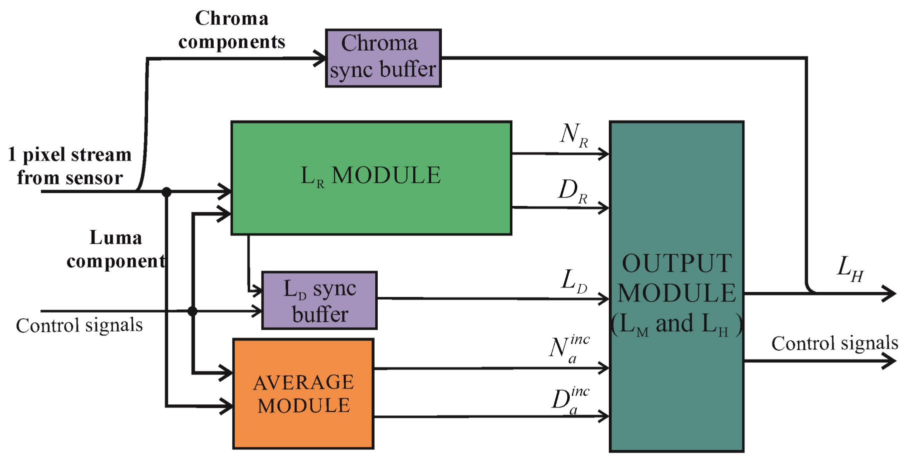


Figure II.3 *General scheme of the proposed.*

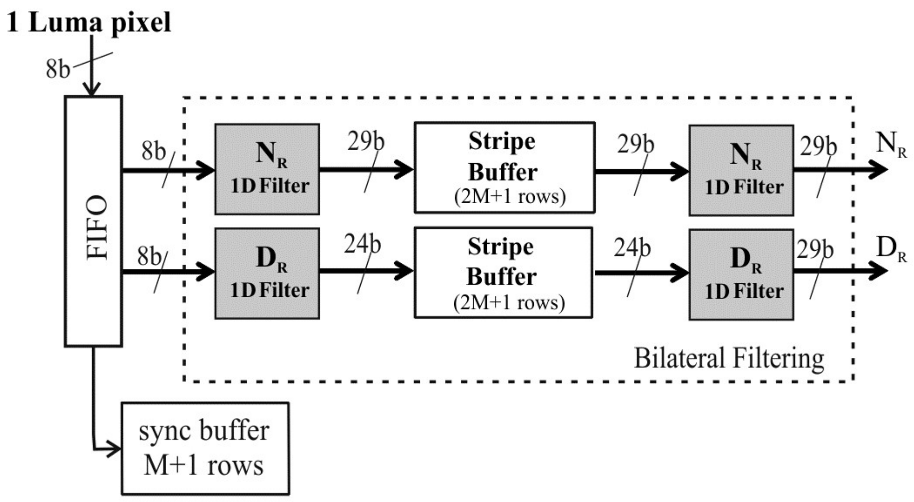


Figure II.4 *Detailed scheme of the 1st stage, showing the sub-modules for the bilateral filtering calculation.*

Finally a further elaboration unit is needed to calculate and update the $L_a^{inc}(x,y)$ value at each clock cycle before the elaboration takes place, in order to obtain a on-the-fly processing and the unavailability of the average Luma of the frame in the beginning of the computation. The operation to compute is shown in (II.11).

$$L_H(x,y)=\left[\frac{1}{L_a^{inc}(x,y)}\frac{L_D(x,y)}{1-L_D(x,y)+\lambda}\right]\times\frac{L_D(x,y)}{L_R(x,y)} \qquad \text{(II.11)}$$

II.4.1 L_R and Output Modules

In principle the filtering stages for the calculation of N_R and D_R changes only for the coefficients used, meaning that the project of the filtering stage is the same for both the structures.

The L_R module exploits the 2D filtering operation and is composed by two 1D filters, in order to exploit the separability property of the gaussian kernels, with a stripe buffer between to separate the 1D filtering operations. It must be noticed that to carry out the elaboration related to the vertical filter the stripe buffer has to be sequentially accessed column-wise. Moreover, assuming $M_X = M_Y = 3\sigma_2$ and that $\sigma = 4$, each 1D filter involves 25 pixels. A parallel straightforward implementation on all the pixels would require the implementation of 100 FP multipliers and 98 FP adders, working on 29 bits data. This justifies the use of $M_Y = \sigma_2$, allowing to reduce the number of FP units to 68 multipliers and 64 adders and to contain the dimensions of the stripe buffer, while still representing a good approximation compared to the previously considered choice of parameters.

Finally, after these computations, the Output module computes the quantities $L_D^2 \times D_R \times D_a^{inc}$ and $(1-L_D+\lambda)\times N_R \times N_a^{inc}$ and their ratio to obtain

$$\frac{L_D^2\times D_R\times D_a^{inc}}{(1-L_D+\lambda)\times N_R\times N_a^{inc}}=\frac{1}{L_a^{inc}}\times\frac{L_D}{1-L_D+\lambda}\times\frac{L_D}{L_R} \qquad \text{(II.12)}$$

Finally, the output of the computation of equation (II.12) is converted from our custom FP29 format to the standard IEEE-754 FP32 one, in order to obtain data to be used for further computations without the need of preliminary conversions.

II.4.2 Stripe Buffer and Multi-Resolution Implementation

The general design of the stripe buffer and its operation principle are depicted in Figure II.5. The principle of work of the structure follows a SIPO

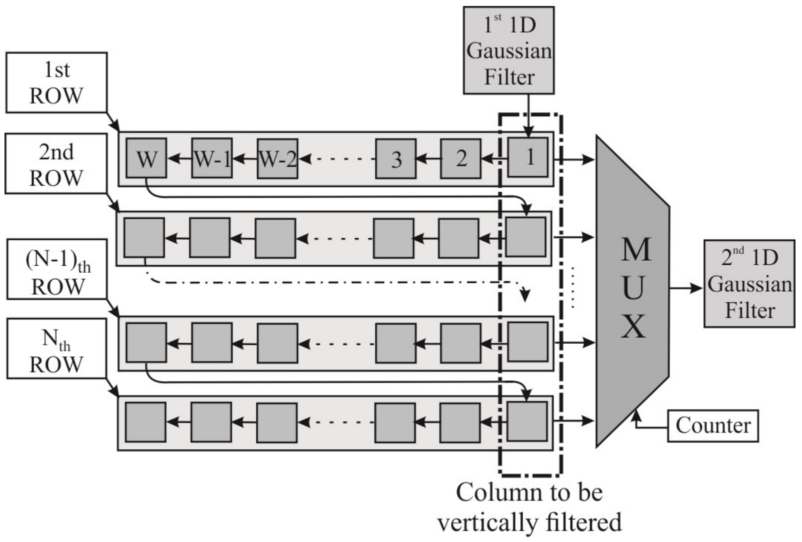


Figure II.5 *Operation principle of the stripe-buffer.*

architecture, storing the incoming partial results, one per clock cycle, and providing as an exit $2M_Y+1$ data to the vertical filtering structure. The SIPO is capable of providing a continuous alignment of the stored data, making it possible to process $2M_Y+1$ per cycle.

Implementing the SIPO using only flip flops (FFs), but it is not convenient both in terms of area and power performances. In fact, such an implementation would require the implementation of a high number of FF units. Due to this, we decided to implement the buffering structures using SRAMs which behavior emulates the SIPO's one. In particular, the availability of hard macros to implement dual-port SRAMs in FPGA made it possible an easy implementation of these structures. The SRAMs implement long shift register capable of providing the output data while being written with a new one. Moreover, we need a number of such structures equal to the dimension of the vertical filter; this could be one implementing each row of the stripe-buffer using a dual-port Block Random Access Memory (BRAM) of appropriate dimensions. Moreover, this kind of approach greatly simplifies the extension of the proposed design in the case of the elaboration of frames having different resolutions. In fact, while the architecture presented in [G. D. Licciardo, A. D'Arienzo, A. Rubino, 2015] was tailored for the processing of Full-HD frames only, it is possible to develop a design working on frames having different resolutions by projecting a unit capable of managing the various First In First Out (FIFO) structures composing the stripe buffer. The width of the image to be processed is used as a control signal by the FIFO structures to obtain a correct storage of the partial data, while two different behavior have to be taken into account depending on if the width of the image to be processed exceeds the dimension of the FIFO rows, *W*, or not.

In fact, it is possible to state that:

- in the case the dimension of the incoming frame is lower than *W*=1920 or equal to it, the memorization of new data in the FIFO structures is interrupted when after the storage of *W* data per row, while the architecture follows the same behavior described in [G. D. Licciardo, A. D'Arienzo, A. Rubino, 2015];
- if the incoming frame has a width higher than *W*, up to 3840, two consecutive FIFO structures are used as a single FIFO to store a single row of partial data of the input pixels. Obviously, in this case only five filtered data per cycle could be transferred to the vertical filter. The correct transfer of the data is achieved using control signals and multiplexer units, which makes it possible to choose which data have to be sent to the second filtering module.

It is then possible to state that in the former case the image filtering will have parameters $M_X = 3\sigma_2$ and $M_Y = \sigma_2$, while in the case of the elaboration of 4K UHDTV image we will have $M_Y = 0.5\sigma_2$. A detail of the proposed technique is sketched in Figure II.6, where the multiplexer structures used to achieve the multiresolution operations are clearly shown.

As it is possible to notice, the implemented method to achieve multiresolution processing does not require big amount of logic structures, while ensuring the required behavior with only a little worsening in the time performances, as proven in the following. However, it shows an increase in terms of arearequired on the FPGA board by the design in terms of occupied LUTs.

II.5 Implementation Results

The developed architecture has been implemented on axc7v2000tflg1925-1 FPGA board, of the family Xilinx Virtex 7, while the Xilinx Vivado software has been used for the synthesis and the verification of the proposed design, and in TSMC CMOS 90nm std_cells. Table II.3 reports the results of the implementation and compare them to the one presented in [G. D. Licciardo, A. D'Arienzo, A. Rubino, 2015].

Aiming to obtain an implementation as most independent as possible from the FPGA board and to have indications for future ASIC implementations of the proposed design, we decided to consider implementations not exploiting the use of DSPs. It is possible to notice how the new implementation shows an increase in terms of area occupation if compared to the design in [G. D. Licciardo, A. D'Arienzo, A. Rubino, 2015]. In fact, the logic units introduced to achieve multiresolution operations

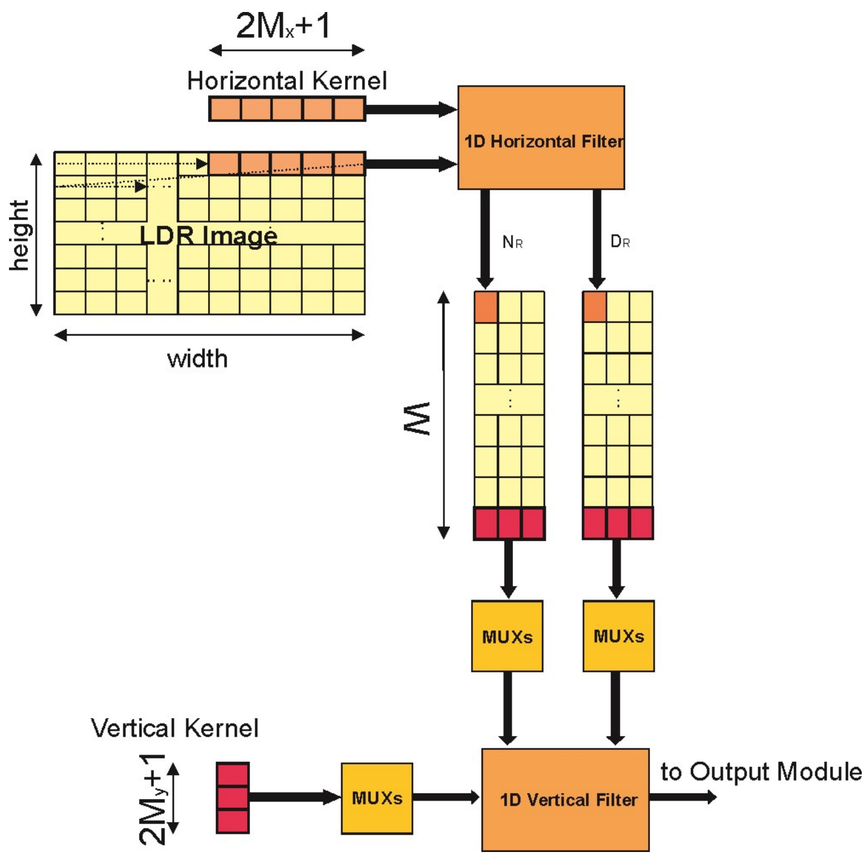


Figure II.6 *Composition of the stripe-buffer with MUX structures for multi-resolution.*

increase the number of LUTs of 7.33%. Also the maximum operating frequency is reduced if compared to the single resolution architecture case, since the arrival time critical path increases of 13.22%, making the maximum operating frequency 60.4 MHz, allowing processing approximately 60 Mpixels per second. The FP multiplier units used in the vertical filtering structure compose the critical path. The structure is then capable of processing a Full-HD frame in 34.4 ms, which means that it can elaborate 29 fps, achieving in this case and in the case of lower resolutions real-time processing. Instead, in the case of 4K UHDTV resolution the processing time of a single frame is 139 ms, which allows the architecture to process only 7 fps, not achieving in this case real-time performances. However, the proposed design shows far greater operating frequencies than SW implementations. In fact, the same algorithm, implemented in MATLAB ver. 2012b and running on a 4 cores/8 threads Intel Xeon at 3.2 GHz, takes approximately 8 hours to process a single VGA (640x480 pixels) frame. Regarding the std_cell implementation, it is worth to notice that the superior flexibility of the design cause an increase of 0.1 mm^2 (0.13%) in terms of occupied area and an overall increase of approximately 12.5% of the worst path delay, causing in turn a little decrease of the fps that can be processed.

It is worth to notice that it would be possible to develop further operations between on FP32 data and fixed data sets in a more HW friendly fashion, using multiplier units exploiting decomposition methods, such the ones used in Distributed Arithmetic (DA), like Bachet's one previously described, in order to obtain a more compact and less power demanding implementation. Furthermore, even in this case the transferring of data to and from the memory structures is one of the largest contribution to the data path delay, making a careful study on the memory structures needed in order to develop ad hoc, faster and more reliable storage units.

TABLE II.3 *Synthesis results of the proposed design.*

	FPGA		std_cell	
	[Licciardo et al.]	Proposed	[Licciardo et al.]	Proposed
LUTs/Area [mm^2]	51.3 k	55.06 k	7.7	7.8
BRAMs	58	58	-	-
Worst Path Delay [ns]	14.632	16.567	7.092	7.978
Full-HD fps	31	29	67	60

II.6 Conclusions

A new HW architecture to process frames having different resolutions acquired from an image sensor and elaborate them to obtain their HDR version, starting from a single LDR frame. The structure results more flexible than the one presented in [G. D. Licciardo, A. D'Arienzo, A. Rubino, 2015] and achieves state-of-the-art performances, obtaining a structure capable of working with multiresolution image sensors.

The main features of the developed design are:

- Streaming elaboration on input data received from image sensors, for on-the-fly conversion of input images;
- Absence of frame buffers to store the input and the partially elaborated image data;
- The possibility to completely substitute off-the-chip DRAM by a reduced amount of embedded SRAM, which enables the implementation of embedded application-specific image processor (ASIP) tightly coupled with image sensors;

- Scalable architecture that allows setting the area/speed ratio, to favor a platform independent implementation from FPLs to ASIC std_cells.

It is worth to underline that the proposed design achieves the above features by exploiting the separability property of the two-dimensional Gaussian kernel, which is largely employed in image and video elaboration methods; therefore, regardless of the specific application, similar characteristics can be reproduced in a large number of different ASIPs, demanding for high integration level and real-time elaboration, as happens for example, in difference-of-Gaussian-based visual search algorithms.

Further studies could be related to the development of a full-custom project of an integrated circuit (IC) to obtain an ASIP for HDR2LDR conversion, capable of achieving higher operating frequencies, lower power dissipations and real-time performances even in the case of 4K UHDTV image processing.

Finally, an example of final elaboration of a HDR obtained image is reported in Figure 7 for the frame "city", complete of the Chroma components. Obviously, all the reported images have been transformed back using a Tone Mapping Operator (TMO) in order to show them on standard HDR screens and on paper, which is the main cause for the darker areas in the derived image, due to the application of the dodging and burning like step in the HDR elaboration. We further underline how this LDR reconstructed image using TMO is just to give an idea of the contrast enhancement using HDR, but it is of course not possible reproducing the effects of the derived HDR result on LDR screens.

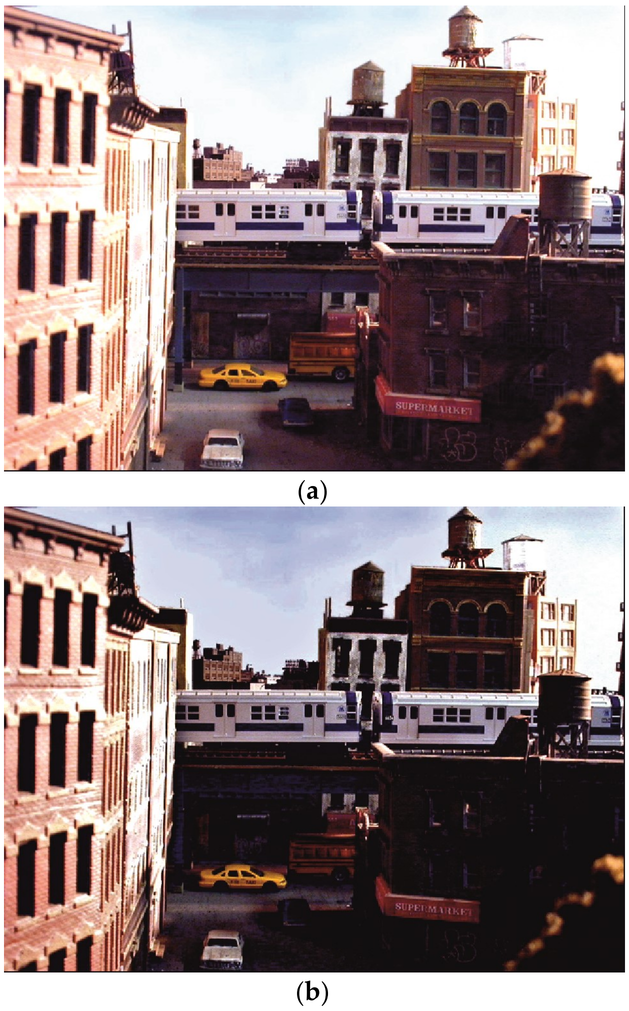


Figure II.7 *Example of LDR to HDR conversion (a) the original LDR frame; (b) the HDR frame obtained through iTM.*

Chapter III
Voltage Sense Amplifiers circuits for SRAMs

III.1 Introduction

In the previously developed designs we saw how the times of response to extract data from the memory and to write data on it. That is why, during the years part of the research has focused around finding new solutions towards achieving better performances memories. During these studies the research group developed a new sensing scheme for SRAMs, in order to obtain better performances in terms of area occupation, power consumption and reliability, obtaining, as will be shown next, a better offset rejection. The present work is supported in part by Europractice Program.

More and more demanding PPA constrains for SRAM memories and the increasing growing storing densities tend to require reduced voltage variations that a single memory cell can induce on the attached bit-line while still obtaining an acceptable accessing time. This trend is in contrast with the increase of the offset voltage, VOS, generated by the mismatch of the components of the sensing circuitry and incremented by the MOSFET scaling. In addition to that, the need for less power demanding architectures, leads to the lowering of the bias voltages, which further worsens the offset related problems and could unacceptably compromise the memory operations. In order to overcome this issues, during the last years, improved sensing circuits have been proposed, using offset compensation and cancellation techniques, capable of mitigating or eliminating the offset in a certain measure [A.J. Bhavnagarwala, X. Tang., J.D. Mandl, 2001], [S.R. Nassif, 2001].

III.2 Background

Semiconductor memories represent one of the main components in nowadays processing units. In fact, from their improvement it is possible to achieve huge benefits for the overall system. The types of developed Random Access Memories (RAMs) are principally of two types: Static Random Access Memories (SRAMs) and Dynamic Random Access Memories (DRAMs). A RAM memory chip is made up memory cells organized in the form of a matrix; that’s why usually we refer to RAM components also specifying the wordlength and the depth of the memory, in order to give an indication of the organization of the memory cells. That’s why the row of the matrix are referred to as word lines (WL), while its

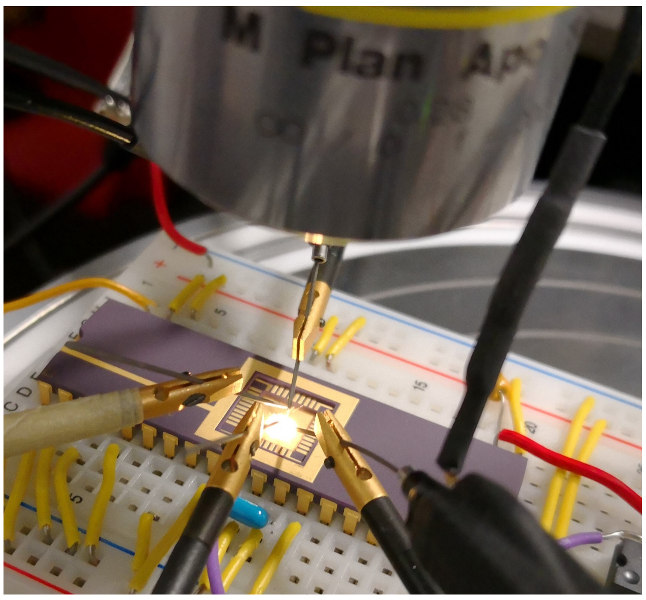


Figure III.1 *Breadboard measurement setup for the developed SRAM offset compensation design.*

columns are also referred to as bit lines (BL), since once the WL is chosen every BL carries only one bit of information. Hence, to access the memory a row address and a column address are needed. Every access operation is carried out in three steps:

- row decoding to activate the word line;
- column decoding to activate the read/write (RW) circuit to access the single memory cell;
- the reading or writing operation itself.

It is worth to underline that in the following we will always refer to circuits in the Complementary Metal-Oxide Semiconductor (CMOS) technology, since they usually represent the ones of interest almost the totality of nowadays applications and for the ones presented in this thesis work.

III.2.1 SRAMs Overview

The static definition of SRAMs is due to the fact that the bits memorized in the memory cells is stored and does not need to be restored until the chip is connected to the power supply. This property is derived from the behavior of the fundamental component of a SRAM, the latch. As it is well-known, the latch shows three possible working points, of which two are stable ones, while the third could be easily perturbed and end up in one of the other two, depending on the entity of the perturbing signal. This behavior is due to the

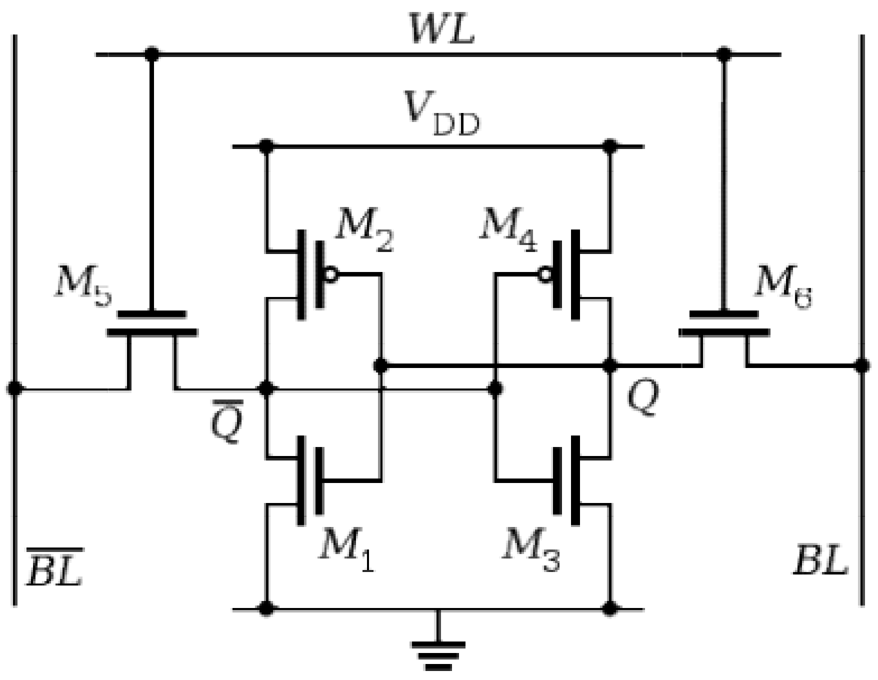


Figure III.2 *6T CMOS SRAM cell [https://en.wikipedia.org/wiki/Static_random-access_memory#/media/File:SRAM_Cell_(6_Transistors).svg].*

positive feedback provided by the inverting amplifiers loop in the latch structure. SRAMs are particularly needed in high speed applications, due to the latch behavior and its capability of fast restoring the data (e.g. cache memories); in fact, the positive feedback guarantees extremely fast commutations during writing operations, while there is no need to restore the information after any reading operation, since the data is not erased during the operation. These performances come at the cost of higher power losses, due to the higher currents needed to charge the parasitic capacitive elements of the reading lines. A 6 transistors (6T) CMOS SRAM cell is shown in Figure III.2.

III.2.2 DRAMs Overview

DRAMs invention is due to the need of obtaining higher memory densities capable of storing higher amounts of data. This is achieved storing the information in a capacitive element. This is done mainly through a reduction of the number of transistor used to build a single cell; the first DRAMs were made up by 3 transistors (3T) while right now the most used design is the one using just 1 transistor (1T), for which a matrix of cells is shown in Figure III.3. The structure of a single cell is almost trivial, since it consists of just a CMOS and a capacitor which charge and discharge are driven by the CMOS. Moreover, in all DRAMs it is necessary to perform a

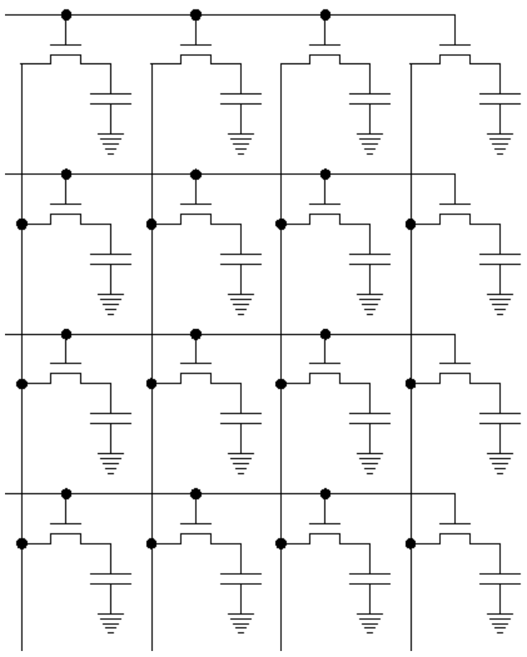


Figure III.3 *1T CMOS DRAM cell matrix.*

periodic refresh operation of the stored data, in order to retain them, since capacitances and the transistors always shows not negligible loss currents, capable of corrupting the data. The refresh operation consists in reading the data and rewriting it in the memory cell; of course, this operation has to be repeated periodically on all the memory cells constituting the memory.

However, both SRAMs and DRAMs need a sensing circuit to correctly perform the RW operations. The developed work consists in the project of a new sense amplifier (SA) which is capable of improving most of the SA performances and paves the way to the development of new memory chips for high performances.

III.3 Sense Amplifiers

As the name suggests, SAs are amplifiers capable of sensing differential voltages or currents and amplify them, in order to obtain the voltage levels which generated them as outputs once again. It is important to highlight that in order to obtain a memory circuit with enough memory density, it is fundamental to have SAs capable of working on high number of cells, allowing to instantiate less SAs while obtaining the same performances in

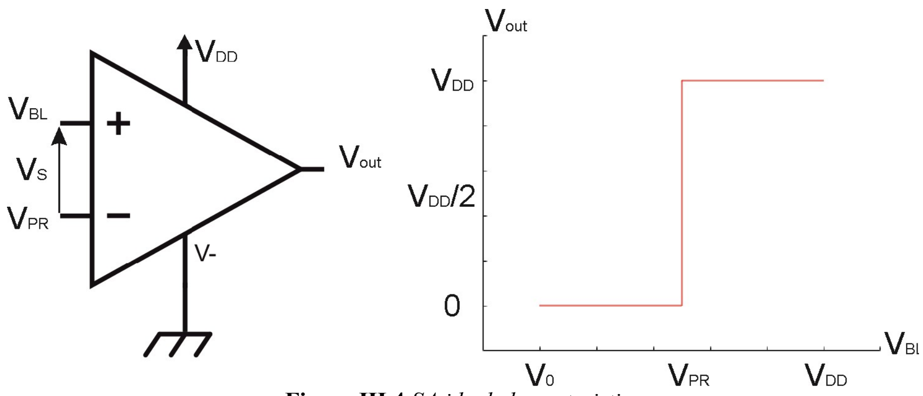


Figure III.4 *SA ideal characteristic.*

terms of correctness and speed of RW operations. Let us examine the case of a voltage sensing amplifier.

Read Operation: Before a WL could be addressed via a row decoder, all the matrix BLs are charged to a fixed voltage known as pre-charge voltage, V_{PR}, using a pre-charge net. Then, the SAs are connected, one for each BL, to the BL itself and to a reference voltage equal to V_{PR}. After the pre-charge phase it is possible to address the chosen WL imposing to it the logic high voltage level to switch on the transistors to access the memory cells of the chosen row, the voltage of the cell is read and produce a variation in the order of mV from the V_{PR} value in output, causing, in turn, the SA to detect the difference between its two terminals and then amplify this difference to obtain the reconstructed original logic level (0 or V_{DD}) of the cell as output. In the end, using the Column decoder, only the output of one amplifier is selected and taken as output, corresponding to the particular cell (or group of cells) to be read.

Write Operation: the write operation follows the read one, with the exception that the column decoder imposes the output value of the amplifiers using low impedance circuits, which is then, send as input to the cell using a simple switch, while the columns on which the data is not imposed retain the previous value.

III.4 Offset Issues

As seen before, the differential signal coming from the accessing of a single memory cell is small, which represents the reason sense amplifying circuits are developed. Ideally, the input/output (I/O) curve of the SA is perfectly symmetrical; in other words, it gives an output half the logic swing when a null voltage is applied as input, while it gives the high logic value and the low logic value for positive or negative differences between the signal coming from the memory cell and the reference one, respectively.

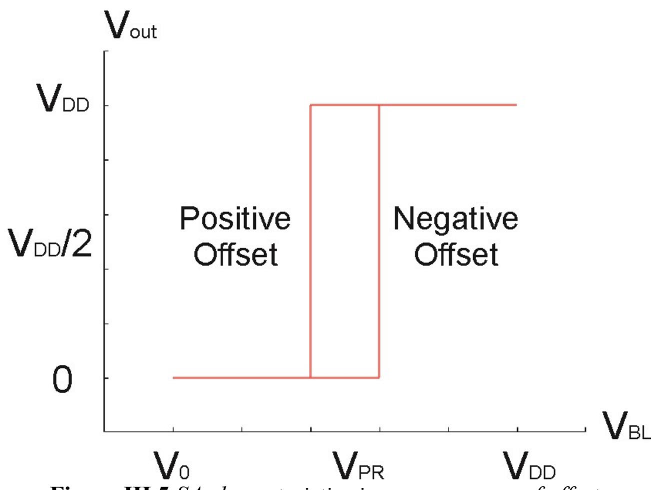


Figure III.5 *SA characteristics in presence case of offset.*

Real life scenarios are however a lot different from that, since the characteristics of the real devices could change due to technological processed tolerances. E.g. the channel length of a MOS device, the thickness of the oxide depositions, the geometry of source, drain and gate regions and the doping profiles could be different from their nominal value in a certain percentage. Due to that, devices having nominally the same performances could show work slightly differently in reality. This is a well-known issue, directly related to the inevitable processes tolerances, but in the particular case of memory SAs makes them to have an input offset voltage, which entity could not be a priori known. The previous scheme could then be represented adding a further voltage on the non inverting terminal and the aim of the SAs has also to compensate for the offset, which effect is shown in Figure III.5 both for a positive and a negative offset. The effect of the offset could then be summarized as follows: when the signal on the BL results smaller in module then the offset of the SA, then a mistake read operation takes place. It is also to be noticed that even if the sense signal is bigger in module than the offset, it still reduces the differential input to the SA, decreasing the speed in the data reading and hence the overall performances of the memory chip.

III.5 Latch CMOS

The CMOS latch is the fundamental circuit used in RW operations on memory cells, due to the amplification it provides to the small differences provided at its input terminals. A CMOS latch is basically composed by a loop of two CMOS inverters, as shown in Figure III.6.

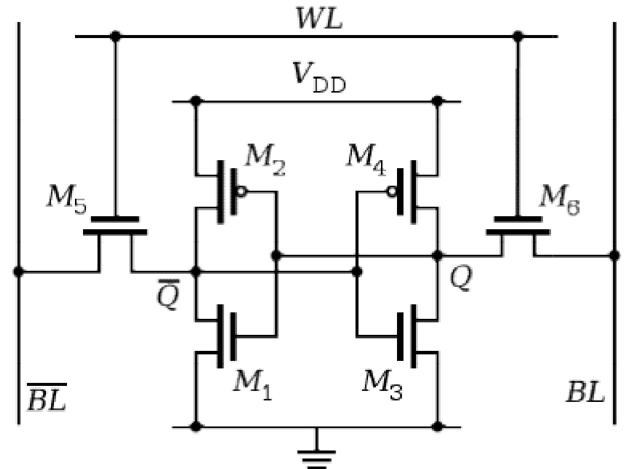


Figure III.6 *Scheme of a CMOS latch.*

More in detail, the inverters are realized in Full Complementary Metal-Oxide Semiconductor (FCMOS) technology, meaning Positive Metal-Oxide Semiconductor (PMOS) are realized using a form factor given by following formula

$$\frac{W_p / L_p}{W_n / L_n} \approx \frac{\mu_n}{\mu_p} \tag{III.1}$$

where W_p represents the width of the width of the PMOS device and L_p represents the length of its channel, while μ_p is the mobility of the holes in the device; the terms reporting the n subscripts are related to the Negative Metal-Oxide Semiconductor (NMOS) counterpart. NMOS devices are usually projected using the minimum allowable sizes, in order to obtain higher densities, while the use of PMOS devices results in an increase if compared to NMOS, mainly due to the lower mobility of holes than electrons. However, this disadvantage is highly compensated by the fact that a FCMOS process, involving also PMOS devices, could achieve an ideally perfectly symmetric I/O curve and an increase in terms of speed due to the fact that the PMOS devices conductivity is in this case equal to the one of NMOS. The ideal characteristic of a FCMOS inverter is reported in Figure III.7, considering the skews and the non instantaneous response of the devices.

As stated before, the latch circuit has got three equilibrium points, of which two are stable and one is unstable. The mathematical formulation for the three states could be written as

$$\begin{cases} V_I = V_{DD} \\ V_O = 0 \end{cases}, \qquad \begin{cases} V_I = 0 \\ V_O = V_{DD} \end{cases}, \qquad V_I = V_O = V_{LT} = V_{DD}/2 \tag{III.2}$$

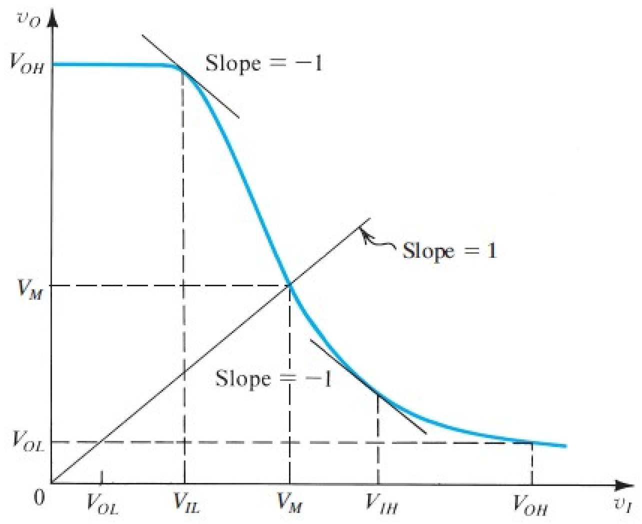


Figure III.7 *Ideal characteristic of a CMOS latch.*

The third state is obviously not stable, since the closed loop gain is equal to the product of the gains of the two inverters on the threshold point, meaning that small variations in either of the two voltages could lead the equilibrium point to move fast towards one of the other stability points. In fact, the high gain produces a fast instauration of a new stable scenario, as evident by the following equations

$$\begin{cases} A_1 << -1 \\ A_2 << -1 \\ A_1 A_2 >> 1 \end{cases} \tag{III.3}$$

From what reported it is clear that, if the voltage at node Q varies for a certain quantity ΔV, the voltage at $\bar{Q}$ varies as $V_{LT} + A_1 \Delta V$, obtaining then on Q, $V_{LT} + A_1 A_2 \Delta V$ and so on until the system does not reach the stability. It is important to notice that the starting voltage difference ΔV could be arbitrary small and still capable of create a positive loop, reaching a stable state. Same approach can be applied for voltage differences at node $\bar{Q}$ obtaining the same results.

III.6 Offset Compensation Schemes

In the last few years several sensing schemes have been proposed and developed, aiming to overcome the issues related to the increase of the offset, principally caused by the scaling oft eh architectures, both in terms of voltages and dimensions of the devices. Two main techniques have been proposed in literature: offset compensation and offset cancellation [P.Bhatia, B.S.Reniwal, S.K. Vishvakarm, 2015], [J.S. Shah, D. Nairn, M. Sachdev 2013]. The most elementary and simple condition to be satisfied by the SA to obtain a correct reading is the following

$$\min\{|\Delta V_S|\} > \max\{|\Delta V_{OFF}|\} \qquad \text{(III.4)}$$

which is to say that the offset voltage value must always be lower than the sensing voltage value, allowing the correct amplifying of the BL signals along the two SLs. Aiming to satisfy inequality III.4 it is possible to maximize the minimum value of the module of ΔV_S or to minimize the maximum value of the module of the offset voltage, ΔV_{OFF}.

III.6.1 Offset Compensation Schemes

Maximizing the minimum value of the module of ΔV_S is of interest if and only if it is possible to amplify the incoming sense signal, because instead it would not be possible to overcome offsets higher than these signals. Hence, the most used technique of this type consists in pre-amplifying the sense signal before sending it to the SAs. Through that amplification the system tries to compensate the offset that is why these techniques are called offset compensation schemes. Offset compensation solution tends to amplify the sensed voltage using charge transferring and differential amplifiers in order to improve the signal-to-noise ratio (SNR).

III.6.2 Offset Cancellation Schemes

Minimizing the maximum value of the module of the ΔV_{OFF} is done using ad hoc schemes capable of directly reducing the offset; for that reason they are usually called reduction or cancellation techniques, even if they could exploit the amplification concepts too. Offset cancellation paradigm aims to reduce the offset voltage exploiting the high gains of operational amplifiers.

Moreover, it is possible to recognize to different approaches:

- Current Sense Amplifiers (CSA) [E. Seevinck, P.J.V. Beers, H. Ontrop, 1991], which operates well in the case of reduced voltage swings, but shows larger area occupation and ace times;
- Voltage Sense Amplifiers (VSA) [B. Wicht, T. Nirschl and D. Schmitt-Landsiedel, 2004], achieving lower area occupation, while having the drawback of higher power consumptions.

Aiming to overcome the principal limitations of the two approaches, hybrid current-voltage schemes have been proposed [M. Sharifkhani, E. Rahiminejad, S.M. Jahinuzzaman, M. Sachdev, 2011], [D. Ahn-Tuan, K.

Zhi-Hui, Y. Kiat-Seng, 2008], which main related issue is represented by the circuit area occupation nonetheless.

The developed system is a novel VSA scheme combining the offset cancellation and compensation strategies. The obtained area occupation for the proposed implementation results lower than in the case of other architectures previously presented. In addition, good performances in terms of energy consumption are achieved compared to other architectures developed in the same technology. In fact, even in the case a slightly higher energy per cycle is required, due to the presence of the inverter amplifiers, this drawback is compensated by a more compact implementation of the overall sensing circuit and by a better offset compensation. The circuit exploits a direct connection of the offset compensating inverters (OC-inverters) to the bit-lines, allowing separating the SA from the BLs while amplifying the voltage variations induced by the memory cell at the same time.

Using Cadence EDA tools and TSMC PDKs at 180nm, simulations have been conducted, showing that the proposed design is tolerant to much higher offset values than previously presented in literature ones, while achieving reduced memory access time, T_{ACC}, and higher operation frequency.

III.7 Proposed Scheme

Several sub-circuits make up the proposed scheme:

- a memory cell which is controlled by the WL signal enabling the reading and writing of the cell;
- a circuit to pre-charge the BLs at the same value before any reading operation, activating the EQ signal;
- a FCMOS Offset Compensation inverters (OC-inverters) per bit-line, which can be closed in feedback using the switches controlled by the PT1 and PT2 signals;
- a latch SA, which is made up by two FCMOS inverters controlled using the signals SAN and SAP.

A general scheme of the proposed sensing circuit and its various operations is depicted in Fig. III.8. It is possible to notice that while the I/O data could be read on the two SLs, SA1 and SA2, the circuit is placed between BL1 and BL2, which are respectively the BL and the complemented BL.

The sequence of the operations of the scheme is described in the following:

- First the circuit undergoes a pre-charge phase to equalize the BLs and the sense lines (SLs) to $V_{DD}/2$. In this phase the memory cell is not accessed, hence WL=0 and the SA latch is not enabled setting SAN= V_{DD} and SAP=0, while EQ, PT1 and PT2 signals

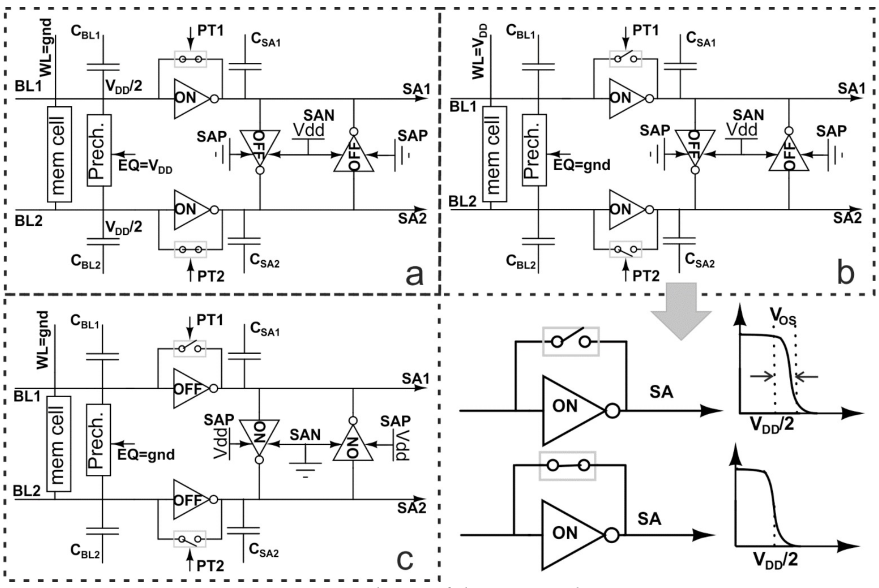


Figure III.8 *Scheme of the proposed circuit.*

are high. In order to optimize power consumption, in this phase the OC-inverters and the SA are switched off.

- The second phase is the offset cancellation one. In [Y. Watanabe, N. Nakamura, S. Watanabe, 1994] a way to strongly reduce the offset problem simply short-circuiting the input and the output is presented. This technique allows the PMOS and the NMOS of the OC-inverters to work near the maximum gain point, in the saturation region, where they behave as amplifiers. This allows canceling the offset which causes the skewing of the transfer characteristics of the OC-inverters. This could result in an attenuation of the voltage variation induced by the memory, which could cause an incorrect reading of the data. The considered offset is only the one due to electrical and geometrical differences mismatches in the parameters of the MOS devices attested on the two lines.
- After that the circuit proceeds to read the data from the memory cell in the offset compensating phase. In this phase WL goes high and the signals EQ, PT1 and PT2 are lowered (EQ=PT1=PT2=0). The access time, T_{ACC}, identifies the WL active period, which should be sufficient to amplify the voltage variation between the bit-lines, since the offset to be compensated by the sensing scheme depends upon the access time. The voltage difference between the bit-lines, Δv_{BL}, is amplified as $\Delta V_{SA} = A_{OC-INV}^{Max} \Delta V_{BL}$.

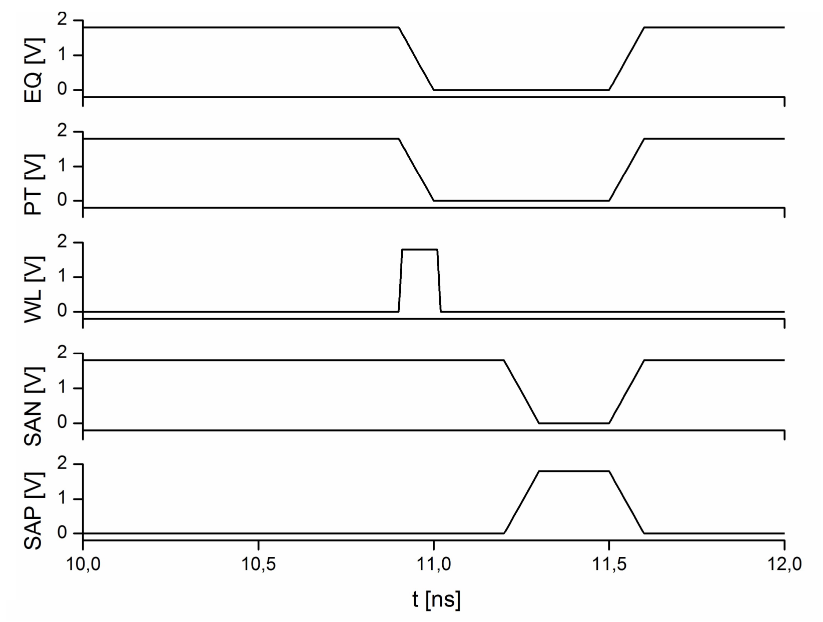


Figure III.9 *Waveform of the control signals over a single period of operation.*

- Finally, the sensing phase could be carried out, switching on the SA latch while turning off the OC-inverters. The SAN signal goes low (SAN=0), while the SAP signal goes high (SAP=V_{DD}), activating the latch and causing the SA lines to reach rail-to-rail values, giving the correct output. It is worth to underline that the OC-inverters invert the output values at the SA with respect to the sensed variation. For this reason, the outputs corresponding to BL1 and BL2 will be the signals SA2 and SA1, respectively.

The waveforms of the control signals for a single period of operation are shown in Figure III.9, while a circuit schematic of a MOS implementation of the described sensing scheme is reported in Figure III.10.

III.8 Results

Simulations for the proposed sensing scheme have been carried out using Cadence ADE environment using the TSMC 180 nm CMOS PDKs. An example of reading from the memory, in the case of a "0" stored value is give in Figure III.11. It is important to underline that the operation period shown in Figure III.11 has been dilated to clearly show the different phases of the reading operation. In the reported simulation, bit-line capacitances CBL1=CBL2=1 pF has been taken into account, while a V_{DD}=1.8 V have

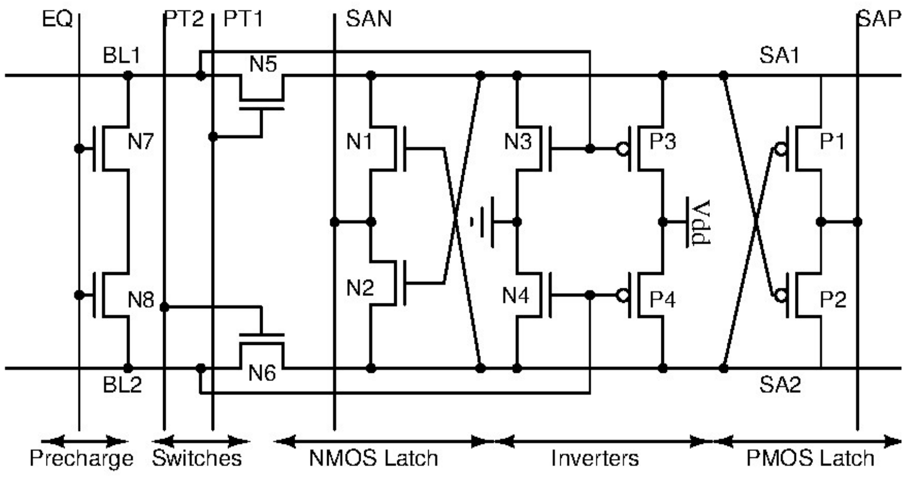


Figure III.10 *MOS implementation of the proposed sensing scheme.*

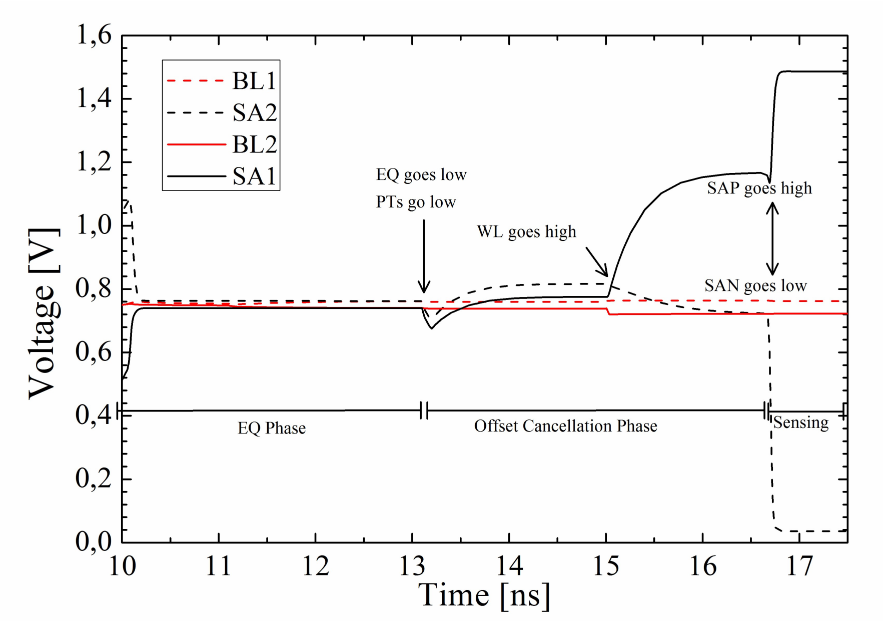


Figure III.11 *MOS BLs and SLs waveforms during the reading of a "0".*

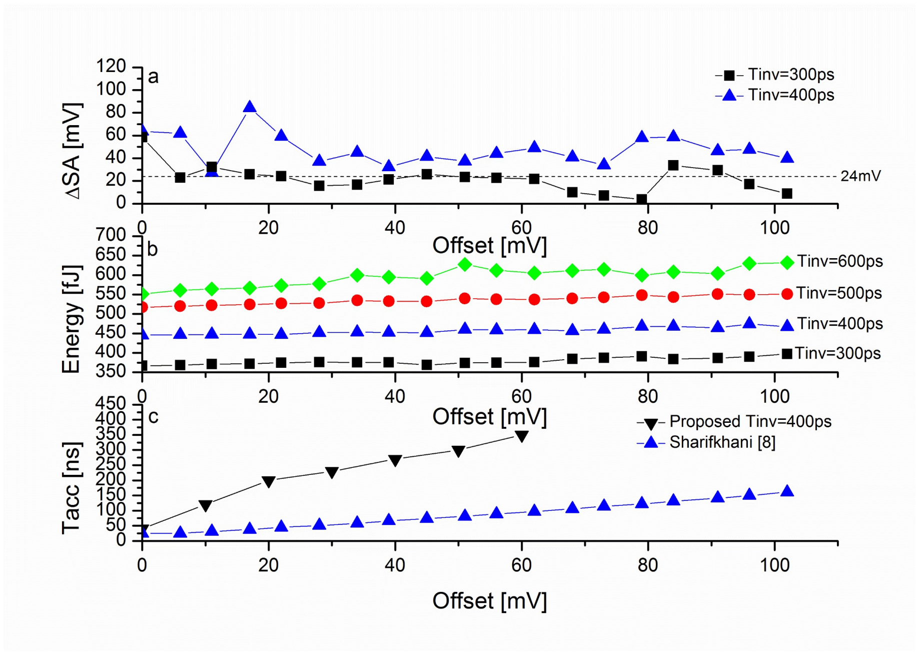


Figure III.12 *a) Sensed voltage varying VOS; b) Energy consumption varying VOS; c) Minimum access times varying VOS at TINV=400ps for the proposed scheme and for the design in [M. Sharifkhani, E. Rahiminejad, S.M. Jahinuzzaman, M. Sachdev, 2011].*

been imposed. The offset is increased varying the threshold voltage of the MOS devices, V_{TH}, obtaining a maximum V_{OS}=20 mV on the OC-inverters.

Moreover, a careful evaluation of the energy consumption and a study of the variations of the minimum sensed signal, T_{ACC} and T_{INV}, which is the period of time in which the OC-inverters must be turned on, has been conducted to obtain a better evaluation of the compensation phase when increasing the offset value. Moreover, the minimum T_{INV} has been chosen in order to achieve a minimum input variation Δv_{SA} before the turning on of the SA latch. Starting from the considerations developed in [S.-H.Woo, H.Kang, K.Park, S.-O. Jung, 2010], according to which in a latch based SA the Δv_{SA} parameter shows a σ = 8 mV in the case of a 180nm process, we imposed T_{INV} to obtain a minimum $\Delta v_{SA} = 3\sigma$; the results are shown in Figure III.12a. As it is possible to notice, the minimum value of T_{INV} capable of obtaining $\Delta v_{SA} > 3\sigma$ for all the considered range of offsets is T_{INV}=400 ps. A comparison in terms of access times is also shown in Figure III.12b where it is possible to notice that increasing T_{ACC} will cause an increase in the energy consumed by the structure. The energy consumption shows an almost linear dependence varying the T_{INV} while it is almost insensitive to the variations of

T_{ACC}. Due to these considerations, it is possible to state that reducing T_{ACC} and T_{INV} is an optimal solution to both achieve higher frequencies of operation and lower energy consumptions. T_{INV} is in particular critical since in this period the inverters shows high static power consumption. Figure III.12c reports a comparison between the access times achieved by the proposed design and the ones related to the scheme reported in [M. Sharifkhani, E. Rahiminejad, S.M. Jahinuzzaman, M. Sachdev, 2011]. The access time required to correctly performing the read operation increase with the offset value, but the increase rate is in our case lower than the one in [M. Sharifkhani, E. Rahiminejad, S.M. Jahinuzzaman, M. Sachdev, 2011]; in particular, our design achieves access times approximately 80% lower than those in [M. Sharifkhani, E. Rahiminejad, S.M. Jahinuzzaman, M. Sachdev, 2011].

The design layout has been developed with Cadence Virtuoso Analog Design Environment using TSMC 180nm PDK libraries. A detailed description of the developed layout is shown in Figure III. 13, in which it is possible to notice all the structure of the offset compensation circuit as well as accessory structures needed for testing and signal management, as the capacitance submodule and the buffer for the generation of the internal signals. Figure III.13 also shows the overall layout obtained using Virtuoso ADE with pins and wire bondings and the final obtained ASIC fabricated thanks to Europractice mini@sic Program 2016.

III.9 Comparisons

Table III.1 details the principal results of the proposed design and compares them with the ones achieved by implementations in the recent literature. Even if a higher number of transistors compose the proposed design, the possibility to use devices with smaller dimensions allows us to obtain a lower area occupation than the one in [D. Ahn-Tuan, K. Zhi-Hui, Y. Kiat-Seng, 2008]. In [D. Ahn-Tuan, K. Zhi-Hui, Y. Kiat-Seng, 2008] it is also possible to find other schemes for charge-transfer, ultra-low power and high speed. From a comparison of these implementations with our design, it is possible to state that the proposed scheme achieves better performances in both area occupation and power consumption. In addition, the performances in terms of sensing delay are improved, due to the offset compensation phase, which reduces the access time parameter. Hence, the design could handle higher offset values than the other implementations without reading errors. A lower period of operation is also achieved, thanks to the lower sensing delay; the minimum period of operation is 1.5 ns. The implementation in [J.S. Shah, D. Nairn, M. Sachdev 2013] has a lower energy consumption, but this solution is only capable of withstanding offsets related to a variation in the threshold voltage value of the MOS devices. On

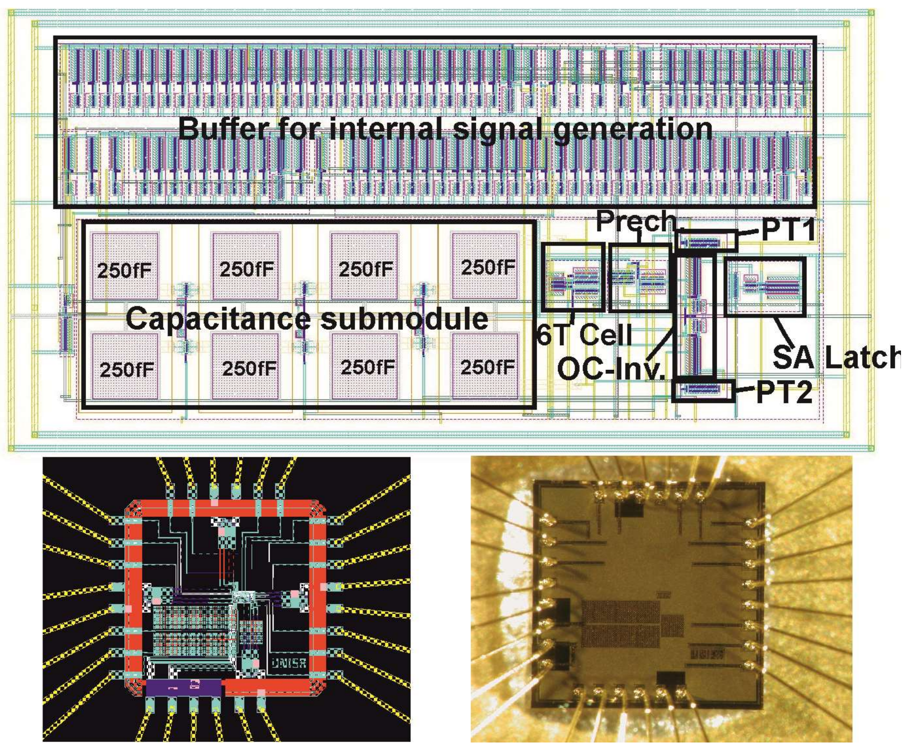


Figure III.13 *Design layout obtained using Virtuoso ADE and TSMC PDK 180nm, overall layout showing also pins and wire bondings and the final obtained ASIC developed thanks to the Europractice mini@sic Program 2016.*

the contrary, the proposed solution obtains good compensation performances also in the case of variations of the geometrical parameters of the devices.

The presented scheme achieves state-of-the-art performances in terms of sensing delay, area occupation and offset compensation in the case of an 180nm technology. Furthermore, the proposed design shows a good tradeoff between energy consumption, area occupation and speed. In order to understand how the circuit performances vary with the scaling of the devices, future studies could aim to investigate the scaling of the proposed scheme. In fact, the latest designs are developed in shrunk technologies [N.Chandoke, N.Chitkara, A. Grover, 2015], [H. Jeong, J. Park, T.W. Oh, W. Rim, T. Song, G. Kim et al, 2016], which raises new challenges in the development of sense amplifier circuits.

TABLE III.1 *Comparison of the proposed design with the related literature.*

	Prop.	Shah	Sharifkhani	Ahn-Tuan
Technology [nm]	180	180	180	180
Bias Voltage [V]	1.8	1.8	1.8	1.8
Number of MOS	16	13	13	16
Area [μm^2]	54	--	--	376
Sensing Delay [ns]	0.13	0.7	0.3	0.26
Max ΔV_{OS} [mV]	100	±35	40	--
T_{ACC} [ps]	58	--	270	--
Energy [pJ]	0.467	0.212	0.84	--

III.10 Conclusions

It is then possible to state that the proposed SA scheme is capable of improving the performances in terms of absolute offset rejection if compared to other circuits in literature aiming to the same result, while achieving state-of-the-art results also in terms of occupied area and sensing delay. Further studies have to be conducted on the real circuit, in order to obtain real case results while understanding how the circuit could be further improved.

However, the desired project and improvement of a memory scheme allow the development of faster ASICs, with particular reference to the ones in which memory accesses represent the main bottleneck mainly due to the high number of memory accesses.

Chapter IV
Gabor Filtering Applications for Visual Search

IV.1 Introduction

After the development of a new SRAM SA scheme, in order to obtain better performances in memory constrained application using high amounts and high density of memory cells, the research was addressed to develop new HW systems for pre-filtering and accelerate the operations in multimedia applications, with particular attention to VS ones. These kind of applications are fundamental in nowadays electronic systems in IoT and NN, where they usually compose the initial stage of processing nodes in dedicated applications, which are nowadays more and more important also in handheld and power constrained devices.

In this field, one of the most used filters is the Gabor filter, due to its attributes, which make it fit for several applications. In fact, Gabor filters are a class of linear band-pass filters, allowing selecting the frequency of interest via tuning of the filter parameters, and, moreover, they are capable of minimizing the joint uncertainty for frequency and spatial position [J.M. Guo, H. Prasetyo, K. Wong, 2014]. The proposed work, developed in this chapter, is then focused on this kind of filters, from the related mathematical background to the implementation and evaluation related to the developed HW system.

Several studies showed the capability of Gabor filters of providing a well-fit description of mammalian visual cortex cells [J. G. Daugman, 1985], which of course make this family of filters very attractive for computer vision applications. Moreover, Gabor filters produce robust features for the detection of edges and corners in image applications [J.-K. Kamarainen, V. Kyrki, H. Kälviäinen, 2002], while showing a certain selectivity along preferential orientations via the selection of the number of angles in the range $[0, \pi]$ along calculate the filtering operations, which guarantees a certain rejection to rotations. These are the main reasons why Gabor filters play a key role also in edge and corner detection, segmentation and gait analysis applications [W. Jiang, K.-M. Lam, T. Z. Shen, 2009], [W.-C. Zhang, F.-P. Wang, T. L. Zhu, Z.-F. Zhou, 2014], [H. Hu, 2013], [B. Kwolek, 2005].

However, in case of real-time requirements the high computational complexity of such filters could be overcome only recurring to hardware (HW) implementations of the filtering operation itself. Moreover, through a careful optimization of the filtering structure, this could result also in low-power consumption operations, while still achieving good accuracy in the results. It has also to be underlined that the recent literature presents very few HW oriented designs employing Gabor filters, due to their complexity. Works in [E. Cesur, N. Yildiz, V. Tavsanoglu, 2011] and [E. Cesur, N. Yildiz, V. Tavsanoglu, 2012] show real-time operations achieved by using reduced 3x3 Gabor-like kernels and one orientation only to implement a Cellular Neural Network (CNN). Another implementation is described in [Y.C.P. Cho, N. Chandramoorthy, K.M. Irick, V. Narayanan, 2012], but it is strictly platform-dependent since the good performances are due to the use of high-frequency overclocked DSPs embedded in modern FPGAs and thus the solution is also of limited interest. Performances of the Gabor filter and their computational complexity give rise to a number of trade-offs, which are regulated by the high number of parameters defining the kernels and the number of orientations. In several cases Gabor-like solutions approximating Gabor filters allows obtaining acceptable performances, while significantly reducing the accuracy, sometimes in ways not acceptable for particular application.

Moreover, these techniques find interesting applications in medical diagnostics. In fact, the significant improvements medical diagnostics underwent in the last years, due to the availability of new apparatus, e.g. x-ray Computed Tomography (CT) scan and Magnetic Resonance (MR), based on the detection of anomalies from acquired images, and therefore to image and data processing techniques allowed to further develop ad hoc multimedia systems for these kind of applications. In particular, the development of methods for the accurate identification of specific elements in the images is fundamental to achieve good detection and acceptable results from a diagnostic standpoint [C.-Y. Lu, B.-Z. Jing, P.P.K. Chan, D. Xiang, W. Xie, et al., 2016], [G. Humpire-Mamani, A. J. M. Traina, C. Traina, 2012]. Furthermore, the presence of artifacts, blurring effects and noise in the acquired images causes difficulties in the interpretation of the results [B. Ergen, A. Çinar, G. Aydin, 2012.], which, in turn, could result in some cases in wrong diagnosis. Finally, several medical exams require high resolution image processing, such mammographic exams, which in some cases could require the processing of up to twelve million pixels.

In this context the development of an ASIC allows the possibility of closely coupling the image processor with medical sensors obtaining images capable of supporting the medical team decisions. This is for example the case of sensors used in laparoscopic exams, where cameras use resolutions up to Full-HD (1920x1080 pixels) and 4K-UHDTV (3840x2160 pixels) and low power operations are required.

The developed work shows a careful choice of the parameters and allows an accurate implementation of 2D Gabor filters with a computational complexity that can be adapted to differently capable target platforms. In order to show this, different HW designs of a Gabor filter based edge detection system have been developed, implementing two, four and eight orientations and all capable of real-time processing with different ADP performances and accuracies. The derived ASIPs are targeted to both FPGAs and ASICs, using CMOS 90nm std_cells. Moreover, developing different design trade-offs it is possible to achieve real-time processing using working frequencies lower than the maximum one, allowing to obtain lower power consumptions, which makes the solutions promising for further System-on-Chip (SoC) developments, integrated with acquisition sensors, together with the obtained results in terms of occupied area.

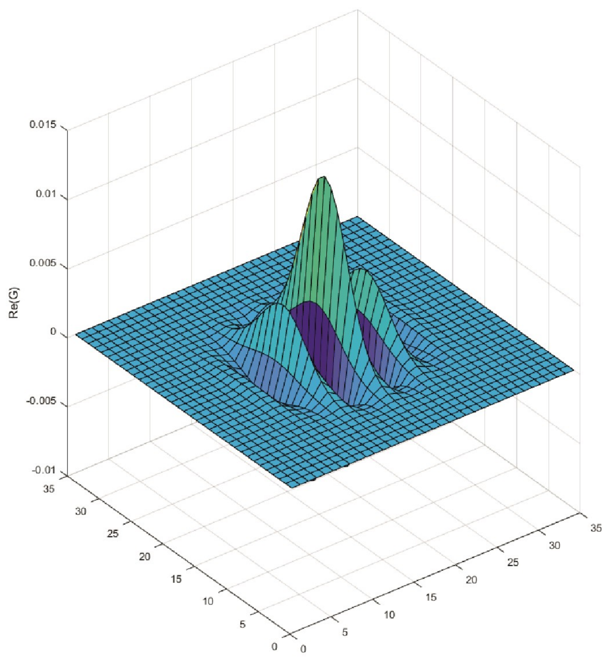


Figure IV.1 *Real part of Gabor filter with f_0=0.2, θ=0, γ=1 and η=1.*

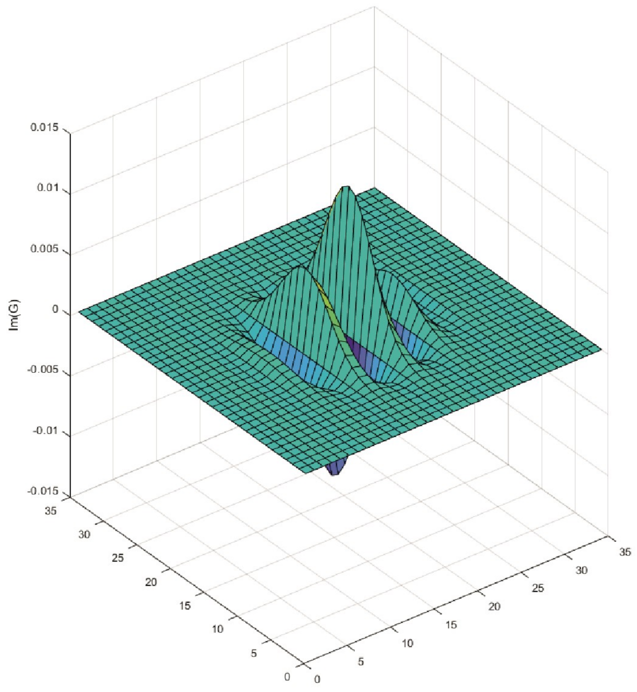


Figure IV.2 *Imaginary part of Gabor filter with f_0=0.2, θ=0, γ=1 and η=1.*

IV.2 Theoretical Background

Gabor functions are made up by the multiplication of two simpler functions: a Gaussian and a complex exponential. It is worth to note the dimensions of the Gabor filter to be used obviously vary with the particular application and that in the following the focus will be on 2D Gabor functions, obtained multiplying a generic two-dimensional Gaussian function for a complex exponential. The obtained function could be written in the form [J.-K. Kamarainen, V. Kyrki, H. Kälviäinen, 2002]

$$\psi(x,y;f_0,\theta)=\frac{f_0^2}{\pi\gamma\eta}e^{-\left\{\frac{f_0^2}{\gamma^2}x'^2+\frac{f_0^2}{\eta^2}y'^2\right\}}e^{j2\pi f_0 x'} \tag{IV.1}$$

where the terms *x'* and *y'* are further specified as

$$\begin{cases} x'=x\cos\theta+y\sin\theta \\ y'=-x\sin\theta+y\cos\theta \end{cases} \tag{IV.2}$$

with:

- x and y representing the spatial coordinates of the considered filter;
- f_0 central frequency of the considered Gabor filter;
- θ rotation angle for the particular orientation of the filter;
- γ sharpness of the Gaussian along the major axis;
- η sharpness of the Gaussian along the minor axis.

An example of a Gabor function for the set of parameters f_0=0.2, θ=0, γ=1 and η=1 is shown in Figure IV.1 and Figure IV.2, for the real and imaginary part of the function, respectively. As it is possible to notice, the real part of the Gabor function is an even function, while its imaginary part is an odd one. These properties allow exploiting the separation property of Gaussian based kernels along their orthogonal directions. In turn, this lead to obtain a reduction in the computational complexity of the filtering operations developed in HW along the orthogonal directions θ_s={0, π/2}, as shown in the following. The FT of the Gabor function is shown in Figure IV.3, where it has to be underlined that just half of the function is shown, since for the odd property of the function two lobes symmetric with respect to the origin of the axis have to expected.

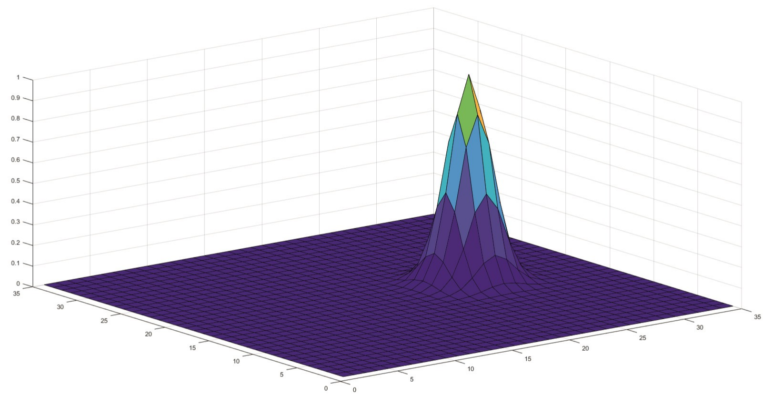


Figure IV.3 *FT of Gabor filter with f_0=0.2, θ=0, γ=1 and η=1.*

It should now be clearer how the features obtained via Gabor filters are stable functions in terms of image rotation, frequency scaling and translation operations. These properties allow us to state that Gabor features are robust in terms of noise rejection, image distortion and objects deformation. In particular, it is now evident how the invariance to image rotations is obtained thanks to the rotation property of the Gabor function, which is to say thanks to chosen set of angles along to perform the filtering operation. Having larger angle sets obviously provides better rejection to a wider set of angle rotations. In fact, the filter response for rotated images could be obtained via filtering the original non-rotated image for the rotated Gabor filter, which can be easily obtained setting θ angle properly [J.-K. Kamarainen, V. Kyrki, H. Kälviäinen, 2002].

Furthermore, from the considerations developed in [W. Jiang, K.-M. Lam, T. Z. Shen, 2009] and [F. Pellegrino, W. Vanzella, V. Torre, 2004] it is possible to say that in the context of edge detection applications it is possible to develop robust results using only the imaginary part of the Gabor function and then of the filtered data. In fact, since the input of the image is always represented by real values in these cases, it is possible to use ass filter only the imaginary function

$$G(x,y,f_0,\theta)=\mathrm{Im}\{\psi(x,y;f_0,\theta)\}=\frac{f_0^2}{\pi\gamma\eta}e^{-\left\{\frac{f_0^2}{\gamma^2}x'^2+\frac{f_0^2}{\eta^2}y'^2\right\}}\sin(2\pi f_0 x') \qquad \text{(IV.3)}$$

which avoids us to develop complicated modulus calculations, achieving a further reduction in the time required to perform the filtering operations and of the overall number of operations to be performed to carry out the final result.

IV.3 Mathematical Remarks and Algorithmic Considerations

From the observations developed in [F. Pellegrino, W. Vanzella, V. Torre, 2004] the best way to detect also thin edges in images is not to obtain the candidate feature via a linear sum of the different orientations results, calculated over each scale. On the other hand, to detect these edges it is more convenient to use only the maximum result along the various orientations for each considered pixel coordinates. In order to avoid the use of a bank of Gabor filters with different central frequencies, which would complicate the design, a multiscale Gabor filter can be instead implemented to achieve a certain invariance to scale changes for the obtained features, as shown in [W. Jiang, K.-M. Lam, T. Z. Shen, 2009] and [G.D. Licciardo, T. Boesch, D. Pau, L. Di Benedetto, 2016]. This solution allows to obtain a certain invariance to scale changes while not complicating excessively the design, allowing to obtain acceptable performances in terms of speed, accuracy, noise rejection, invariance, power and area occupation.

The method is applied on grayscale images, filtered using the Gabor functions derived from the chosen set of parameters and orientations at each scale. Since Gabor filters could assume negative values, only positive values coming out from the filtering stage are taken into account, to obtain only positive or null values at the end of the calculations, according both to the significant values for the image coding and to a maximum selection methodology. This allows to obtain a set of $N_{Sc}\cdot N_{Or}$ candidate features, where N_{Sc} represents the number of considered scales in the design and N_{Or}

represents the number of orientations calculated for each scale. From the set of candidates obtained for each coordinate of the image to be filtered only the maximum result is chosen as the proposed feature to compose the final resulting image.

The following sections are dedicated to the determination of the parameters of the Gabor filters, obtained developing several considerations on the parameters themselves and the chosen number of scales, in order to establish a set of parameters fit for the development of a HW edge detection design, capable of obtaining acceptable trade-offs in real case scenarios.

IV.3.1 Filter Central Frequency, f_0

f_0 represents the central spatial frequency of the Gabor filter, which is always defined in the range [0, 0.5], since it is possible to state that the maximum frequency to be considered, f_N=0.5, is given by the Nyquist theorem. The central frequency has to be carefully chosen, since frequencies below 0.1 could cause a degradation in the obtained features accuracy due to higher blurring in the resulting image and to the impossibility of following sharp edges. On the other hand, frequencies above 0.2 may require filters with narrower bandwidths, requiring an unwanted oversampling of the kernel with respect to the resolution grid of the input image [W.-C. Zhang, F.-P. Wang, T. L. Zhu, Z.-F. Zhou, 2014]. Moreover, it could be possible to consider more central frequencies in order to obtain more results and a more robust system, obviously at the cost of implementing more units, and hence at the cost of higher area and power consumption or lower operating frequencies.

IV.3.2 Set of Orientations, θ

As previously seen, the number of considered orientations is one of the primary concerns in Gabor filter applications, if not the one of the most important ones, since it determines the rejection to rotations of the original image. The considerations to be developed in this case regard the desired trade-off between the computational complexity of the design to be developed and the level of disturbance rejection to be achieved. In fact, using a growing number of orientations tends to give better performances in terms of detected edges, but at the same time another convolution per scale has to be performed for any new direction to be computed. In turn, this increases the total number of orientations to be computed and the number of overall operations to be carried out; in particular, the increase in the number of MAC operations is linear with the number of considered operations.

Another important characteristic to notice is that the number of orientations to be considered is always a power of 2 and then the complexity of the operation increase by such a factor every time we increase the number of angles; furthermore, the angles span uniformly the space and thus the different rotation angles differ by fixed quantities.

Table IV.1 and Table IV.2 report the PSNR and SSIM indexes calculated taking as reference the image obtained filtering with the eight orientations set, $\theta_{Eight}=\{k\pi/8\}$, where k is an integer in the range [0,7], on two different sets:

- a public domain test image set composed by a significant subset of online test image database available at [https://homepages.cae.wisc.edu/~ece533/images/];
- a public domain test image database, used for medical applications related to retinal blood vessel detection, DRIVE database [https://www.isi.uu.nl/Research/Databases/DRIVE/].

From the reported data, it is possible to state that, in terms of detection capability, the two orientations implementation, requiring the lowest number of operations, fails in detecting several edges if compared to the reference case, while the four orientations case combines edge detection comparable to the eight directions case, without overly increasing the overall number of operations required. In the following, also a method to obtain an eight orientation design based on some simplifications is carried out, in order to achieve acceptable accuracies when required.

TABLE IV.1 *PSNR and SSIM values obtained comparing the imaginary part of 2Or and 4Or results with the eight orientations case for [https://homepages.cae.wisc.edu/~ece533/images/] database.*

Image	2Or		4Or	
	PSNR [dB]	SSIM	PSNR [dB]	SSIM
airplane	21.90	0.72	25.16	0.87
baboon	20.11	0.58	25.47	0.86
boat	23.40	0.68	26.78	0.84
Lena	23.42	0.66	27.56	0.88
peppers	23.49	0.68	26.00	0.84
Average	*22.46*	*0.66*	*26.19*	*0.86*

TABLE IV.2 *PSNR and SSIM values obtained comparing the imaginary part of 2Or and 4Or results with the eight orientations case for [https://www.isi.uu.nl/Research/Databases/DRIVE/] database.*

Image	2Or		4Or	
	PSNR [dB]	SSIM	PSNR [dB]	SSIM
Average	*25.70*	*0.82*	*29.71*	*0.91*

IV.3.3 Gaussian Sharpness Values, γ and η

The γ and η parameters are the standard deviations of the Gaussian envelope along *x'* and the Gaussian projection of the Gabor filters along *y'*, respectively. Their tune is fundamental to avoid blurring problems and excessive sensitivity of the filters to small variations of the images. Moreover, their values must be evaluated in conjunction with the considered scales kernel dimensions, N_S. Another important evaluation to be developed regards the quality of the input image set, since the parameters could be finely tuned to obtain improvements for a certain specific application over another. In particular, it is possible to proceed through objective and subjective evaluations of the results in order to obtain a better behavior for the considered system. In the following, γ=1 has been imposed to include at least one period of the periodic component in the Gaussian envelope, and thus to ensure the presence of the odd function needed for edge detection [R. Mehrotra, K. R. Namuduri, R. Ranganathan, 1992]. This is due to the fact that the Gaussian shape has been adequately approximated by imposing the minimum filter dimensions to 6γ+1 as shown in [G.D. Licciardo, A. D'Arienzo, A. Rubino, 2015]. The previous considerations result in giving rise to a filter having minimum dimensions 7x7; however, in our design this filter has been combined to a 9x9 one to implement a multiscale architecture improving the accuracy and the stability of the final results. Finally, η=15 has been imposed in order to simplify equation (IV.3) in a HW friendly fashion, as will be clearly shown in the following.

IV.3.4 Scales Dimensions, N_s

In the development design only filters having the same dimensions along the two dimensions, N_x and N_y, for which N_x= N_y =N_s; this is done both to simplify the design and to consider only symmetric Gabor kernels. Through several observation it has been possible to state that impose high values of N_s will result in excessive blurring along the detections of the edges, while lowering the parameter too much will result in too sensitive systems in the case of noisy input images.

Due to these considerations and aiming to achieve both good edge detection and a saving in the used resources and arithmetic units, we decided to implement two scales having dimensions 7x7 and 9x9. Moreover, it is worth to notice that, if necessary, it is possible to obtain further enhancements in the edge detection results using higher number of scales or orientations. In this sense, the proposed choice is obviously a trade-off between the detection capability and accuracy achievable by the design and its complexity, which mainly depends on the overall number of arithmetic operations involved in the calculation process.

Two examples of the obtained results for two images taken from the previously considered databases are shown in Figure IV.4 and Figure IV.5, together with the varying number of orientations used and the established set of parameters.

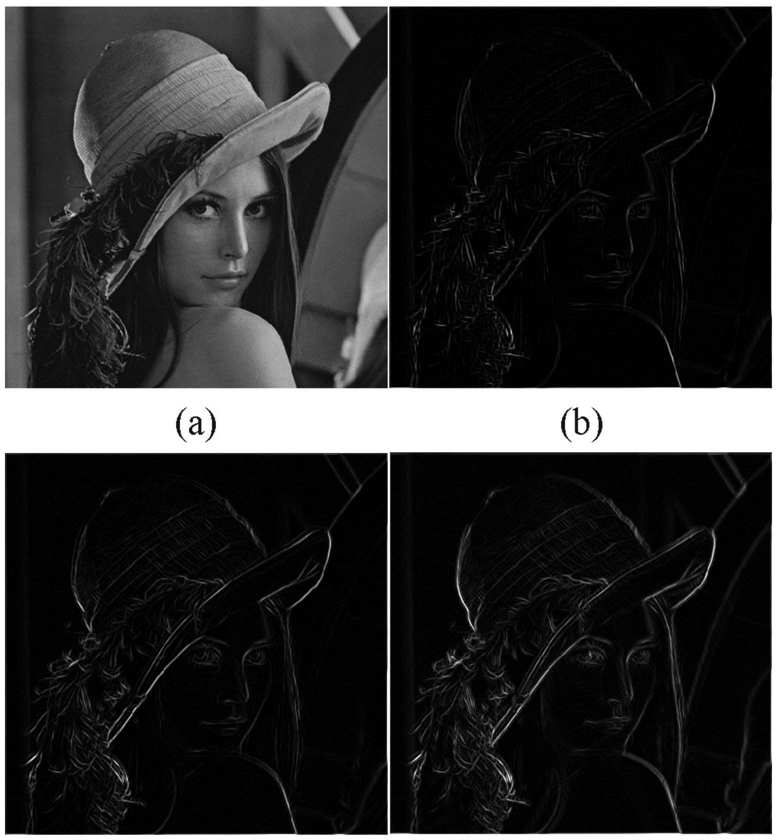


Figure IV.4 *(a) Original Lena image and obtained results for* $f_0 = 0.2$, $\gamma = 1$, $\eta = 15$ *considering 7x7 Gabor filter and (b) two, (c) four and (d) eight orientations.*

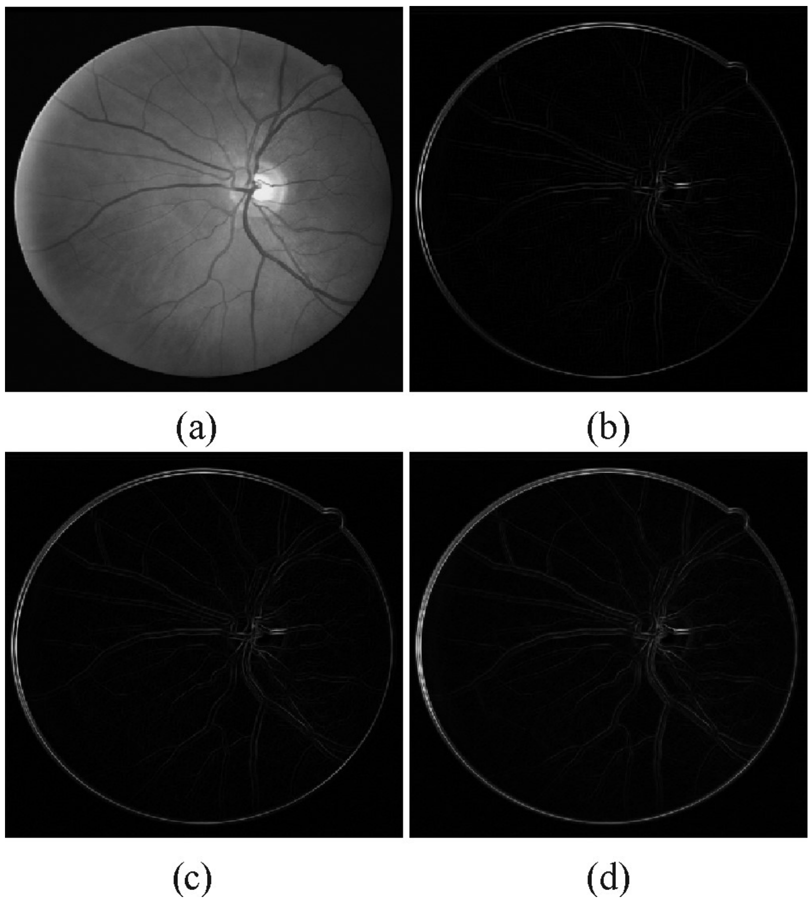


Figure IV.5 *(a) Original DRIVE dataset image and obtained results for* $f_0 = 0.2$, $\gamma = 1$, $\eta = 15$ *considering 7x7 Gabor filter and (b) two, (c) four and (d) eight orientations.*

IV.4 Numerical Remarks

Usual image filtering application implemented via SW use FP32 data according to the IEEE-754 FP32 format. As seen in Chapter I, these implementations are often too heavy to be directly implemented in HW. In fact, while FP32 obviously guarantees high accuracy in the final results, the use of this type of coding is usually avoided, since it results in an unacceptable degradation of the performances in terms of working frequencies, power consumptions, area occupation and resource usage. This is obviously related to the general higher complexity of the arithmetic units used in FP32 operations in combination with the high number of MAC units to be considered in such filtering operations. Due to this computational burden and the consequent unacceptable performance degrading, HW implementations usually recur to FI formats instead of FP ones, accepting a certain degradation in the accuracy of the results and choosing at the same time a number of significant bits in the codelength capable of guaranteeing results close enough to the a FP implementation.

Considerations have been developed starting from the theory developed in [G.D. Licciardo, T. Boesch, D. Pau, L. Di Benedetto, 2016] for the case of 2D Gaussian filters for the modeling of a worst case scenario on the obtainable tiles to be processed for the various considered angle orientations. The so developed model allows for the development of a method for choosing a minimum codelength in which the Least Significant Bit (LSB) is not affected by the approximation errors, meaning that adding more bits would not result in an improvement in the overall results, since them would be affected by the approximation errors. The data obtained from these considerations on approximations are reported in Table IV.3, varying the number of bits of the coefficients codelength.

TABLE IV.3 *Maximum error on the Gabor Filter coefficients varying the FI codelength for the chosen set of parameters.*

Number of Bits	Maximum Error
14	1.04E-04
15	5.95E-05
16	**2.98E-05**
17	1.27E-05
18	7.58E-06

The FI codings under test for the Gabor kernel coefficients approximations are made up by 1 bit for the sign, 1 bit for the integer part and varies the number of bits used for fractional part. We choose for the filter coefficients a FI16 coding, in order to obtain good accuracy, while containing the size of the arithmetic units, at the same time.

After this first evaluation, in order to achieve a better understanding of the error propagation derived from the coefficients approximation, the worst case errors for the whole filtering operation have been computed along the different orientations. In this case the pixels of the input tile have been imposed to have maximum values (255) in correspondence of the positive values of the filter and minimum ones (0) in correpondence of the other filter values. Since the final results are by definition all positive numbers, as preoviously stated, no sign bit will be considered in the final results. This allows us to consider smaller codelenghts and in particular to obtain a FI12 coding in which the 8 Most Significative Bits (MSBs) represent the integer part, while the 4 LSBs respresent the fractional one. Finally, it has to be noticed that the reduction of the codelength for the final results is mainly related to the propagation of the errors caused by the approximations on the original Gabor kernel coefficients along the convolution calculations. Due to that, a FI codelength in which the LSB results not affected by those approximations and errors has been chosen. Table IV.4 reports the results discussed above, varying the codelength for the final result.

TABLE IV.4 *Maximum error on the filtering results for the chosen codelength of the Gabor coefficients.*

Number of Bits	Maximum Error	LSB
10	1.05E-01	2.50E-01
11	4.95E-02	1.25E-01
12	**3.57E-02**	**6.25E-02**
13	3.57E-02	3.13E-02
14	2.00E-02	1.56E-02

IV.5 Proposed Designs

The chosen designs to be implemented and considered are mainly five:

- a two orientations implementation (2 Or);
- a canonical four orientations implementation (4 Or);
- a shared four orientations implementation (4 Or-Shared);
- a canonical eight orientations implementation (8 Or);
- a shared eight orientations implementation (8 Or-Shared).

The considered sets of orientations are then

$$\begin{cases} \theta_{Two} = \{0, \pi/2\} \\ \theta_{Four} = \{0, \pi/4, \pi/2, 3\pi/2\} \\ \theta_{Eight} = \{0, \pi/8, \pi/4, 3\pi/8, \pi/2, 5\pi/8, 3\pi/2, 7\pi/8\} \end{cases} \tag{IV.4}$$

It is important to highlight how the shared designs have been derived from the canonical ones through a careful resource reorganization and units sharing in case in the specific application would be preferable to sacrifice the maximum frequency to obtain the same accuracies of the canonic designs, while saving resources and obtaining smaller area occupation. The such obtained trade-offs are fundamental to correctly evaluate what would be the best fitting design for the specific application.

The general backbone of all the designs is sketched in the block diagram of Figure IV.6, while a detail of the various units composing it is provided in the following.

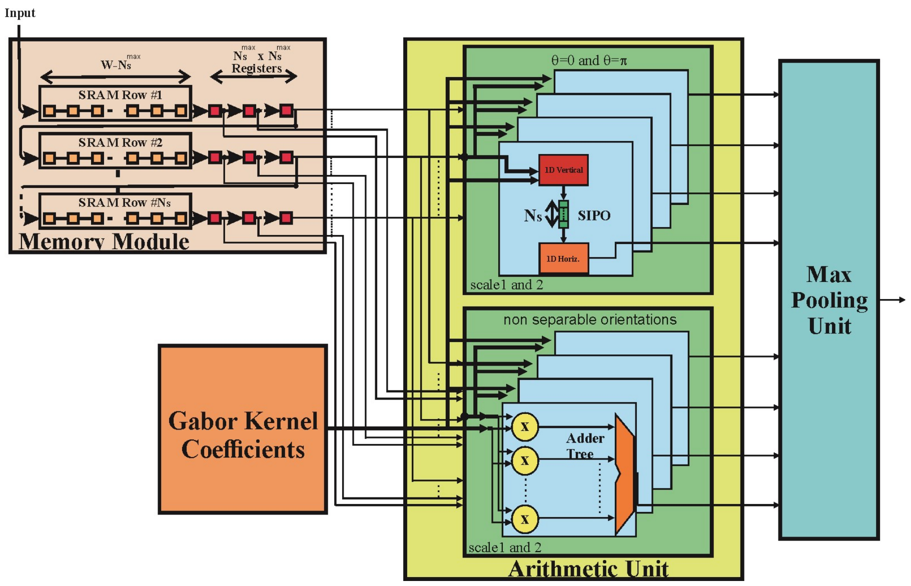


Figure IV.6 *General block scheme of the proposed designs.*

IV.5.1 Gabor Coefficients Memories Figure IV.5 ***General block scheme of the proposed designs***

The coefficients of the kernels of the Gabor filters are stored in dedicated memory structures, in order to be sent to the Arithmetic Units at any time the calculations have to start. Another information comes from the consideration that the function in equation (IV.3) is a symmetric one; thanks to that characteristic it is possible to write that

$$G(i,j,f_0,\theta)=G_{i,j}^{\theta}=G_{i,j}^{\theta+\pi/2} \qquad \text{(IV.5)}$$

for $i, j \in I$ and $\theta \in [0, \pi/2]$, it is possible to state that the dimensions of the data to be stored and, thus of the memories to be implemented, can be reduced, since a lower number of coefficients is needed, due to the symmetry itself. In particular, for the orientations $\theta=0$ and $\theta=\pi/2$ all the rows and columns, respectively, have got the same associated value and hence it is possible to store only N_S coefficients instead of N_S^2.

In the case of the orientations $\theta=\pi/4$ and $\theta=3\pi/4$ another symmetry could be found along the diagonal axis of the associated kernel matrices (minor and major diagonal, respectively), allowing us to state that only $(2N_S-1)$ coefficients are needed in this case.

Finally, for the orientations $\theta = \pi/8$, $\theta = 3\pi/8$, $\theta = 5\pi/8$ and $\theta = 7\pi/8$ it is not possible to simply define such a symmetry along an axis and just a central symmetry could be found, allowing us to obtain a reduction of the coefficients up to one half of starting ones. All the considerations developed apply for the particular set of parameters, which allows to obtain a considerable reduction in the total number of coefficients to use, thanks to their symmetry and to considerations related to their numerical approximations.

Considering the previous results and the use of the FI16 coding, three memories of 112 bits, 256 bits and 384 bits must be instantiated when N_S=7; for N_S=9, the dimensions of the memories to be stored increase to 144 bits, 400 bits and 640 bits. Figure IV.7 reports the memory requirements for different scales, varying the number of orientations, while Figure IV.8 details the symmetry of the kernels for the various orientations.

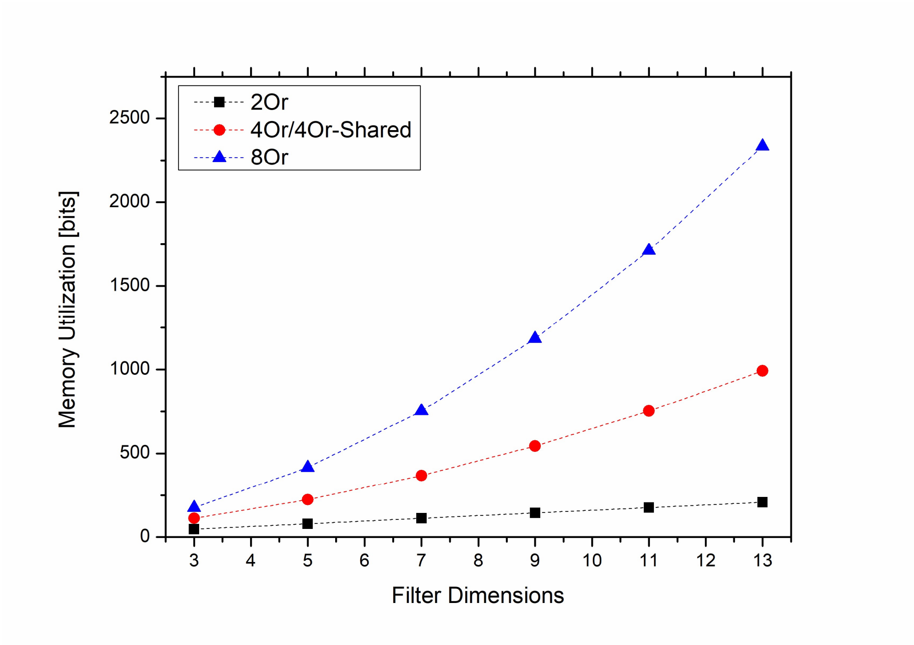


Figure IV.7 *Memory utilization in bits varying the filter dimensions.*

IV.5.2 Memory Module

The Luma components of the incoming pixels are coded on 8 bits, acquired in raster scan order from the image source and are then stored in a Memory Module in order to wait for the filtering operations. The Memory Module is arranged to operate like a long FIFO, having overall dimensions Wx$N_S^{\max}$, where W is the width of the input image and $N_S^{\max}$ is the dimension

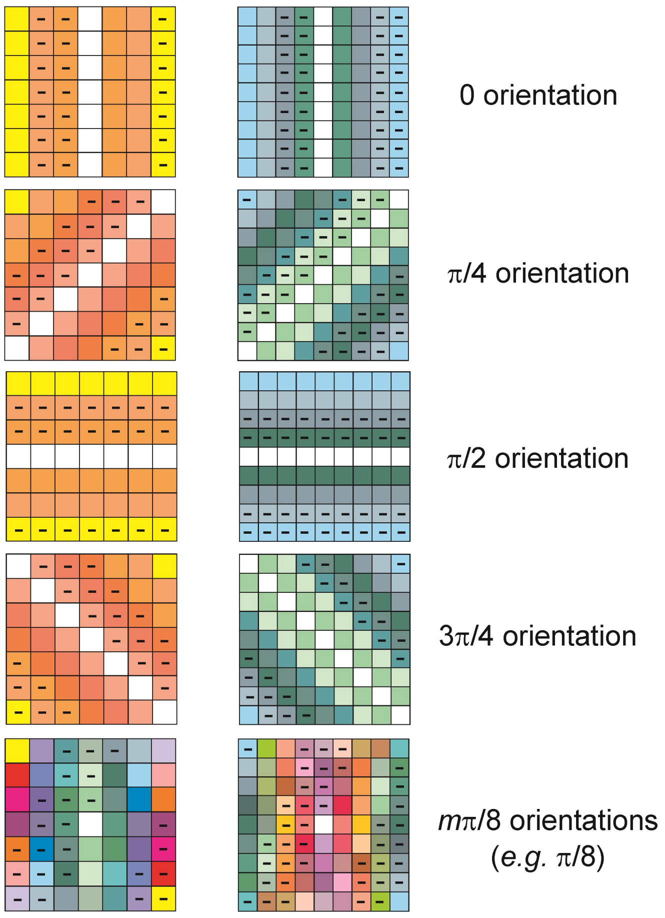


Figure IV.8 *Scheme of the various kernel matrices and their symmetries.*

of the greatest kernel. Due to these considerations, the design uses N_S^{max} dual-port RAMs, one for each row, which has dimensions 1x(W-N_S^{max}), while a bank of N_S^{max}xN_S^{max} registers terminates the rows to make the pixels available in parallel to the Arithmetic Unit. The overall behavior is the same shown for the memory module in Chapter I.

IV.5.3 Arithmetic Unit

The Arithmetic Unit represents the main calculation unit of the proposed design; it computes the partial results elaborating the initial data coming from the Gabor Coefficients Memories and the Memory Module and develops the needed filtering operations for each orientation, combining MAC operations.

Exploiting the separability property of equation (IV.5) along the directions θ_{Two} as

$$\begin{cases} G_0 = G(i,j,f_0,0) = \dfrac{f_0^2}{\pi\gamma\eta}\sin(2\pi f_0 x)e^{-\frac{f_0^2}{\gamma^2}x^2} \times e^{-\frac{f_0^2}{\eta^2}y^2} = G_0^H \times G_0^V \\ G_{\pi/2} = G(i,j,f_0,\dfrac{\pi}{2}) = \dfrac{f_0^2}{\pi\gamma\eta}\sin(2\pi f_0 y)e^{-\frac{f_0^2}{\gamma^2}y^2} \times e^{-\frac{f_0^2}{\eta^2}x^2} = G_{\pi/2}^V \times G_{\pi/2}^H \end{cases} \quad \text{(IV.6)}$$

it is possible to reduce the complexity of the computation along these directions from O(N_s^2) to O(N_s). Furthermore, the choice of taking η=15 in the parameters choice simplifies (IV.6) as:

$$\begin{cases} G_0 = G_0^V \times G_0^H = I^T \times G_0^H \\ G_{\pi/2} = G_{\pi/2}^V \times G_{\pi/2}^H = G_{\pi/2}^V \times I \end{cases} \quad \text{(IV.7)}$$

where I is the vector defined having all unitary values, I=[1,1,…,1].

It is worth to underline that the impossibility to separate the kernel along the remaining orientations is the main reason of the increment of arithmetic components. Figure IV.9 reports an analysis of the required number of adders and multipliers as a function of the filter dimensions and of the number of orientations. Aiming to mitigate these problems, the Shared designs presents a reduced Arithmetic Unit, obtained by re-using the same structures to calculate both the $\pi/4$ and $3\pi/4$ orientations in the case of the four orientations implementation and to calculate the coupled orientations ($\pi/8$, $5\pi/8$) and ($3\pi/8$, $7\pi/8$), respectively.

These implementations reduce the number of adders and multipliers used, with respect to the canonical designs, as reported as reported in Table IV.5. Obviously, the cost to pay is in this case the increase of the latency time, because the shared designs need two clock cycles to filter a single pixel along all the orientations for each scale.

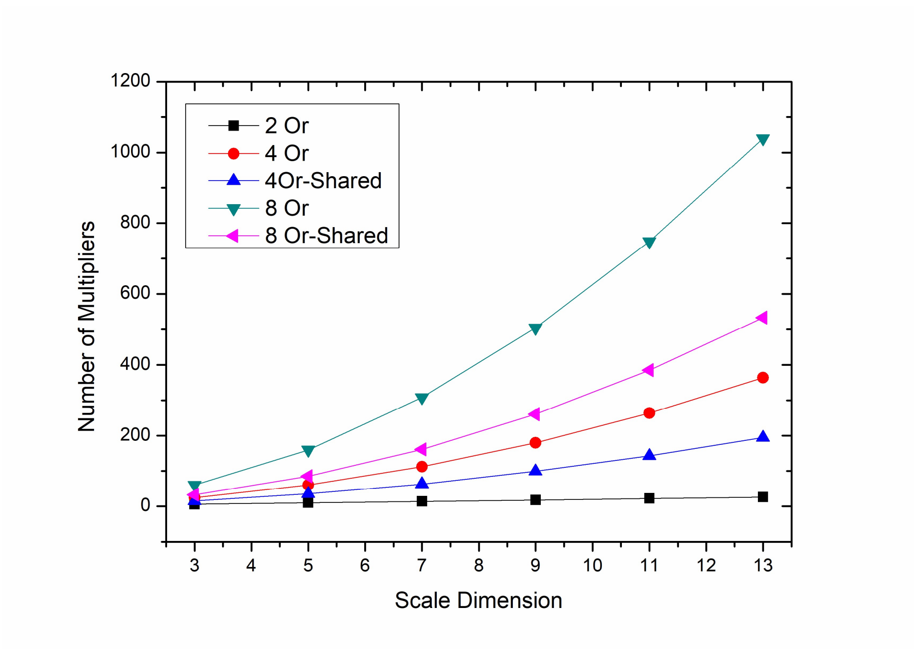


Figure IV.9 *Number of the total multiplier units, varying the filter dimensions.*

TABLE IV.5 *Adders and multipliers used in the considered designs with two scales.*

Orientations	Adders	Multipliers
2 Orientations	56	32
4 Orientations	568	292
4 Orientations Shared	312	162
8 Orientations	824	812
8 Orientations Shared	568	422

IV.5.4 Control Unit

The Control Unit is a finite state machine (FSM) capable of managing the correct transferring of the data to and from the Arithmetic Unit and its sub-units, through several control signals. The FSM also takes into account the case of processing near the image corners or edges, in order to correctly filter the image, juxtaposing null masks in these regions to complete the incoming tiles. Moreover, it synchronizes the used scales, which is fundamental to obtain all the filtered data related to the same pixel available at the input of the Max Pooling Unit at any beginning of a new clock period.

IV.5.5 Max Pooling Unit

The Max Pooling Unit compares the results coming from the Arithmetic Unit, computed for the various orientations and scales, and selects the greatest value among them

$$O = \max\{C_{1,0}, C_{1,\pi/8}, C_{1,\pi/4}, C_{1,3\pi/8}, C_{1,\pi/2}, C_{1,5\pi/8}, C_{1,3\pi/4}, C_{1,7\pi/8}, C_{2,0}, C_{2,\pi/8}, C_{2,\pi/4}, C_{2,3\pi/8}, C_{2,\pi/2}, C_{2,5\pi/8}, C_{2,3\pi/4}, C_{2,7\pi/8}\} \qquad \text{(IV.8)}$$

as a feature candidate to compose the final resulting image of the edge feature candidates.

IV.6 Synthesis and Results

The designs have been targeted to a Xilinx Virtex 7 XC7V2000tflg1925-1, as part of the proFPGA DUO ASIC prototyping board and to TSMC CMOS 90nm std_cells using Cadence Encounter RTL Synthesis tool. Synthesis results have been reported in Table IV.6, Table IV.7 and Table IV.8.

Now, it is possible to further justify the use of FI coding over FP32 one, due to its minor complexity. In fact, it is possible to state that from the developed HW designs a single FP32 multiplier would require 681 LUTs compared to the 126 LUTs used in the FI implementation, while a FP32 adder would require a 239 LUTs compared to the 77 LUTs for the FI implementation. From this consderations it is possible to say that a FP32 implementation of the Arithmetic Unit for the 4Or architecture, would approximately 334604 LUTs compared to the 59453 LUTs obtained using the chosen FI coding. It is worth to underline that the Xilinx Virtex 7 XC7V2000tflg1925-1 has 1221600 available LUTs, beeing a high-end board, but a FP32 design could not be implemented on lower end boards and moreover, an 8 Or design would be totally not implementable in a FP32 scenario. Meanwhile, the std_cell implementation of the same design using TSMC CMOS 90nm technology would require an area of approximately 13 mm^2, which is usually unacceptable for applications in which usually one of the main focuses is integration and resource savings.

TABLE IV.6 *Synthesis of the designs on FPGA platform using DSPs.*

	FPGA		
	With DSPs		
	2Or	4Or	4Or-Shared
Target platform	Virtex 7	Virtex 7	Virtex 7
LUTs	4022	27042	22281
FFs	2864	16220	11833
DSPs	32	260	146
Path Delay[ns]	5.60	6.40	10.00
Power [W]	0.775	1.629	1.748
fps	86	75	48
	8Or	8Or-Shared	Cesur et al.
Target platform	Virtex 7	Virtex 7	Stratix 4
LUTs	36604	27132	14025
FFs	24508	15249	20321
DSPs	316	174	202
Path Delay[ns]	5.60	11.20	6.73
Power [W]	1.446	1.493	--
fps	86	43	71

TABLE IV.7 *Synthesis of the designs on FPGA platform without using DSPs.*

	FPGA		
	Without DSPs		
	2Or	4Or	4Or-Shared
Target platform	Virtex 7	Virtex 7	Virtex 7
LUTs	8233	59930	41265
FFs	3440	21812	14912
DSPs	--	--	--
Path Delay[ns]	5.60	6.40	10.00
Power [W]	0.821	1.898	2.327
fps	86	75	48
	Without DSPs		
	8Or		8Or-Shared
Target platform	Virtex 7		Virtex 7
LUTs	76672		49298
FFs	30564		18566
DSPs	--		--
Path Delay[ns]	5.60		11.20
Power [W]	1.733		1.789
fps	86		43

TABLE IV.8 *Synthesis of the designs in std_cells.*

	Std_cells		
	2Or	4Or	4Or-Shared
Target platform	90 nm	90 nm	90 nm
Area [μm^2]	1002929	2414128	1801557
Delay[ns]	2.86	3.37	6.83
Power [mW]	14.013	76.453	27.135
fps	168	143	70

It is worth to underline that the reported power estimations are normalized at 100 MHz, while the fps performances are evaluated on Full-HD (1920x1080 pixels) image resolution.

From the above tables, it is possible to notice that the 2 Or architecture achieves the highest working frequency and the lowest power consumption and area occupation. However, it is the worst design in terms of accuracy, because it returns worse results in terms of edge detection. Thus this kind of implementation could not be used in applications where the accuracy is of primary concern. On the contrary, the 4 Or-Shared and the 8 Or-Shared implementations provide the same results of the 4 Or and 8 Or implementations in terms of accuracy, respectively. At the same time, using these designs it is possible to achieve great reductions in terms of area, while sacrificing the maximum working frequency of the overall system. In fact, the main drawback is represented by the higher delay when compared to the other solutions. However, considering that it exhibits a delay of 6.83 ns in std_cells and 10 ns in FPGA, real-time performances are achieved up to equivalent frame resolutions of 2209x2209 and 1825x1825 for the ASIC and FPGA respectively, all compatible with Full-HD standard. Therefore, considering the ADP product and the accuracy of the results, the shared solutions appear as the optimal choice, if there are no strict requirements in terms of power budget and area consumption.

In fact, the 4 Or-Shared implementation obtains a better edge detection compared to the 2 Or at the cost of a 79.6% area overhead and 93.6% increase in power consumption for an ASIC implementation. At the same time, the 4 Or-Shared provides the same accuracy of the 4 Or implementation, while reducing the area occupation of 25.4% and the power consumption of 64.5%. The 8 Or-Shared and the 8 Or implementation are not yet being developed and studies using std_cell technology, which would be object of further studies. However, from the FPGA data it is possible to develop some considerations to state what has to be expected in that case. These implementations would obtain a better edge detection compared to the 4 Or at the cost of an area overhead and greater power consumption, while the 8 Or-Shared provides the same accuracy of the 8 Or implementation reducing the area occupation.

The data have been obtained imposing the constraints on the PVT (Process, Voltage, Temperature) point as WORST process, HIGH voltage and HIGH temperature, to address the worst-case timing scenario.

IV.7 Conclusions

The proposed work develops the design of new ASIPs to implement the Gabor filter both for general purpose applications and medical imaging ones. The proposed designs are relevant for pre-processing applications in portable and resource-constrained devices.

Thanks to HW friendly implementations, state-of-the-art performances are achieved both on FPGA and std_cell implementations; in particular it has been possible to develop an 8 Orientation architecture capable of achieving better detection while obtaining acceptable ADP performances, which is a the only design doing so present in literature at the moment.

Future works could regard more studies on the proposed ASIC architecture towards its possible fabrication, while considering further developments to the overall architectures, in order to achieve better designs in terms of power and energy. To improve the design Distributed Arithmetic techniques could be used to further simplify the arithmetic circuitry, to explore the possibility further increasing the number of orientations and scales and to reduce the area occupation to make the ASIP more feasible for integration with image sensors [S. Venkatachalam, S.-B. Ko, 2017]. These developments are needed in order to make the design suitable for more resource-constrained applications and to achieve even better detection results. Finally, further developments could regard the implementation of the proposed designs on more shrunk technologies, in order to obtain better ADP performances and achieve the integration of the ASIP with the related image sensors.

Chapter V
Neural Networks in Multimedia Processing

V.1 Introduction

Since the last part of the research topics was devoted to VS and segmentation applications in multimedia processing, it seemed a natural consequence to shift the focus of the research also on the latest applications related to these techniques, represented by NN and NN dedicated HW accelerators. The interest in this kind of applications is a natural consequence of the research on Gabor filters applications; in fact, several implementations of Gabor filters are used as pre-filtering stages for NN or in combination with them to provide better results [E. Cesur, N. Yildiz, V. Tavsanoglu, 2012].

The aim of a NN is to recognize patterns in the input data; a pattern is defined as a particular arrangement of features or descriptors. At the same time it is possible to define pattern classes as families of patterns sharing some common characteristics. A generic pattern could be represented as a numeric vector and the recognition of a particular class starting from its features could be performed, for example, evaluating the most likelihood function or the Euclidean distance between the class vector and the obtained candidate vector. As an example, let’s consider the simple classification of different rectangles in an image. If we want to select among other rectangles only the ones having particular dimensions it is possible to associate to all the rectangles a point on two variables graph, which variables represent the width and the height of the particular rectangle. It is then possible to associate to the particular measurements we want to recognize a class, associated to a region in the two variables graph, and evaluate the distance of the sample rectangle from the class to evaluate if it belongs to it or not. This is a simple example, but with the relative modifications it is possible to extend this kind of considerations to fingerprints analysis, face recognition, gait analysis and several other problems which are nowadays more and more important in consumer electronics.

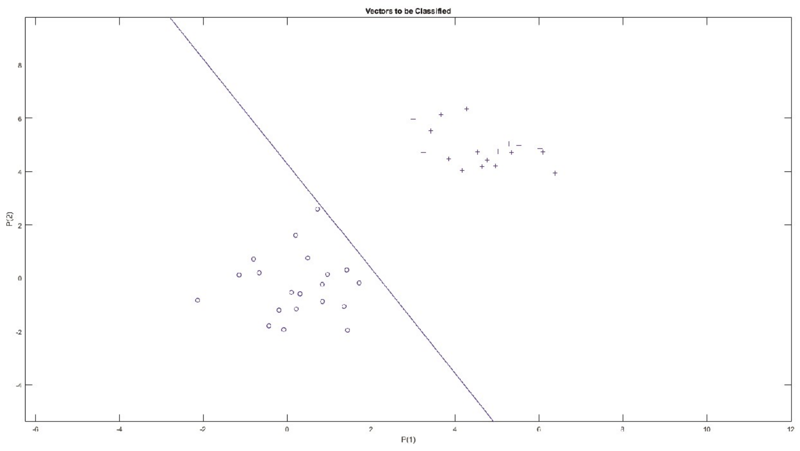


Figure V.1 *Example of simple region separation in pattern recognition.*

The recognition could be performed following mainly two different approaches: a decision-theoretic approach and a structural one. Structural techniques are based on relationships of the patterns shapes and are based on shape numbers, which are score functions taking into account the similarity of the shapes of the boundary regions, while decision-theoretic approach relies on the use of discriminate functions and quantitative evaluations, not considering the structural relationships of the pattern shapes. In the following only decision-theoretic approach will be taken into account, while structural techniques will not be further discussed.

If **x** represents a generic n-dimensional pattern vector, having K pattern classes, χ_1, χ_2, …, χ_K, the problem reduces to find K functions, s_1, s_2, …, s_K, such as that a pattern x is associated to the class χ_i if and only if

$$s_i(x) > s_j(x) \qquad j = 1,...,K; j \neq i \qquad \text{(V.1)}$$

Decision boundaries between the various contiguous regions are defined by the values in the plane for which the following equality is obtained

$$s_i(x) = s_j(x) \qquad \text{(V.2)}$$

An example of decision theoretic approach is matching, in which each class is represented by a model pattern vector; after that, any new input pattern is assigned to the class which it is closest in terms of a previously established metric. As previously stated, the simplest choice is the Euclidean distance metric, also known as minimum distance classifier, but, of course, it is not the only one possible (e.g. correlation derived matching).

The probability for a particular pattern to be associated to a particular class s_i is identified by $p(s_i|x)$; a misclassification of a pattern verifies whenever a pattern x belonging to the class s_i is associated to another class s_j, such loss in the pattern is indicated by L_{ij}. The conditional average risk indicates the average loss on all the considered classes

$$r_j(x) = \sum_{i=1}^{K} L_{ij} p(x | s_i) P(s_i) \tag{V.3}$$

It is possible to compare the various r_j(x) and assign every incoming pattern to the class having the minimum value in order to minimize the conditional average risk. Such a classifier is called Bayes classifier since it exploits the Bayes a posteriori probability. The classifier is optimal in the sense of minimizing the misclassification probability. However, this holds iif the probability density functions of patterns and the probability of occurrence of each class are known a priori. Due to these limitations, usually assumptions on the density functions and estimation of the parameters have to be made in order to use Bayes classifiers. The most common assumption is to consider the Gaussian probability density function for the density functions $p(x|s_i)$. In the general n-dimensional case the Gaussian density are specified by the mean vector, m_i, and the covariance matrix, C_j, approximated as

$$m_j(x) = \frac{1}{N_j} \sum_{x \in s_i} x \tag{V.4}$$

$$C_j(x) = \frac{1}{N_j} \sum_{x \in s_i} x x^T - m_j m_j^T \tag{V.5}$$

Moreover, since we are dealing with Gaussian functions it would be convenient to consider the logarithm of the Bayes decision functions instead of the Bayes decision function themselves. This obviously does not affect the result, since the logarithm function is monotonic and the values considered for the Bayes decision functions are all positive. Furthermore, the optimal classifier in a Bayes sense is defined by the following attributes:

- the pattern classes are described by Gaussian functions;
- all covariance matrices are equal to the identity matrix;
- all classes have the same probability to occur.

In this case the pattern classes we obtain are Gaussian clouds having the same shape in n-dimensions, which are usually called hyperspheres.

V.2 Neural Networks

The approaches seen so far use simple patterns and estimate simple statistical parameters for the different pattern classes. Sets of patterns are also known as training patterns, while the related sets of parameters are also known as training set; having a training set it is possible to obtain the set of decision relations defining the decision regions in the considered variable space, this process is known as training or learning. Basically, the training patterns are used to derive the parameters related to the decision function of each region. Obviously, after having defined the decision boundaries it is possible to estimate the goodness of the considered classifier by simply classifying actual patterns and verifying the correctness of the classification.

In this scenario neural networks are utilized to develop the training for the coefficients and vary the coefficients of the network in a way that the coefficients could be modified by any of the coefficients of the decision functions, in order to obtain the requested decision functions by successive adaptations of the coefficients. NNs are mainly composed by two components:

- the neurons, which are units calculating a given function of the inputs entering the neuron itself (which is usually a weighted average of the inputs which is then passed to non-linear activation function, usually a sigmoid, plus a certain bias);
- the connections between the neurons, which represent the coefficients for which multiply the inputs to the neuron and are the trainable coefficients described above. The training of the coefficients could be achieved using different algorithms (such like Gradient based and Back-propagation techniques), but particular attention has to be paid to avoid overfitting issues, which could worsen the classification performances of the NN.

A general base scheme of a NN is presented in Figure V.2. The units called hidden units are needed, together with back-propagation technique to obtain NNs capable of describing non linear functions [D. E. Rumelhart, G. E. Hinton, R. J. Williams, 1986].

The desired training algorithm has to derive a set of weights, *w*, and biases, *b*, capable of approximating the desired output *y*(*x*) for all the training inputs *x*. To do that, usually a cost function is derived, such that the vectors *w* and *b* which minimize the function are to be found

$$C(w,b) = \sum_x \|y(x) - a\|^2 \tag{V.6}$$

Then the problem becomes the minimization of a cost function which could be carried out using the gradient descent algorithm, well-known since the XIX century. The method consists in choosing a random starting point *x*, computing the derivatives of the function related to this point, using the gradient, choosing the minimum and negative value of the derivative, recalculating the derivative and repeat these steps until a minimum is reached. Obviously, we want to possibly reach the absolute minima of the function and not any random local minima; this could be achieved using

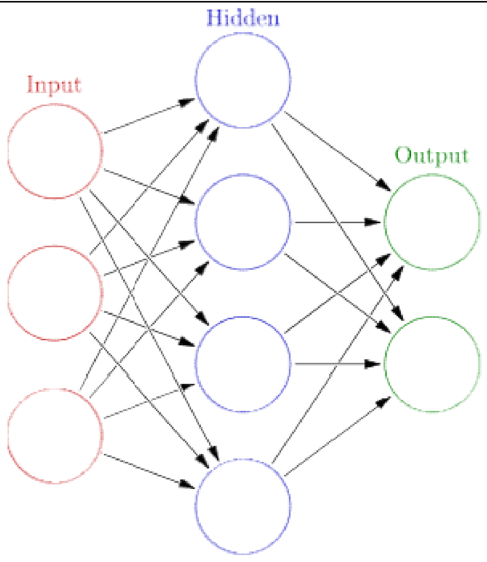


Figure V.2 *Neural Network base scheme [https://en.wikipedia.org/wiki/File:Colored_neural_network.svg].*

techniques like the stochastic gradient algorithm, in which the introduction of random noise allows to escape local minima, but in most problems that would not be an issue as stated in [Y. LeCun, Y. Bengio, G. Hinton, 2015].

Back-propagation algorithm is used to compute the gradient components, $\partial C/\partial w$ and $\partial C/\partial b$, needed in any gradient descent technique, in a relatively fast way, obtaining the training of the network in an acceptable time. As stated before, back-propagation was first developed in the '60s and '70s [A. E. Jr Bryson, Y.-C. Ho, 1969], [S. Linnainmaa, 1970] and then rediscovered and further developed in [D. E. Rumelhart, G. E. Hinton, R. J. Williams, 1986]. For back-propagation to work, the cost function need to satisfy two conditions:

- it can be written as an average over the cost functions obtained over individual training examples. This assumption is needed because the back-propagation algorithm calculates the partial derivatives on single training examples;
- it can be written as a function of the outputs of the NN. This means that the outputs are not function of a particular set of bias and weights, but only of the activations.

Hence, the back-propagation concerns about the variations in the weights and biases in order to optimize the cost function. We then define the error function as

$$\delta_i^l = \frac{\partial C}{\partial q_i^l} \tag{V.7}$$

where q_i^l represents the input to the generic i-th neuron of the l-th layer. Back-propagation is mainly developed through four fundamental equations:

$$\delta^L = \nabla_a C \circ \sigma' q^L \tag{V.8}$$

where $\circ$ represents the Hadamard product, a represents the activations, σ the activation functions and δ^L is the error of output layer;

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \circ \sigma' q^L \tag{V.9}$$

which relates the error in the considered layer to the one in the next layer;

$$\frac{\partial C}{\partial b_i^l} = \delta_i^l \tag{V.10}$$

$$\frac{\partial C}{\partial w_{ij}^l} = a_j^{l-1} \delta_i^l \tag{V.11}$$

two equations to consider the change rate of the cost function with respect to the biases and weights. What is fundamental about the back-propagation is the simultaneous calculation of all the partial derivatives, using just the results from the forward layer of the NN to develop the backward one, as can be noticed from the equations set.

To sum up, the resulting algorithm will then proceed as follows:

- forward propagation is calculated through the layers;
- the cost function is calculated;
- backward propagation is calculated and error functions for any neuron is evaluated;
- the weight's gradient is obtained multiplying the input activations and the calculated delta functions;
- the learning rate (defined as a percentage of the weight's gradient) is subtracted from the original weights.

This algorithm is then repeated until the NN does not reach an acceptable and desired performance.

Particular importance have multi layer NN, which are capable of learning complex, non linear and multidimensional tasks, resulting particularly appealing in speech, image and video recognition applications. In fact, in such applications the use of fully connected layers is not possible because of the high number of data to be processed, since starting from images of a few hundred pixels and considering one hundred hidden units in the network, one would already need several thousands of weights and furthermore this kind of NN does not show particular invariance to translations and distortions of the input signal; this will result in larger fully connected layer networks having the same performances of smaller non fully connected ones and also in longer training times [Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, 1998]. For these reasons, several NN configurations which avoid using only fully connected layers have been derived over the years as it is possible to notice in Figure V.3, each one having different performances for different applications. An important class of NN is represented by Convolutional Neural Networks (CNN), in which the main operation is represented by a convolution followed by addition of the biases and a threshold or sigmoid function for any layer. Varying the number and the type of layers or the number of neurons per layer will cause major changes to the performance of the overall network, as shown in [A. Krizhevsky, I. Sutskever, G. E. Hinton, 2012]. Furthermore, subsampling operations are developed since after finding a feature its absolute position in the original series of data becomes less important and this allows reducing the resolution of the feature maps and thus the number of convolutions and operations to be carried out by the NN and also the number of weights and coefficients to be considered in the layers following the subsampling operation. Thanks to all these characteristics, CNNs are capable of obtaining three main features: shift, scale and distortion invariance. These are achieved dividing the processing in smaller local blocks, which represent the convolution units, capable of processing the data related to a small portion of the previous layer to process and obtain local information; the operation is then repeated layer after layer in order to obtain more and more descriptive features from the combination of the ones obtained processing the previous layers.

In the following we present in more detail the architecture of one of the most used CNNs nowadays, AlexNet, and then present the developed work on weights coefficients coding using Vector Quantization (VQ) technique, for which an ad hoc HW decompression unit has been developed.

V.2.1 AlexNet

AlexNet was first presented in [A. Krizhevsky, I. Sutskever, G. E., Hinton, 2012] and it is one of the most studied CNNs since then. The CNN was developed to work on ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2010 dataset, which is composed by 1 million high resolution images classified in 1000 categories. While the ImageNet dataset is composed by images of different resolutions, AlexNet CNN was thought to work with 256x256 resolution images only, but it is important to highlight that no preprocessing has been done on the original images, but only cropping operations to select 256x256 regions of the image to be processed. The architecture of the network is shown in Figure V.4, in which the

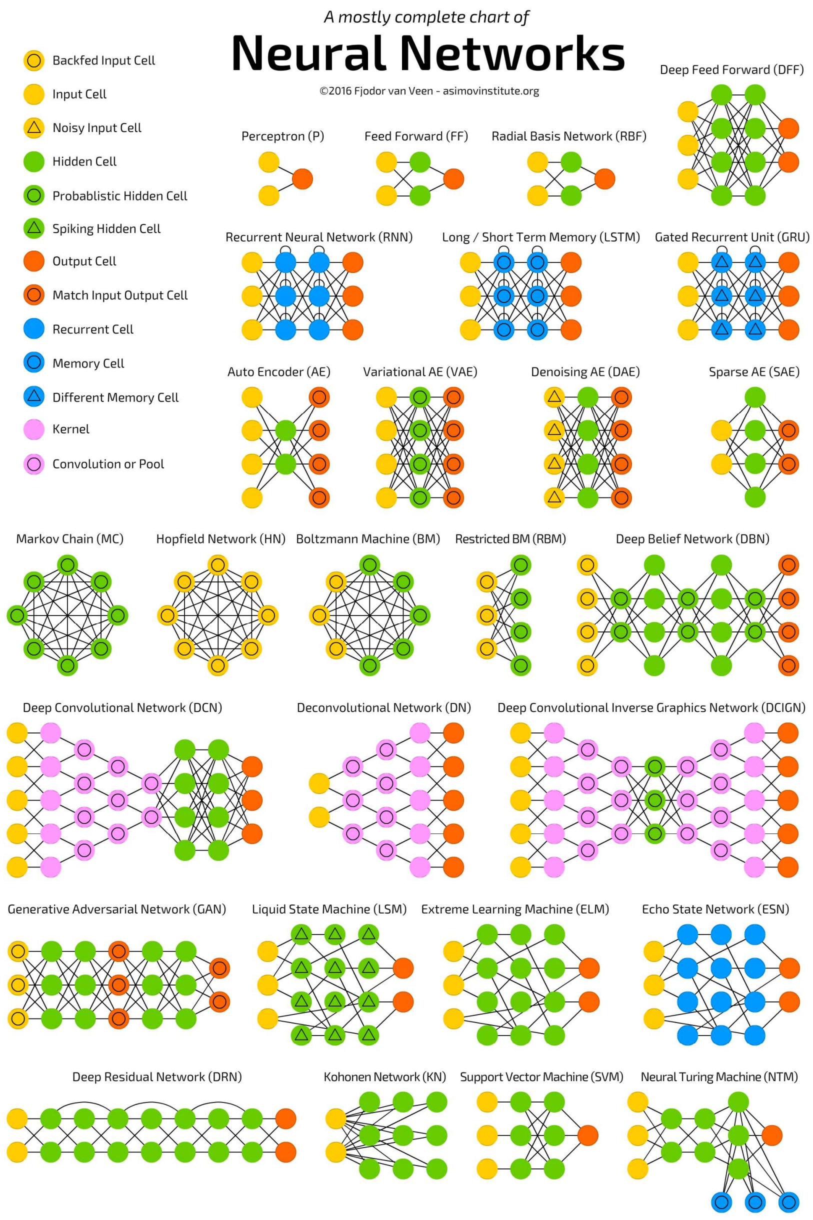

A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)

Figure V.3 *Several Neural Network developed schemes [http://www.asimovinstitute.org/neural-network-zoo/].*

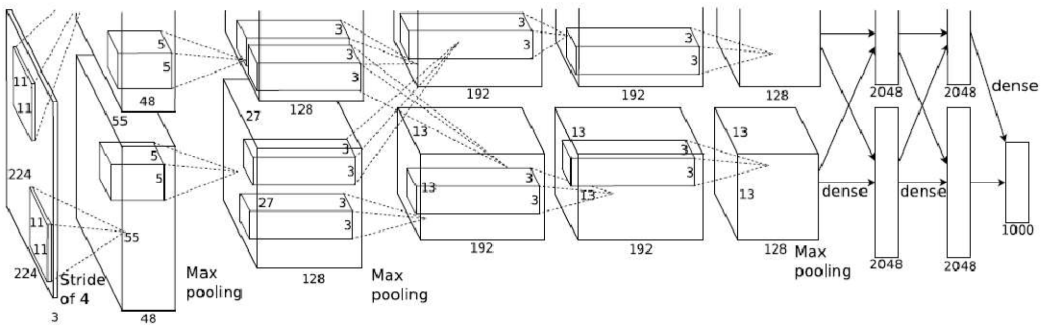


Figure V.4 *AlexNet architecture scheme [A. Krizhevsky, I. Sutskever, G. E. Hinton, 2012].*

different layers and their dimensions are highlighted. AlexNet is composed by five convolutional layers and three fully connected layers. The first five layers are the convolutional ones, followed by the fully connected and by a softmax unit producing the output distribution over the specified 1000 classes. Table V.1 sums up the size and the number of the various kernels used in AlexNet implementation, while the fully connected layer are made up by 4096 neurons each. Some of the kernels used by the CNN are reported in Figure V.5 However, the whole AlexNet architecture has 60 million different parameters.

TABLE V.1 *Size and number of kernels involved in AlexNet CNN.*

	Size	*Number*
Conv. Layer 1	11x11x3	96
Conv. Layer 2	5x5x48	256
Conv. Layer 3	3x3x256	384
Conv. Layer 4	3x3x192	384
Conv. Layer 5	3x3x192	256

It is important to notice that while usually the neurons' outputs are modeled using a *tanh*(·) form or in general a sigmoidal-like one, in AlexNet they are modeled using a simplified model known as Rectified Linear Unit (ReLU), which function is shown in Figure V.6 [V. Nair, G. E. Hinton, 2010]. The use of these saturated units allows obtaining a huge acceleration in the training phase and is also simpler to implement in HW than sigmoidal-like functions.

AlexNet represented the first successful application of CNNs on large datasets winning the first prize in ILSVRC 2012 obtaining a top-5 error rate of 15.3%, which represents the percentage of test examples for which the correct label is not in the first five results. The obtained results paved the way to a huge increase in efforts in CNNs, since the results represented a

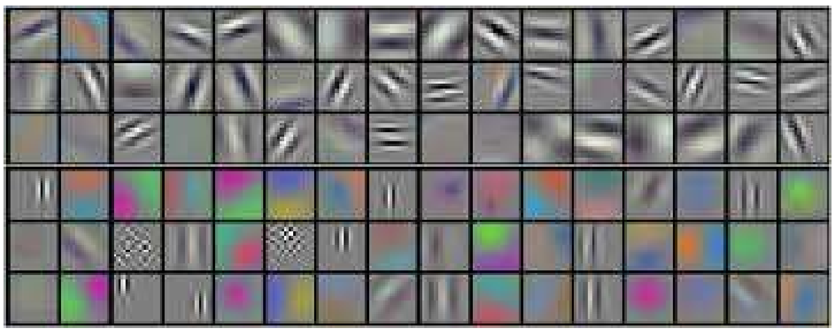


Figure V.5 *The 96 kernels used for the processing of the first convolutional layer in AlexNet*

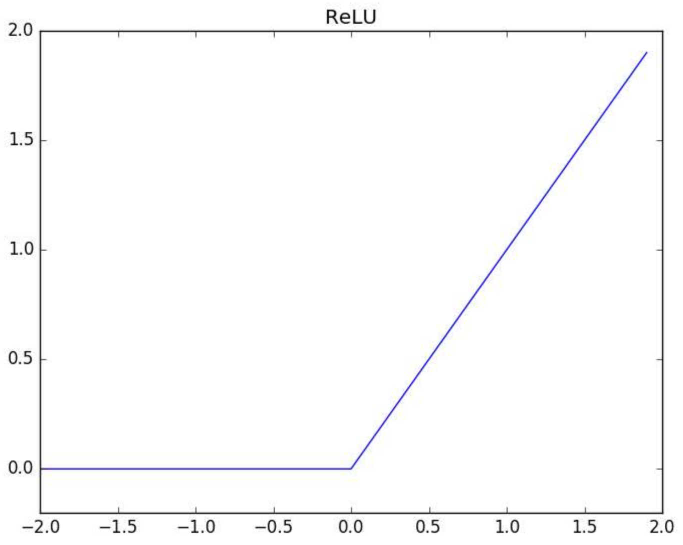


Figure V.6 *ReLU function graph.*

huge improvement on several considered datasets. Another important point was the proof of the importance to obtain deep neural networks, since, as stated in their work, even the removal of one single layer causes huge degradations in the overall results.

More and more systems aim to use CNNs and AlexNet-like NNs in HW and not only in SW or GPUs, in order to obtain better performances. This is due to the considerable increase in the number of operations to be carried out from the system. In fact, from the 60 millions parameters and 1 GOps of the AlexNet, we moved to systems having up to 150 million parameters (ResNet) and requiring up to 20 GOps (VGG19). One of the main problems in the use of these networks in HW is obviously represented by the amount of memory to be used, while obtaining compatible performances in terms of occupied area and power. One of the possible solutions for such a problem is to use VQ technique in order to compress the kernel representation and then decompress them when needed. The implementation of a decompression unit for VQ is the work developed by the candidate at STMicroelectronics, Cornaredo (MI) in the last year of his PhD, specifically made for low power NN applications.

V.3 Chip Implementation for CNNs

In [G. Desoli, V. Tomaselli, E. Plebani, G. Urlini, D. Pau, V. D'Alto, T. Majo, F. De Ambroggi, T. Boesch, S. Singh, E. Guidetti, N. Chawla, 2016], a new chip architecture specific for CNNs was presented. The use of CNNs in portable devices is more and more required for IoT applications; to use these architectures in handheld and portable device is fundamental to develop an ad hoc SoC, in order to obtain acceptable performances in terms of power consumption and efficiency. The SoC has also to be as much as possible reusable to implement different NN algorithms and solve problems as scalability, cost and bandwidth issues. The system has been developed in FD-SOI 28nm technology and is capable of working at frequencies up to 1 GHz with a power consumption of 6 µW/MHz at a voltage of 0.6 V. The SoC is made up by 8 DSP clusters (each one with a 16 KB cache, a 64 KB local RAM, a 64 KB shared RAM), 4 SRAM banks achieving a total memory of 4x16x64 KB and a coprocessor sub system.

In particular, the work has been focused on the development of a particular unit of the coprocessor, which represents the main unit of the entire design. The coprocessor is composed by 8 convolutional accelerators to develop the convolution operations, 16 Direct Memory Accesses (DMAs) stream engines for the data exchange among to and from different units, which allows to support dataflow based processing and could cast the data to multiple units at the same time, several units for the image treatment (e.g. for cropping, color conversions, coding, decoding and a corner detector), a decompression unit, sensor input interfaces, one microphone interface and a Digital Visual Interface (DVI) output interface. The connections of the various blocks of accelerator are reconfigurable at startup and at runtime, creating processing chains and dedicated connections for the data exchanges among the various units. The operations developed using the DMAs are then synched with the one carried out by the DSP clusters using ad hoc interrupts. In turn, this results in better power management and high flexibility if compared to classic data transfer structure like buses or HW data paths. The chip is capable of mapping the CNN like the AlexNet.

TABLE V.2 *Number of operations for any layer of AlexNet CNN.*

Layers	*Operation*	*Num. of Ops. [M]*	*Mem.*
Conv. Layer 1	Conv. 11x11 ReLU, Norm, Pool	105	35 K
Conv. Layer 2	Conv. 5x5 ReLU, Pool	223	307 K
Conv. Layer 3	Conv. 3x3 ReLU	149	884 K
Conv. Layer 4	Conv. 3x3 ReLU	224	649 K
Conv. Layer 5	Conv. 3x3 ReLU, Pool	74	442 K
Fully Conn. Layer 1		37	37 M
Fully Conn. Layer 2		16	16 M
Fully Conn. Layer 3		4	4 M
Total		**832**	**59.317 M**

In particular it is possible to make the chip develop the heavy calculations part, thus obtaining an optimization of the NN performances. In fact, the convolutions only represent almost the 85% of the total operations of any layer of the network in which convolution is involved. It is important to highlight that in this scenario the convolutional accelerators are capable of developing the work of 16 DSPs, while the other layers for ReLUs, pooling and normalization units could be carried out using other dedicated structures or DSPs. A detail of the number of operations for any layer of the AlexNet CNN is reported in Table V.2. The chip accounts for 2318 KB of on-chip SRAM for the weight parameters and 1436 KB for the feature maps, plus approximately 10 MB of external RAM dedicated to the fully connected layers. As it is possible to notice the on-chip memory is considerably smaller than the amount detailed in the above table; however, the various operations are developed in series, so the parameters and the feature maps are updated during the computation to develop the following required operations. The operations are thus developed in smaller batches both for the weights kernels and the feature maps, the partial results are obtained, accumulated and reused. In particular, as will be shown next, the total memory dedicated to the kernels could be reduced using a compression/decompression technique, in order to optimize the total data to be transferred to perform the calculations for any single batch. Moreover, the use of the DMAs allows executing the batches using different configurations according to the requirements to achieve serial or parallel batch processing on different DMAs at the same time, thus optimizing the performances in terms of power and bandwidth.

As stated before, it is possible to consider a compression/decompression operation to further improve the memory usage and the bandwidth during the batch operations. The compression of the weights could be performed in several ways, such as K-Nearest Neighbors (KNN), allowing a non linear reduction of the parameters to eight or fewer bits at the cost of an increase in the AlexNet top-1 error rate of 0.3%. The results at 200 MHz and 0.575 V as nominal operating conditions shows that the network can carry out all the operations needed for an AlexNet in 17.1 ms, developing approximately 1.3 GOps and consuming only 61 mW in the case of weights and features coded on 16 bits and only 41 mW in the case of weights and features coded on 8 bits, while processing one 227x227 pixel image.

Moreover, the chip is capable of implementing not only the operations needed for an AlexNet but also for other CNNs and has been already tested for applications like emotion detection, autonomous control for simulations (e.g. for videogames) and, of course, object recognition.

V.4 Vector Quantization Theory

The version of the chip presented before used Scalar Quantization (SQ) technique for the weights kernels, while the candidate developed a HW decoder unit to allow the use of VQ. In this section the theory related to SQ and VQ is analyzed before describing the developed HW unit.

V.4.1 Scalar Quantization

As stated in paragraph i.2, quantizing a series of data corresponds to associating the nearest approximating value taken from a finite set of determined fixed values as a response to a certain analog input. Scalar quantizers (so called because of the fact they are one-dimensional) could be defined as functions mapping a certain real input range into a finite range of *N* rational values called codebook, *C*, having a certain predefined size (which is usually related to the input range dynamic or the sensor specifications):

$$Q : \Re \longrightarrow C \qquad (V.12)$$

In this case the outputs of the quantizer are usually called output levels instead of codebooks. The resolution of the quantizer is given by $r = \log_2 N$ and it gives the number of bits needed for the quantization. For every word of the codebook it is possible to derive an associated partition of $\mathbb{R}$, in which any set of points associated to one of the *N* values of the codebook is called a cell, R_i, any R_i has no intersection with the other R_j, when $i \neq j$, and for which it is possible to write

$$R_i = \{x \in \Re : Q(x) = y_i\} = Q^{-1}(y_i) \qquad \text{(V.13)}$$

Moreover, the quantizer is defined as a regular one iff each cell is an interval and any output of the quantizer is internal to the interval, which means y_i is not a boundary point.

Every quantizer could be seen as composed by two main sub-units: an encoder, *E*, and a decoder, *D*. These two operations uniquely determines the quantizer, since the encoder is specified by the input partition, while the decoder is specified by the output values (the codebook), which is all that is needed for the quantizer description. The encoding operation relates the inputs to the index of the codebook, $\mathbb{R}\rightarrow I$, while the decoding one maps the obtained index to the related codebook, $I\rightarrow C$. It is possible to note how the project of the encoder and the decoder could be developed independently up to a certain point, and in particular that the decoder can be easily implemented using a look-up table procedure to store the codebook values.

The simplest form of the encoder is a selection form of the form

$$S_i(x) = \begin{cases} 1 & x \in R_i \\ 0 & otherwise \end{cases} \qquad \text{(V.14)}$$

and then produces the index to be sent to the decoder, which looks in the table of values and produces the *i*-th word of the codebook as output.

It is important to notice that due to the separation of the tasks it is possible to find an optimal encoder given the decoder and, viceversa, an optimal decoder given the encoder. The first problem is equivalent to finding the best possible partition given *C*. It is found that the best encoder is the one in which the obtained partition consists of all the inputs closer to the *i*-th word of the codebook than the other outputs; in other words, the best encoder given the decoder satisfies the nearest neighbor condition. On the contrary, fixing the encoder and wanting to obtain the best decoder, we obtain the so called centroid condition. The centroid condition minimizes the squared distortion measure, implying that the *i*-th element of the codebook is given by $y_i = E[X \mid X \in R_i]$given X a random variable representative of the input behavior; however, a more general centroid condition could be found to consider also other distortion measures for specific cases. From this

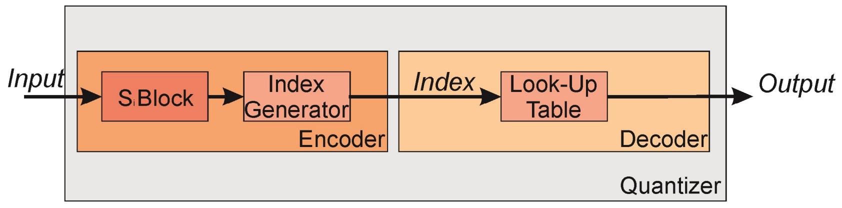


Figure V.7 *General block scheme of a quantizer unit, highlighting the encoder and decoder units.*

condition follows that in the case of a uniform input probability density function all the y_i will be the midpoint of the considered cell. However, it has to be underlined that these conditions cannot alone guarantee the optimality of the quantizer [S.P. Lloyd, 1982]; nevertheless, the Gaussian probability density function (pdf) satisfies the conditions for which these two conditions could be considered necessary and sufficient, meaning that in all the cases in which a Gaussain pdf could be used a optimal quantizer could be obtained.

Before moving on to the VQ, let us describe one of algorithms used to derive scalar quantizers. We describe this algorithm because, even if it was derived for SQ, it is possible to extend it to VQ problems and, hence, it results of particular interest for the present discussion. The algorithm was developed by Lloyd in 1957 and its main concept consists in taking a given codebook and starting from that obtain an improved one. The algorithm is iterative, so it will continue to improve the codebook until a suitable criterion is met. Having defined the average distortion of a codebook referred to a distortion measure $d(x, y)$ as $D=E[d(X, Q(X))]$ for the quantizer $Q(x)$, it is possible to first define the Lloyd iteration procedure and then the algorithm. The Lloyd iteration to improve the codebook implies to

- take a codebook $C_k=\{y_i\}$ and find the optimal partition into nearest neighbor cells;
- use the centroid condition to find the optimal codebook, C_{k+1}, related to the cells defined using nearest neighbor algorithm.

Then the algorithm could be described as follows

- an initial codebook C_1 is chosen and the number of iterations is set to $k=1$;
- from the codebook C_k Lloyd iteration is performed to obtain C_{k+1};
- the average distortion is calculated for the new C_{k+1}. If the difference between the distortions related to C_k and C_{k+1} is less than a certain threshold the algorithm ends, otherwise it take $k+1$ as the new k and repeat the operations from the second step.

Further details on SQ could be found in literature [R. M. Gray, 1992].

V.4.2 Vector Quantization

Vector Quantization represents an extension of SQ to a finite set of ordered values instead of a single one, allowing applying the concept of quantization to multiple dimensional data. It is important to underline that SQ could be considered as a particular case of VQ, considering monodimensional vectors. VQ finds application in complex digital signal processing, especially in cases in which the input signals are already inherently digital ones and a compression of the input set is desirable. This technique assumes particular importance in the case of pattern description, which could be well described using vectors and in applications in which the input data usually show a certain correlation, such as in the case of parts of speech waveforms or images. The compression due to the results of VQ can in these cases provide huge benefits to the further processing of the data, since a reduction in the number of bits of the operands causes a complexity reduction in the overall operation to be considered. As in the previous case, it is possible to state that a vector quantizer Q of size N and dimension m could be viewed as a function

$$Q:\Re^m \longrightarrow C \tag{V.15}$$

where C is the codebook $C = (y_1,\ldots,y_N)$ where any element of the codebook is a m-dimensional real vector. The number of bits needed per vector component is calculated as

$$r = (\log_2 N)/m \tag{V.16}$$

is known as the resolution of the quantizer and provides a measure of the maximum achievable accuracy, given that the codebook is well designed. The resolution is only related to the number of words in the codebook and the dimensionality of the input vectors, but it does not take into account the number of bits used for the representation of the codevectors, since those considerations are related to the architecture and to the correct description of the codebook, but not strictly to the quantizer itself, which is assumed to be described with an appropriate number of bits per codevector. Moreover, as before, to any element of the C set is associated a partition of $\mathbb{R}^m$ called cell, such that

$$R_i = \{x \in \Re^m : Q(x) = y_i\} = Q^{-1}(y_i) \tag{V.17}$$

Also the separation between encoder and decoder still applies for vector quantizers, since

$$E:\Re^m \rightarrow I \qquad \text{and} \qquad D: I \rightarrow \Re^m \tag{V.18}$$

where I is the set of indexes corresponding to the codebook elements and the encoder could be described as in equation (V.14).

A fundamental result about VQ is given by the following theorem, stating that a VQ developed coding will always be at least as good as any other [R. M. Gray, 1992]; in other words, there are no coding techniques which outperforms VQ coding in terms of bit rate and resolution.

Theorem: For any given coding system mapping a signal vector into one of N binary words and reconstructs the approximate vector from this binary word, there exists a vector quantizer with codebook size N giving the same reproduction as the given coding system for any input vector.

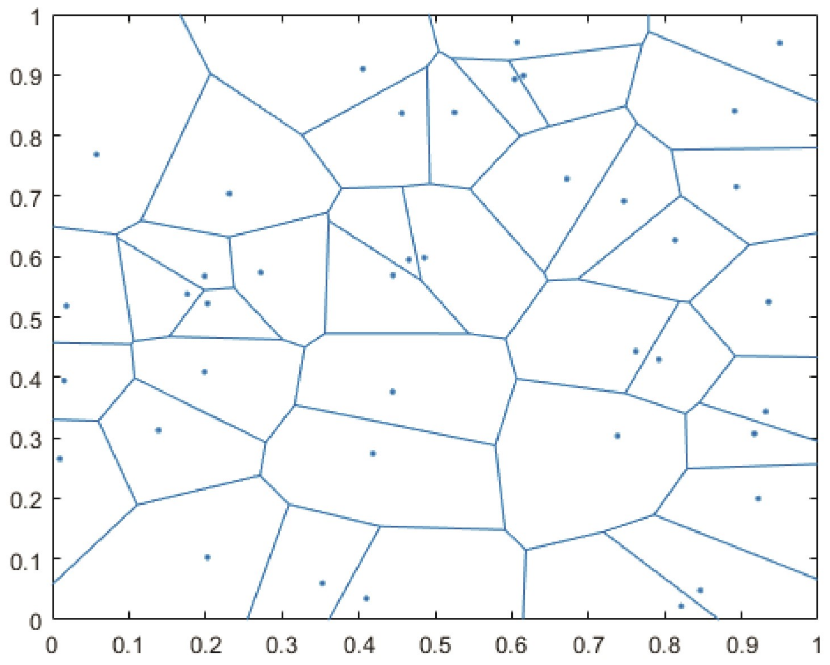


Figure V.8 *Example of representation of the cells obtained using VQ encoding on a 2D vector space.*

An example of the cell partition obtained using VQ on an arbitrary two dimensional input space is given in Figure V.8, where a regular vector quantizer is considered.

It is important to highlight that VQ works better in the representation of correlated data, since it is possible to derive Voronoi cells concentrating in the denser regions and obtain better lower distortions between the quantization result and the original data. A distortion measurement could be derived assigning a cost function, $d(x, \underline{x})$, associated to the quantizer input, x, and to the reproduction vector obtained at the exit of the quantizer, $\underline{x}$. It is then possible to quantify the quantizer performance using the average distortion given by

$$D = E[d(X, \underline{X})] \qquad \text{(V.19)}$$

which is the time average of the distortion function and which magnitude gives an indication of the performance of the overall system. Moreover, if the process is stationary and ergodic it is possible to state that not only an above limit to the (V.19) exists, but also that it equals the statistical expectation $d = D$. One of the most used measures of distortion is the well-known squared error also known as Euclidian distance and defined as

$$d(X,\underline{X}) = \|X - \underline{X}\|^2 = \sum_{i=1}^{k} \left(X_i - \underline{X_i}\right)^2 \qquad \text{(V.20)}$$

A special class of vector quantizers to be discussed is the nearest neighbor or Voronoi vector quantizers, which have the feature by the property of being completely characterized by means of the codebook and a distortion measure and which are optimal to minimize the average distortion. This in intrinsic in the Voronoi vector quantizer definition, since such a quantizer is defined as

$$R_i = \left\{x : d(x,y_i) \le d(x,y_j), \forall j \in I\right\} \qquad \text{(V.21)}$$

meaning that each cell is formed by all the points x showing a lower distortion if reproduced with y_i than with any other codevector of the codebook.

It is then possible to find an optimal vector quantizer starting from the given one and improving it iteratively; the first codebook to be used could be generated using different approaches (e.g. random coding). The optimization procedure is repeated until a vector quantizer sufficiently close to an optimal is obtained. It is important to notice that this procedure while not guaranteeing the optimality of the quantization, for which there are no closed form solutions, guarantees at least to reduce or leave unchanged the distortion measure iteration after iteration. The improvement of the starting codebook could be obtained using a generalized formulation of the previously presented Lloyd algorithm which is also known as k-means algorithm [J. MacQueen, 1967].

The generalized Lloyd iteration for the improvement of the codebook knowing the statistics is in this case given by the following steps:

- take a codebook C_l={y_i; i= 1,…, N} the nearest neighbor condition is used to find an optimal partition of the set into quantization cells. If ties occur in the evaluation of the distortion measure assign the considered point to the cell having the lowest index;
- using the centroid condition the next codebook is computed for the cells found in the previous step.

Then the algorithm could be described as follows, in which it is possible to notice that only the second step is changed compared to the previously presented in the non generalized Lloyd algorithm:

- take the initial codebook C_l and set l=1;
- from the codebook C_l Lloyd iteration is performed to obtain the improved codebook C_{l+1};

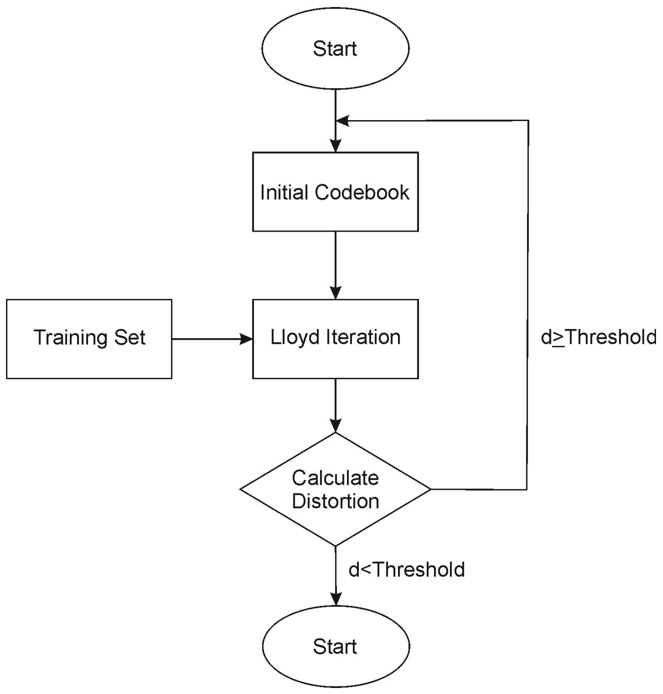


Figure V.9 *Flow chart of generalized Lloyd algorithm.*

- the average distortion is calculated for the new C_{l+1}. If the difference between the distortions related to C_l and C_{l+1} is less than a certain threshold the algorithm ends, otherwise it take l+1 as the new l and repeat the operations from the second step.

A flow chart of the algorithm is visible in Figure V.9. It is easy to notice from what has been said that the generalized Lloyd algorithm is a sort of descent algorithm, since each iteration reduce the average distortion and every iteration generates a new codebook, leading to a local minimum in the average distortion measure. Due to that, while the generalized Lloyd algorithm guarantees to stop in a local minimum it cannot guarantee to find the global minimum and thus the optimal codebook.

Another algorithm very used for the definition of the codebooks in VQ problems is the Linde-Buzo-Gray (LBG) algorithm [Y. Linde, A. Buzo, R. Gray, 1980]. This is an iterative algorithm which exploits a splitting method starting from an initial codebook C_0:

- the initial codevector is obtained calculating the average of the training sequence

$$c_{in} = \frac{1}{N}\sum_{n=1}^{N} X_n \qquad \text{(V.22)}$$

- this vector is split in two subvectors obtained multiplying the initial vector for two coefficients

  $$c_1^{(0)} = (1-\varepsilon)c_{in} \qquad c_2^{(0)} = (1+\varepsilon)c_{in} \qquad \text{(V.23)}$$

  where ε is a small positive threshold (usually assumed to have values around 0.001);
- the optimal codebook for the partial step is obtained using the Lloyd's method;
- the second and third steps are repeated on the new codebooks, increasing the number of codevectors after any iteration until the desired number of codebooks is obtained to finally obtain the desired optimized level quantizer.

The algorithm represents a method to form a good codebook starting from a single codevector or from a limited number of codevectors. It thus provides a method to produce a codebook from scratch starting from a training sequence and proceeding with successive approximations derived quantizers.

V.5 Proposed HW Decompression Unit

Several architectures exploiting VQ related results have been used during the years for speech and image recognition, but not too many works could be found on the use of VQ in neural network environment. However, two examples of the use of this technique on NNs could be found in [Z. Huang, X. Zhang, L. Chen, Y. Zhu, F. An, H. Wang, S. Feng, 2017], while a discussion on different compression methods to be used in NNs could be found in [Y. Gong, L. Liu, M. Yang, L. Bourdev, 2014]. In the proposed work we focused on the project and development of a decompression unit given that the required sub-optimal codebooks were already calculated using k-means nearest neighbor. We underline that in the following the compression steps have been separately conducted.

The developed unit will be part of the developments of the previously described chip architecture dedicated to NNs developed at STMicroelectronics. In a first implementation of the chip the technique used to reduce the burden of the weights to be memorized was a scalar quantization one. The structure is right now used for the decompression of the weight kernels only and it is used to mainly obtain memory optimizations and a more flexible structure to implement different NNs. The structure has to be capable of working both with 8 bits and 16 bits data and to manage different numbers of codevectors and codewords per codevector. In particular, for the different implementations a number of codewords from one to eight has been considered, where each codeword could be written in 8 or 16 bits format; it is important to note that the decompression factor is equal to the number of considered codewords per codevector and, in

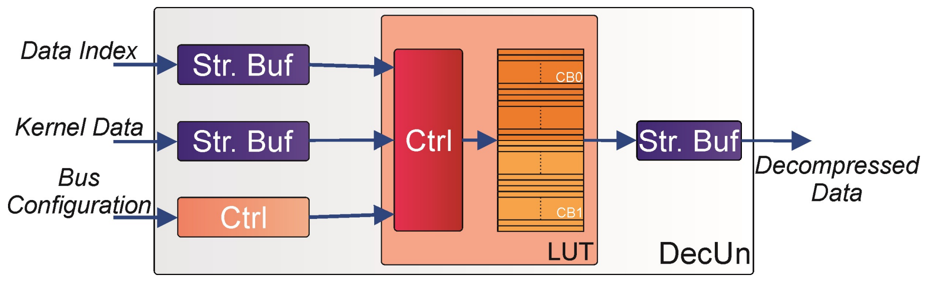


Figure V.10 *Block scheme of the proposed decompression unit.*

particular, the case of only one codeword corresponds to the use of a scalar quantizer. These considerations lead to consider a dimension for the single codeword that could vary from a minimum of 8 bits (in the case of codevectors composed by one codeword of 8 bits) to a maximum of 128 bits (in the case of eight codewords per codevector of 16 bits each). Moreover, the size of the considered codebook could be chosen differently according to the application; different sizes have been taken into account, from 16 up to 256 codevectors per codebook. Therefore, the structure has to provide different solutions to meet the needs of different applications, achieving flexibility, reusability and ensuring good performances in terms of working frequency, complexity and area occupation.

The main components composing the developed vector quantizer decompressor are:

- a memory module to store the codebooks;
- input/output streaming engines to provide data to the structure and read them from it;
- bus interfacing signals of the unit with the other units of the overall architecture;
- buffering structures for the correct management of the I/O streams;
- control and glue logic.

A general block scheme of the proposed architecture is presented in Figure V.10.

Since the proposed decompression unit is based on streaming inputs provided by dedicated stream engines, it could achieve better performances in terms of maximum working frequency and latency time if compared to a non-streaming architecture, which would require transmitting all the data over a bus, which would be also have a more complex arbitration. The data to be transmitted over the two streams are the indexes to access the codebook, namely the data to be decompressed, and the codevectors to be then read, which are organized in codebooks and written into a dedicated memory. The advantages of using streaming signals come with the drawback of the need of start and stop signals which have to be activated and deactivated in an event based fashion.

In the case of codevectors made up by more than 4 codewords it is possible to allow the allocation of the same codevector on two memory locations instead of a single one, exploiting the possibility of considering two following addresses as one, as described in V.11. The stream engine could provide a maximum of 24 bits per cycle and the codevectors could be formed by more than this number of bits. Since the stream engine cannot, in general, carry a single codevector per cycle to the unit, the stream carrying the codebook data is written in a buffering structure from which the data will be then transferred to the memory as soon as at least a codevector is ready to be written. In particular, in order to obtain a better management of the stream data, a circular buffer structure capable of 96 bits has been chosen, to allow the writing of new data while ensuring reasonable dimensions of the buffer. An example of the behavior of the circular buffer for this application is depicted in Figure V.12 for the writing of 4 codewords of 16 bits each. One problem in the development of the different considerations related to the NN operations is the latency time needed to write the first codebook to be read in the further processing. For this reason, in the developed unit, after the writing of the first codebook, cycles which are not used to read the memory are used to continue to perform writing operations, aiming to obtain a new codebook available to be read, before the operations on the first one are completed and thus reducing the latency time between one batch processing of a network layer and the other. Due to these considerations, the memory cut to be used has been chosen to be 1024x64 bits (*maximum number of codewords per codevector* x *maximum length of the data format* x *maximum number of codevectors* x 2 = 8x16x256x2 = 65536 bits). These choice guarantees enough space to manage the various cases to be dealing with, while also having a fitting geometry, since its dimensions are powers of two. A second buffer has then to be put after the output from the codebook, since the output stream carries out one codeword per cycle, on a maximum of 16 bits. It is worth to note as a special case that when considering codevectors made up by just one codeword, it would be not possible to write the second codebook while reading the first one, since the memory will be accessed in read mode at the beginning of each cycle. However, it is important to note that in case of necessity the codebook could be loaded also using the bus interface. The number of cycles between one reading and the next one increases with the number of words to be read, up to one read operation every 8 clock cycles for the case of 8 codewords of 16 bits each.

In order to correctly develop all the read and write operations, several control signals are needed. In particular, signals for the streaming management of the data to write and the index to read are needed to detect the start and end of the streaming frame, to correctly stop the streams without losing any data and, in general, to synchronize data coming from the

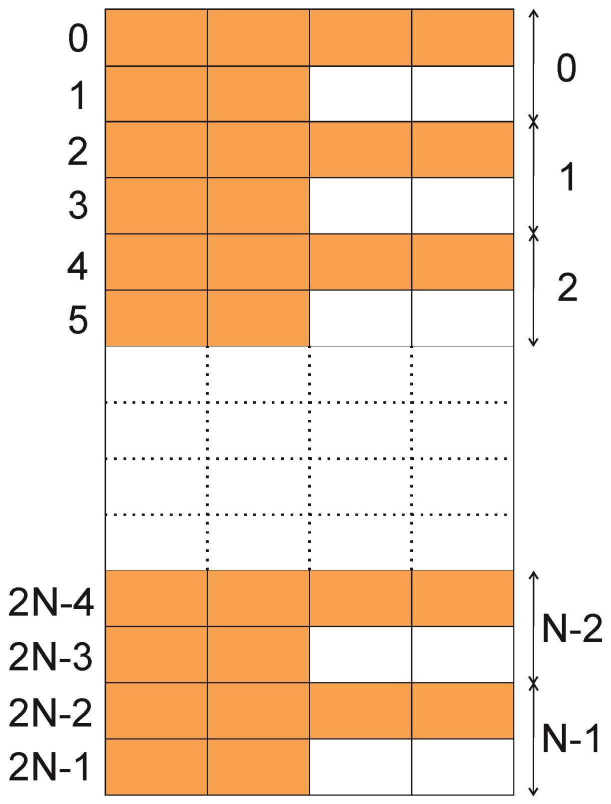


Figure V.11 *Example of reorganization of the look-up table addresses in the case of considering 6 codewords per codevector.*

streams, the buffers and the memory. Moreover, the data coming from the bus interface are used to configure the codebook read and write operations, establishing the number of codevectors and codewords to be used. From the discussion previously developed, it is possible to state that one bit is enough to set the data format of the kernels, which could assume only two values, while seven bits are needed to provide the number of codevectors per codebook ([1:256]) and other three are used for the number of codewords per codevector ([1:8]). Another important parameter is the number of batches to be processed with a single codebook. For this parameter we choose to use five bits, which allows taking into account a maximum of 31 batches on which to use the same codebook, while reserving one value for the case in which the codebook has to be used on a hypothetically infinite number of batches.

Other fundamental signals in the proposed unit are the counters needed for the correct writing of the previously described circular vector for all the different considered lengths of the codevectors to be written. In particular, at the end of obtaining a fast writing of the codebook without complicating the

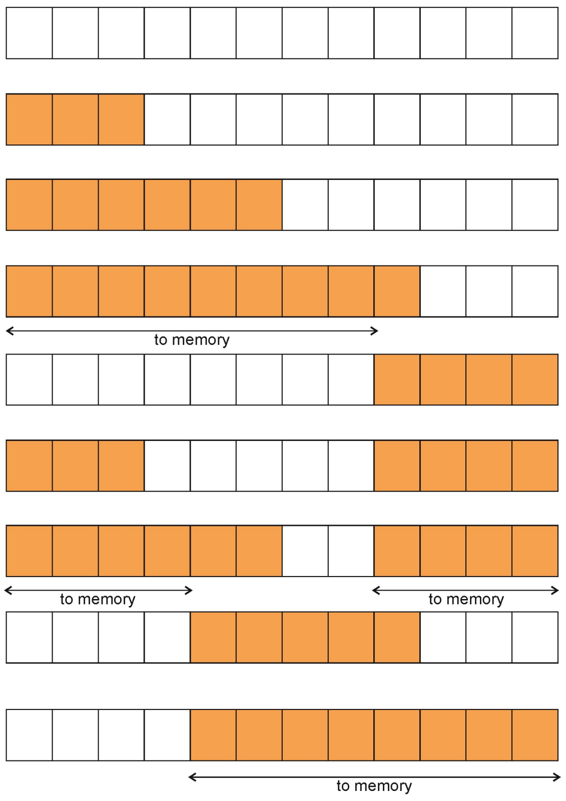


Figure V.12 *Example of use of the circular vector to write three codevectors made up by 4 codewords of 16 bits each; any vector location represents a byte, the colored locations are considered written.*

writing operation too much, we choose to reuse portions of the circular buffer before restarting the writing operation of the buffer and resetting the counters. This is done to speed up the writing of the codebooks in the memory.

Given these assumptions it is possible to summarize and describe the overall behavior of the units as it follows:

1. the incoming partial data, composing the weights kernel are stored on a circular buffer;
2. when a data packet stored on the partial buffer is enough to write one codevector, namely when

$$(wr - rd) \cdot 8 \geq f \cdot cw_{loc} \tag{V.24}$$

   where *wr* and *rd* represents the write and read pointers of the buffer, *f* is the format of the incoming weights (8/16 bits) and cw_{loc} is the number of considered codewords per memory address (which is maximum 4), the next look-up table location is written;

3. as soon as the first codebook has been written the decoding operations could begin; the codevectors are read and stored in a partial buffer from which they are then sent as output one codeword per clock cycle;
4. if it is possible to interleave write operations to the reading ones, a new codebook is written while reading the first one, obtaining a lower latency time for the subsequent batch processing;
5. the reading operation is repeated until the current batch of frames is completely decompressed;
6. the next codebook becomes active and the processing restart from step 4. until no further frame has to be processed and the unit output goes in idle state.

Several tests were led on the unit to achieve a thorough understanding of its functioning and to take into account all the various test cases, considering different combinations of the parameters determining the structure behavior. In particular automated tests to check out the correctness of the writing operation to the memory and of the readings from it were conducted for all the combination of the parameters. A first evaluation of the tests has been conducted checking the outputs corresponding to the incoming data and roughly verifying the correctness of the unit behavior. Then the same tests have been run on the same structure working together with other units in the top architecture and the results have been compared to the ones previously obtained. The two tests gave the same results to the same inputs, which together with the previously developed evaluations ensure the correct behavior of the decompression unit for any combination of the setting parameters.

The structure has been developed for FPGA prototyping and in 40nm technology obtaining a maximum operating frequency of 650 MHz, meaning that when the codebook is ready, $6.5 \cdot 10^8$ weights per second could be provided to the subsequent calculation units. Moreover, due to the simplicity of the developed architecture the area occupation of the structure could be considered negligible in almost every relevant NN applications. In fact, not taking into account the memory, the logic design needs only 13 kgates, while an area occupation six times higher is obtained considering also the memory storing the codebooks.

V.6 Conclusions

The proposed work develops the design of a HW accelerator dedicated to the decompression unit to implement part of a quantizer scheme based on VQ technique to be used in a chip architecture dedicated to NNs acceleration. The design is relevant for the pre-processing of data in applications dedicated to the filtering of data and thus very useful for convolutional NNs. The development of the unit has been done in a HW friendly fashion, in order to reduce latency times and achieve better working frequencies and low area occupation. In particular, we managed to obtain a structure capable of managing codevectors composed by a number of codewords from one to eight and having different formats, which allows covering several cases useful in HW accelerators dedicated to NN framework.

Future works could regard further integration of the proposed architecture with other HW accelerator units for NN data processing, together with the evaluation of related compression techniques in order to further improve the actual performance of the developed accelerator.

Chapter VI
Conclusions

The aim of the thesis work was to project and develop HW accelerators for fast computing in multimedia applications. The starting point for the research has been the research and development on new multiplier structures to improve the performances of FP32 multipliers involved in image and applications aiming to high resolutions and devoted to FP processing, such like HDR filtering related ones. The researches resulted in a multiplier structure based on adder chains and normalization units instead then FP32 multipliers, fit for applications dealing with fixed filters and aiming to high precisions for further data processing. The methodology is based on operands partitions derived from DA theory; it allows deriving compact and low power consuming circuit to improve the calculation of filtering operations, mainly on MAC and convolution operations. The burden of FP32 devoted circuits is substituted using simplified adders and memory structures for storing pre-multiplied coefficients. A second implementation of this solution has been derived in the case of applications working on memory constrained devices.

Starting from the developed considerations about high precision MAC units, we improved an existing LDR2HDR processor to make it work on different resolutions input images, up to 4K UHDTV. Hence, the structure is more reusable and flexible if compared to the previous versions of the same architecture and implement a streaming elaboration of input data for on-the-fly conversion of LDR images without using any frame buffer for the partially elaborated data and reducing the dimensions of the other buffers and memory modules. Moreover, the developed ASIP use a reduced amount of SRAM instead of off-chip DRAM, enabling the coupling of the developed processor with the sensors. Finally, the structure is scalable and reconfigurable to set the area/speed ratio performances exploiting the separability property of the two-dimensional Gaussian kernel, which is largely employed in image and video elaboration methods and a careful HW project. The developed studies allow us to say that it would be possible to develop dedicated ICs for HDR2LDR conversion to achieve higher operating frequencies, lower power dissipations and real-time performances even in the case of 4K UHDTV image processing. Still, in this kind of applications the time related to the transferring of the data to and from the memories represents a huge issue.

To kick the latter problem related to the memory structures in multimedia applications, a new SA scheme for SRAMs has been proposed. The main feature of the proposed scheme is a better offset rejection if compared to other circuits in literature aiming to the same result, while also reducing the sensing delay, allowing faster writing and reading operations and keeping contained area requirements. An IC has been developed and realized in the development of the project and further studies will regard real case results and tests, while understanding how further improvements of the same could be achieved. Hence, the desired project allow the development of faster ASICs, especially the ones presenting memory accessing constrains, in which the high data transferring rate to and from the memories represents the bottleneck.

Then the research focused back on filtering structures, reconnecting to the previous projects. In particular, the design of new ASIPs implementing Gabor filters has been developed to pre-process images in portable and resource-constrained devices. Thanks to HW optimizations and a careful design of the sub-units composing the overall architecture it has been possible to develop several designs using 2, 4 and 8 orientations of the Gabor filters and capable of achieving acceptable detection while obtaining acceptable ADP performances and trade-offs for the different applications. Further developments are needed to obtain more fitting architectures for resource constrained devices and achieve even more accurate detections. In particular, it would be interesting to develop further analysis dedicated to the ASIC implementation of the proposed designs, in order to gain more insight towards a better description of a dedicated chip to be used in edge detection, segmentation and VS applications.

Finally, the previously developed knowledge has been used to develop part of a quantizer structure to be used in NN context, capable of obtaining a decompression scheme dedicated to the weight kernels of the NN, exploiting VQ technique. The architecture is part of a more complex HW framework to accelerate NN processing and calculations, in which it allows developing better pre-processing of the data for applications mainly devoted to filtering using convolution. The unit has been developed in HW friendly fashion, to obtain a good area occupation and lower design latency times, while also achieving higher maximum operating frequencies. The structure is flexible, in the sense that it can work on several codebook and codevector structures, having different number of codevectors, codewords for codevector and data format, allowing to be reusable to implement different NNs schemes. These features also allow using the unit in combination with different HW accelerator units working on data on which a compression/decompression operation would be beneficial.

The interests of the candidate during the PhD followed the developing of HW filters designs, starting from the simplest operations which could be found in such structures and moving towards the research on NN HW accelerators, which represents, nowadays, one of the most active and interesting research field in digital design and circuits and systems developments. Even if several results have been achieved, the presented designs could be further developed and improved and it is possible to find different developments needed especially for the developments of suitable ASICs, which would represent the goal of a HW design project and which has been developed, in part, only for the SA and decompression schemes. The candidate aim to further develop these researches in the near future, if possible, as well as to further develop his knowledge on NNs, at the end of achieving HW improvements through a better knowledge of digital designs dedicated to artificial intelligence.

Bibliography

Ahn-Tuan D., Zhi-Hui K., Kiat-Seng Y., “Hybrid-Mode SRAM Sense Amplifiers: New Approach on Transistor”, IEEE *Trans. Circuits and Systems II: Express Briefs*, 2008

Aksoy L., Flores P., Monteiro J., “Efficient design of FIR filters using hybrid multiple constant multiplication on FPGA”, *Computer Design (ICCD)*, 2014.

Akyüz A. O., Fleming R., Riecke B. E., Reinhard E., Bülthoff H. H., “Do HDR displays support LDR content?: A psychophysical evaluation”, *ACM Trans. Graphics*, 2007

ANSI/IEEE754 – 1985, “IEEE Standard for binary floating - point arithmetic”, 1985.

Arish S., Sharma R. K., “An efficient floating point multiplier design for high speed applications using Karatsuba algorithm and Urdvha - Tyriagbhyam algorithm”, *Signal Processing and Communication (ICSC), 2015 International Conference on*, 2015.

Banterle F., Artusi A., Debattista K., Chalmers A., “Advanced High Dynamic Range Imaging: Theory and Practice”, *CRC Press*, 2011.

Banterle F., Chalmers A., Scopigno R., “Real-time high fidelity inverse tone mapping for low dynamic range content”, *Proc. Eurographics Short Papers*, 2013.

Banterle F., Ledda P., Debattista K., Bloj M., Artusi A., Chalmers A., “A psychophysical evaluation of inverse tone mapping techniques”, *Computer Garphics Forum*, 2009

Banterle F., Ledda P., Debattista K., Chalmers A., “Expanding low dynamic range videos for high dynamic range applications”, *Proc. SCCG*, 2008.

Basiri M. A., Sk N. M., “An efficient hardware - based higher radix floating point MAC design”, *ACM Trans. on Design Automation of Electronic Systems*, 2014.

Bhatia P., Reniwal B.S., Vishvakarma S.K., “An offset-tolerant self-correcting sense amplifier for robust high speed SRAM”, *19th Symp. On VDAT*, 2015.

Bhavnagarwala A.J., Tang. X., Mandl J.D., “The impact of intrinsic device fluctuation on CMOS SRAM cell stability”, *IEEE J. of Solid-State Circuits*, 2001.

Berkeman A., Owall V., M. Torkelson, “A low logic depth complex multiplier using distributed arithmetic”, *IEEE J. Solid- State Circuits*, 2000.

Bryson A. E. Jr, Ho Y.-C., “Applied Optimal Control”, *Blaisdell Publishing Co.,* 1969.

Cesur E., Yildiz N., Tavsanoglu V., "An Improved FPGA Implementation of CNN Gabor-Type Filters", *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, 2011.

Cesur E., Yildiz N., Tavsanoglu V., "On an Improved FPGA Implementation of CNN-Based Gabor-Type *Filters", Circuits and Systems II: Express Briefs, IEEE Transactions on*, 2012.

Chalmers A., Debattista K., "HDR video past, present and future: a perspective", *Signal Processing: Image Communication*, 2017.

Chandoke N., Chitkara N., Grover A., "Comparative analysis of Sense Amplifiers for SRAM in 65nm CMOS technology", *Electrical, Computer and Communication Technologies (ICECCT)*, 2015.

Cho Y.C.P., Chandramoorthy N., Irick K.M., Narayanan V., "Multiresolution Gabor Feature Extraction for Real Time Applications", *2012 IEEE Workshop on Signal Processing Systems*, 2012.

Cooley, James W., and John W. Tukey. "An algorithm for the machine calculation of complex Fourier series" *Mathematics of computation*, 1965.

Daugman J. G., "Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters," *Journal of Optical Society of America A*, 1985.

Debevec P. E., Malik J., Recovering high dynamic range radiance maps from photographs", *Proc. SIGGRAPH*, 1997.

Desoli G., Tomaselli V., Plebani E., G. Urlini, D. Pau, V. D'Alto, T. Majo, F. De Ambroggi, T. Boesch, S. Singh, E. Guidetti, N. Chawla, "The Orlando Project: A 28 nm FD-SOI Low Memory Embedded Neural Network ASIC", *International Conference on Advanced Concepts for Intelligent Vision Systems*, 2016.

Durand F., Dorsey J., "Fast Bilateral Filtering for the Display of High-dynamic-range Images", *ACM Trans. on Graphics*, 2002.

Ergen B., Çinar A., Aydin G., "Gabor wavelet and unsupervised Fuzzy C-means clustering for edge detection of medical images," *Innovations in Intelligent Systems and Applications (INISTA), 2012 International Symposium on*, 2012.

Gong Y., Liu L., Yang M., Bourdev L., "Compressing Deep Convolutional Networks using Vector Quantization", *Computer Vision and Pattern Recognition*, 2014.

Gonzalez R. C., Richard E. Woods, "Digital Image Processing", 3rd Edition, *Prentice Hall*, 2007.

Gray R. M., "Vector Quantization and Signal Compression", *The Springer International Series in Engineering and Computer Science*, 1992.

Guo J.M., Prasetyo H., Wong K., "Vehicle Verification Using Gabor Filter Magnitude with Gamma Distribution Modeling", *IEEE Signal Processing Letters*, 2014.

Hardy G. H., Wright E. M., “An introduction to the theory of numbers (sixth edition)”, *Oxford University Press*, 2008.

Hewlitt R. M., Swartzlantler E. S., “Canonical Signed Digit representation for FIR digital filters”, *Signal Processing Systems, 2000, IEEE Workshop on*, 2000.

http://www.asimovinstitute.org/neural-network-zoo/

http://www.atnf.csiro.au/people/lop009/multiwave.html

https://homepages.cae.wisc.edu/~ece533/images/

https://en.wikipedia.org/wiki/File:Colored_neural_network.svg

https://it.wikipedia.org/wiki/File:Digital_video_resolutions_(VCD_to_4K).svg

https://en.wikipedia.org/wiki/Static_random-access_memory#/media/File:SRAM_Cell_(6_Transistors).svg

http://resizing.info/p2s.html

https://www.isi.uu.nl/Research/Databases/DRIVE/

Hu H., “Enhanced Gabor Feature Based Classification Using a Regularized Locally Tensor Discriminant Model for Multiview Gait Recognition”, *Pattern Analysis and Machine Intelligence,IEEE Transactions on*, 2013.

Huang Z., Zhang X., Chen L., Zhu Y., An F., Wang H., Feng S., “A vector-quantization compressor circuit with on-chip learning ability for high-speed image sensor”, *IEEE Access*, 2017

Humpire-Mamani G., Traina A.J.M., Traina C., “k-Gabor: A new feature extraction method for medical images providing internal analysis”, *Computer-Based Medical Systems (CBMS), 2012 25th International Symposium on*, June 2012.

Huo Y., Yang F., Brost V., “Dodging and burning inspired inverse tone mapping algorithm”, *Journal of Computational Information Systems*, 2013.

Jeong H., Park J., Oh T.W., Rim W., Song T., Kim G. et al., “Bit-line Precharging and Preamplifying Switching PMOS for High-Speed Low-Power SRAM”, *IEEE Trans. Circuits and Systems II: Express Briefs*, 2016.

Jiang W., Lam K.-M., T. Z. Shen, “Efficient Edge Detection Using Simplified Gabor Wavelets”, *Man, and Cybernetics, Part B (Cybernetics), IEEE Transactions on Systems*, 2009.

Kamarainen J.-K., Kyrki V., Kälviäinen H., “Robustness of Gabor Feature Parameter Selection,” *Proceedings of the IAPR Workshop on Machine Vision Applications*, 2002.

Koenderink J.J., “The structure of images”, *Biol. Cybern. 50, 363-370*, 1984.

Kovaleski R. P., Oliveira M. M., “High-quality brightness enhancement functions for real-time reverse tone mapping”, *Visual Computer*, 2009.

Krizhevsky A., Sutskever I., Hinton G. E., “ImageNet Classification with Deep Convolutional Neural Networks”, *Advance in Neural Information Processing Systems*, 2012.

Kwolek B., “Face Detection Using Convolutional Neural Networks and Gabor Filters”, *Artificial Neural Networks: Biological Inspirations – ICANN 2005*, 2005.

Licciardo G. D., D'Arienzo A., Rubino A., “Stream processor for real - time inverse Tone Mapping of Full - HD images”, *IEEE Trans. on VLSI Systems*, 2015.

Licciardo G.D., Boesch T., Pau D. and Di Benedetto L., “Frame Buffer-less Stream Processor for Accurate Real-Time Interest Point Detection,” *Integration, the VLSI Journal*, 2016.

LeCun Y., Bengio Y., Hinton G., “Deep Learning”, *Nature*, 2015.

LeCun Y., Bottou L., Bengio Y., Haffner P., “Gradient-based learning applied to document recognition”, *Proceedings of IEEE*, 1998.

Linde Y., Buzo A., Gray R., “An Algorithm for Vector Quantizer Design”, *IEEE Transactions on Communications,* 1980.

Lindeberg T., “Scale-space theory: A basic tool foranalysing structures at different scales”, *Journal of Applied Statistics,* 1994.

Linnainmaa S., “The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors”, 1970.

Lloyd S. P., “Least squares quantization in PCM”, *IEEE transactions on information theory*, 1982.

Lu C.-Y., Jing B.-Z., Chan P.P.K., Xiang D., Xie W., et al., “Vessel enhancement of low quality fundus image using mathematical morphology and combination of Gabor and matched filter”, *Wavelet Analysis and Pattern Recognition (ICWAPR), 2016 International Conference on*, 2016.

Masia B., Agustin S., Fleming R. W., Sorkine O., Gutierrez D., “Evaluation of reverse tone mapping through varying exposure conditions”, *ACM Trans. Graphics* ,2009.

MacQueen J., “Some methods for classificiation and analysis fo multivariate observations”, *Proceedings of the Fifth Berkeley Symposium on Mathematics Statistics and Probablity*,1967

Meher P.K., “New approach to Look - up - table design and Memory - based realization of FIR digital filter”, *Circuits and Systems - I: Regular Papers, IEEE Trans. on*, 2009.

Mehrotra R., Namuduri K. R., Ranganathan R., “Gabor Filter-based Edge Detection,” *Pattern Recognition,* 1992.

Mikolajczyk, K., “Detection of local features invariant to affine transformations”, *Ph.D. thesis, Institut National Polytechnique de Grenoble, France*, 2002.

Mikolajczyk K., Schmid C., “A performance evaluation of localdescriptors,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005.

Myszkowski K., Mantiuk R., Krawczyk G., “High Dynamic Range Video”, *Morgan & Claypool*, 2008.

Nair V., Hinton G. E., "Rectified linear units improve restricted Boltzmann machines" *Proc. 27th International Conference on Machine Learning*, 2010.

Nassif S.R., "Modeling and analysis of manufacturing variations", *IEEE Custom Integrated Circuit Conf.*, 2001.

O'Shea E., "Bachet's problem: as few weights to weigh them all", *arXiv: 1010.5486*, 2008.

Parhi K. K., "VLSI Signal Processing Systems: Design and Implementation", *John Wiley & Sons*, 2007.

Park S. K., "The r-complete partitions", *Discrete Mathematics*, 1998.

Paul B. C., Fujita S., Okajima M., "ROM - Based Logic (RBL) design: a low - power 16 bit multiplier", *IEEE Journal of Solid - State Circuits*, 2009.

Peled A., Liu B., "A New Hardware Realization of Digital Filters*", IEEE Transactions on A.S.S.P.*, 1974.

Pellegrino F., Vanzella W., Torre V., "Edge Detection Revisited,"*Systems, Man, and Cybernetics – Part B: Cybernetics*, *IEEE Trans.on,* 2004.

Reinhard E., Ward G., Pattanaik S., Debevec P, "High Dynamic Range Imaging: Acquisition, Display and Image-based Lighting", 2nd Edition, *Elsevier*, 2006.

Rempel A. G., Trentacoste M., Seetzen H., Young H. D., Heidrich W., Whitehead L., Ward G., "Ldr2Hdr: On-the-fly reverse tone mapping of legacy video and photographs", *ACM Trans. Graphics*, 2007.

Rødseth Ø. J., "Enumeration of M-partitions", *Discrete Mathematics*, 2006.

Rosenfeld A., Kak A.C., "Digital Picture Processing", *Academic Press, 2 Ed.*, 1982.

Rumelhart D. E., Hinton G. E., Williams R. J., "Learning Representations by Back-Propagating Errors", *Nature,* 1986.

Seevinck E., Beers P.J.V., Ontrop H., "Current-mode techniques for high-speed VLSI circuits with application to current SA for CMOS SRAM's," *IEEE J. Solid-State Circuits*, 1991.

Shah J.S., Nairn D., Sachdev M. "An energy-efficient offset-cancelling sense amplifier", *IEEE Trans. Circuits and Systems II: Express Briefs* 2013.

Sharifkhani M., Rahiminejad E., Jahinuzzaman S.M., Sachdev M. "A Compact Hybrid Current/Voltage Sense Amplifier With Offset Cancellation for High-Speed SRAMs", *IEEE Trans. On VLSI Systems*, 2011.

Själander M., Larsson - Edefors P., "Multiplication acceleration through twin precision", *IEEE Trans. on VLSI Systems*, 2009.

Tesler L. G., "Optimal denominations for coins and currency", *Economic Letters*, 1995.

Tsoumanis K., Axelos N., Moshopoulos N., Zervakis G., Pekmestzi K., "Pre - encoded multipliers based on non - redundant radix - 4 signed - digit encoding", *Computers, IEEE Trans. on*, 2016.

Van Hove L., “Optimal denominations for coins and bank notes: in defense of the principle of least effort” *J. of Money, Credit, and Banking*, 2001.

Venkatachalam S., Ko S.-B., “Design of Power and Area Efficient Approximate Multipliers,” *Very Large Scale Integration (VLSI) Systems,IEEE Transactions on*, 2017.

Vigliar M., Licciardo G. D., “Multiplierless coprocessor for difference of Gaussian (DOG) calculation,” *U.S. Patent 20 130 301 950*, 2013.

Voronenko Y., Püschel M., “Multiplierless Multiple Constant Multiplication”, *ACM Transactions on Algorithms (TALG)*, 2007.

Watanabe Y., Nakamura N., Watanabe S., “Offset Compensating Bit-Line sensing Scheme for High Density DRAM’s”, *IEEE Journal of Solid State Circuits*, 1994.

Wicht B., Nirschl T., Schmitt-Landsiedel D., “Yield and speed optimization of a latch-type voltage sense amplifier”, *IEEE J. of Solid-State Circuits*, 2004.

Witkin, A. P., "Scale-space filtering", *Proc. 8th Int. Joint Conf. Art. Intell., Karlsruhe, Germany, 1019-102*, 1983.

Woo S.-H., Kang H., Park K., Jung S.-O.,“Offset voltage estimation model for latch-type sense amplifiers”, *IET Circuits, Devices & Systems*, 2010.

Wyckoff C.W., “Silver Halide Photographic Film Having Increased Exposure-Response Characteristics”, *US Patent 3,450,536*, 1969.

Xilinx, “*Virtex - 7 Family, DS183 (v1.23)*”, 2015.

Zhang W.-C., Wang F.-P., Zhu T. L., Zhou Z.-F., “Corner Detection using Gabor Filters”, *IET Image Processing*, 2014.

Acronyms and Symbols List

1D	One-Dimensional
2D	Two-Dimensional
3D	Three-Dimensional
1T	One Transistor
3T	Three Transistors
6T	Six Transistors
ADE	Analog Design Environment
ADP	Area-Delay-Power
ASIP	Application Specific Image Processor
BL	Bit Line
BRAM	Block Random Access Memory
CMOS	Complementary Metal-Oxide Semiconductor
CNN	Convolutional Neural Networks
CPU	Central Processing Unit
CSA	Current Sense Amplifier
CSD	Canonical Signed Digit
CT	Tomography
DA	Distributed Arithmetic
DFT	Discrete Fourier Transform
DMA	Direct Memory Access
DRAM	Dynamic Random Access Memory
DSP	Digital Signal Processor
DVI	Digital Visual Interface
EM	Electromagnetic
FF	Flip Flop
FFT	Fast Fourier Transform
FIFO	First In First Out
FP	Floating Point
FPGA	Field Programmable Gate Array
fps	frame rate per second
FS	Fourier Series
FSM	Finite State Machine
FT	Fourier Transform
FCMOS	Full Complementary Metal-Oxide Semiconductor
GPU	Graphic Processing Unit
HDL	Hardware Description Language
HDR	High Dynamic Range
HW	Hardware
IFT	Inverse Fourier Transform

ILSVRC	ImageNet Large Scale Visual Recognition Challenge
I/O	Input/Output
IoT	Internet-of-Things
KNN	K-Nearest Neighbors
LBG	Linde-Buzo-Gray
LDR	Low Dynamic Range
LSB	Least Significant Bit
LUT	Look-Up Table
MAC	Multiply and Accumulate
MB	Modified Booth
MR	Magnetic Resonance
MSB	Most Significant Bit
MUX	Multiplexer
NMOS	Negative Metal-Oxide Semiconductor
NN	Neural Networks
OC	Offset Compensating
pdf	probability density function
PDE	Partial Differential Equation
PMOS	Positive Metal-Oxide Semiconductor
PPA	Power Performance and Area
PSNR	Peak Signal-to-Noise Ratio
RAM	Random Access Memory
ReLU	Rectified Linear Unit
RGB	Red Green Blue
ROM	Read Only Memory
RW	Read/Write
SA	Sense Amplifier
SIFT	Scale Invariant Feature Transform
SIPO	Serial Input Parallel Output
SL	Sense Line
SNR	Signal-to-Noise Ratio
SoC	System-on-Chip
SQ	Scalar Quantization
SRAM	Static Random Access Memory
sRGB	standard Red Green Blue
SSIM	Structural Similarity Index Measure
SW	Software
TMO	Tone Mapping Operator
UHDTV	Ultra High Definition TV
UV	Ultraviolet
VLSI	Very Large Scale Integration
VQ	Vector Quantization

VS	Visual Search
VSA	Voltage Sense Amplifier
WL	Word Line