Journal of Computer Science 1 (2): 207-214, 2005
ISSN 1549-3636
© Science Publications, 2005

A Real-time Image Processing with a Compact FPGA-based Architecture

[1]Ridha Djemal, [2]Didier Demigny and [1]Rached Tourki
[1]Faculté des Sciences de Monastir, Avenue de l'Environnement 5019 Monastir, Tunisie
[2]Laboratoire ETIS, ENSEA Université Cergy Pontoise, France

Abstract: This study have presented a filed programmable gate array implementation of a real time video smoothing algorithm. In comparison with smoothing video techniques like deblocking filters in H.264 or smoothing in JPEG2000, the proposed method is implemented in hardware and its computational cost and complexity are reduced where all pixel processing related to uncompressed video is done on the fly. Our proposed architecture tries to optimize the design of a modified version of the Nagao filter in order to make video smoothing with respect to real time constraints. This filter have to smooth video before applying an edge extraction approach used in industrial applications requiring a high quality production like manifacturing process control. The proposed architecture based on the RC1000P-P Virtex prototyping Board is analyzed to gain an understanding of the relationships between algorithmic features and implementation cost. Experimental results indicate that using this prototyping board with optimized hardware architecture; we can deliver real-time performances and an improvement in the video quality. This filter is capable to process a real time video with a high resolution and deliver 25 images per second at 10 MHz clock cycle.

Key words: Smoothing Technique, Field Programmable Gate Array FPGA, Prototyping Board

INTRODUCTION

Real-time image processing systems have gained an increasing importance in many fields, including applications such as: multimedia, industrial inspection and medical engineering. In addition, many industries such as plastics and textiles industries, require some form of technology for controlling their production quality. One common quality control technique employs cameras that capture the production activities for object dimension measurement with the assistance of an edge extraction algorithms [1]. In order to improve the edge detection technique we propose to smooth the video stream with a specific algorithm which operates in real time manner in order to make better the video quality and the edge detection.
Furthermore, in image processing related to quality control applications where the inspection has to be accurate, it is difficult to analyze the information of an image directly from the gray-level intensity of the image pixels. Indeed, this value depends upon the lighting conditions. More important are the local variations of the image intensity. The size of the neighbourhood where the contrast is computed must be adapted to the size of the objects that we want to analyze (edge detection). This size defines a resolution reference for measuring the local variation of the image. Generally, the structures we want to recognize have very different sizes. Hence, it is not possible to define a priori an optimum resolution for analyzing an image. In this respect, we have to develop a smoothing technique in order to enhance the image contrast and to reduce noise. Consequently, this digital filter can be used in the pre-processing phase, as noise elimination, or in the enhancement of the edge detection approach which is a derivative method.
This study have proposed a new smoothing technique implemented in only one FPGA-Virtex with additionnal memories. This technique consists in allowing on the central point with a neighbourhood of 5x5 pixels, the sum of neighbourhood 3x3 among all with a minimal extent. Remember that the extent is defined as the difference between the Maximal and the minimal intensity related to contiguous pixels.
In this context, we are particularly interested in addressing the results of the real time video smoothing based on the modified version of the Nagao algorithm, since it is used to improve the video quality by eliminating the noise inside the sequence. The other motivation is to present a generic methodology for rapid prototyping of a real time algorithm and shows how this compact hardware plate-form is flexible enough to accurately support such complex applications having real time requirements. The video standards require the processing time less than 40 Ms per image (with a size of 512^2) which indicates that a pixel must be computed each 100 ns taking into account the synchronization aspect. In this respect, the sampling frequency has to be about 10 MHz.

Related Work: Smoothing is a pre-processing technique which can be found both in the segmentation chains [2] and in the compression of images. Indeed, the deblocking filter, for example, is used in the H.264/MPEG-4 video coding standard and carrying out simple operations in order to analyze artifacts localized on coded block boundaries and attenuate them by applying a selected filter [3]. In the same way in JPEG2000 standard, the technique of smoothing is used in order to improve video quality which could be to deteriorated by performing the image compression. The implementation of these techniques requires a high-rank operation such as multiplication, division and square root which are not easy to implement in FPGA technology.

The mentioned filtered techniques are very interesting in video enhancement but are dedicated for a compressed video without a strict time constraints where our context relies on the not compressed video to be enhanced. In [4], many techniques of smoothing have tried to solve the conflict between a good filtering and the safeguarding of contrast. An efficient smoothing is generally accompanied by edges erosion (diffusion). The filters of Sobel [5] carry out a directional smoothing conjugate with a calculation of the gradient in the orthogonal direction. The low number of points used in smoothing limits the associated image quality. The filter of Deriche [6] takes up this idea by exploiting a neighbourhood of greater dimension associated with a parametric impulse response which allows a better smoothing and an optimal detection of a certain type of contours.

Another approach developed by Tomita and Tsuji [7] have proposed a new smoothing technique as pre-processing operations for image in order to facilitate the homogeneous edge detection [8]. It consists in assigning the central pixel of the 5x5 neighbouring pixels the mean intensity of a 3x3 neighbourhood subsets with central pixels I, A, B, C and D (Fig. 1a). The main idea is that a 3x3 neighbourhood including an edge will lead to a variance stronger than a neighbourhood only disturbed by the noise.

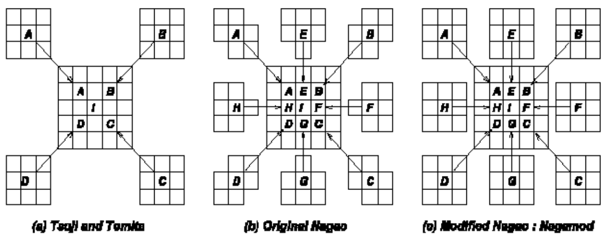


Fig. 1: Neighbourhood Structure

The disadvantage of this technique consists on discarding small regions and this phenomenon is amplified when we iterate this technique many times onto the same image. Nagao et al. have addressed this problem [8] and have proposed a new structure of the neighbourhood as depicted in Fig. 1b to enhance its performance. The hardware implementation of the nagao filter is not easy to perform because of the irregular mask shapes which complicate enough pixels processing. In addition, computing the variance and the mean requires an important amount of hardware and processing time particularly with the FPGA technologies. In order to simplify its hardware implementation without any video quality regress, Demigny [6] has proposed a modified version in which he considered 9 neighbourhoods of 3x3 pixels with the following centres : A, B, C, D, E, F, G, H and I (Fig. 1c) without appreciable decreasing of the video quality [9]. In addition, the variance processing is replaced with *extent* and *sum* computing. These simplifications allow a new nagao version called *Nagamod technique* and involve a hardware implementation in a real-time environment.

Nagamod Filter: As depicted in the above section, the Nagamod version presents a particular neighbourhood structure which facilitates the hardware implementation of the filter without deteriorating the associated performances. Consider a 5x5 neighbouring pixels, these pixels luminance has some correlation which we would take advantage of. In fact, pixels of intensity related to the current image arrive according to the scanning order of the line. We can consider the image as a mono-dimensional pixels flow. So, two consecutive pixels of the same line are separated with one clock cycle corresponding to the delay z^{-1} in terms of Z transform. Similarly, two pixels belonging to the same column of two consecutive lines are separated with N clock cycles (delay equal to z^{-N}) where N represents the pixels number per line.

There are many architectural solutions to implement the Nagamod filter. In all solutions, pixels processing is performed using two blocks: the first one performs the computation of the sum and the distance related to the neighbourhood 5x5, where the second block determines the neighbourhood 3x3 with minimal distance through the three neighbourhoods in order to be matched. In this respect, the Nagamod architecture uses some basic operators to deliver the minimum, the maximum and the extent between three consecutive pixels. In addition, we have to compute the sum and to make the selection between three couples of 3x3 neighbourhoods with specific operators. Consequently, the architecture can be composed by an association of only two basic components named B_1 and B_2. The B_1 component is based on three basic units which give the minimum and the maximum of three pixels and perform the extent and the sum of them [9]. This block delivers the extent and the sum related to the 3x3 neighbouring pixels where the B_2 bloc gives the sum of the 3x3 neighbouring pixels among the 9 subset of 3x3 neighbourhoods.

Given the extent and the sum of the 3x3 pattern as depicted in the Fig. 1c (A E B) (H I F) and (D G C), we have to determine the minimal extent among them. The component B_2 performs this operation onto the 9 groups of 3x3 neighbourhoods and delivers the minimal extent with its associated sum (Fig. 2b).

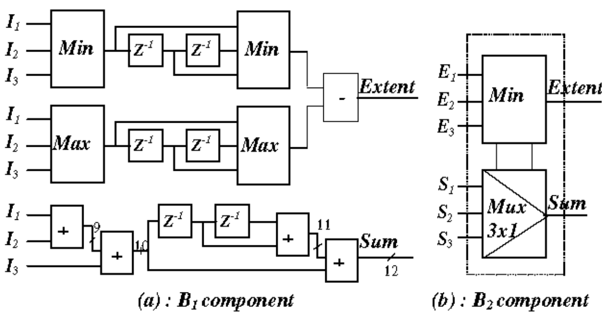


Fig. 2: B_1 and B_2 Components

Nagamod Hardware Solutions: The Nagao filter can be implemented according to different organizations with regard to the performance needs. If we do not have timing constraints, we can implement this algorithm in software manner and obtain the required results. But, if we want to apply this technique to real time application like video smoothing, the software implementation is not suitable and the hardware implementation becomes mandatory. Furthermore, the target architecture can introduce some additional timing constraints. For example FPGA technology combined with a synchronization board have a limited frequency, about 10 MHz (to deliver 25 images per second), and extended gate delay has strict constraints compared to the ASIC one. Three hardware solutions have been analyzed with equivalent rapidity in order to choose the appropriate one for our hardware environment. The comparison criterion is based on the resources needed in term of memory and controller complexity. We note that the required amount of memory with its memory controller represent the main cost in such architecture. These remarks can be applied for other filter architectures like Sobel and Diriche [10]. In the first solution organization (Fig. 3), the architecture is composed of one B_1 block working in conjunction with two B_2 blocks. The data is buffered both in the input and in the internal part of the architecture. This structure makes the implementation more difficult to perform and particularly the memory controller part.

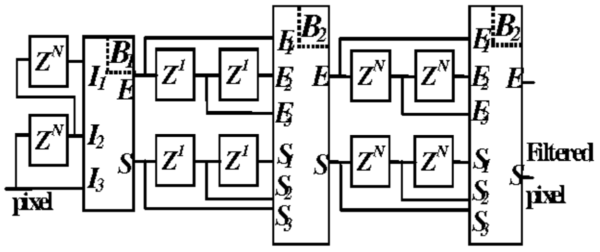


Fig. 3: Solution 1

Figure 4 shows the overall components of the second hardware solution. This version is compact and requires less amount of memory which can be the external of the computing component.

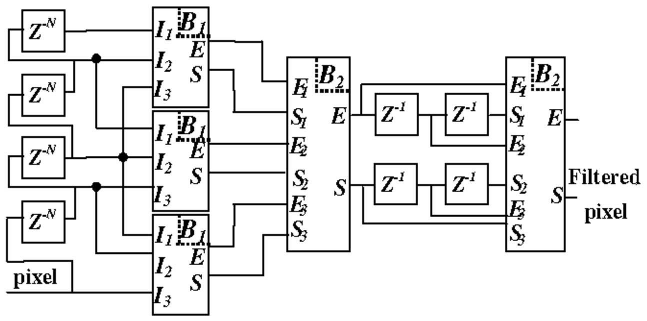


Fig. 4: Solution 2

The third solution, shown in Fig. 5, considers the principle used in the second solution and computes all the operations for two pixels in a parallel way.

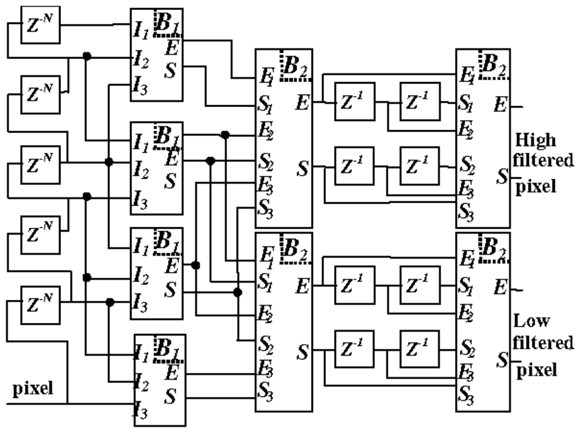


Fig. 5: Solution 3

This solution uses three B_1 blocs and four B_2 blocs. However, it can be very interesting if we want to compute the gradient behind. In addition, this solution does not require as much memory as that required by the first one. If the delay z^{-1} is implemented with 8-bit register to code the pixel intensity, a delay z^{-N} is implemented using the FIFO (*F*irst *I*n *F*irst *O*ut) memory organization to buffer the N pixels intensities of the image line. The Table 1 shows that the second solution is more suitable four our application context and requires a reduced amount of memory. The Table 1 resumes the amount of buffer space related to each solution.

Table 1: Architecture Requirements

Architecture organization	Internal 8-bit reg	FIFO number	Required Size	Memory complexity
Solution 1	4	6	16 Kbits	Complex
Solution 2	4	4	24 Kbits	Less complex
Solution 3	8	5	20 Kbits	Less complex

Nagamod Filter Hardware Implementation

System onfiguration: In order to verify the real-time performance and functionality of the proposed Nagamod filter, a commercially available device, the camera, the grabbing board and the RC1000P-P FPGA based board have been used as depicted in the Fig. 6. The first one, named cyclope board, represents the synchronization one responsible for extracting only the pixel output after grabbing the video with a camera in order to be processed within the FPGA board and restitutes the synchronization before displaying the filtered video sequence. The second borad consists on FPGA prototyping board (RC1000-PP) in which we have to download our architecture bit-stream through the PCI interface of the PC-2.

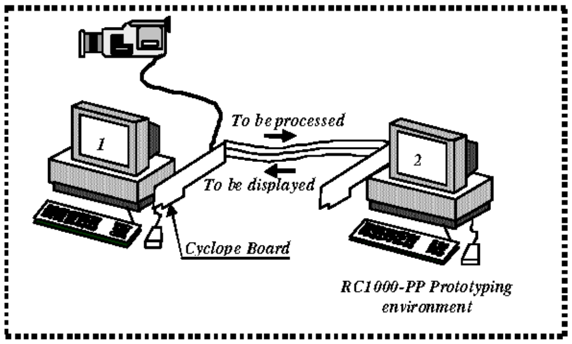


Fig. 6: Scheme of the Experimental Setup

After being filtered the image pixels intensities are sent to the Cyclope board again in order to be displayed on the screen 1 according to the video standard with a 10MHz frequency clock speed.

Video Signals Control: The Cyclope Board presents some specific signals in order to synchronize the image displaying and achieves the real time character according to the high resolution (25 images per second). This board is responsible for grabbing video sequence and interacts with the camera, the prototyping board and the display with the following signals as depicted in the Fig. 7.

* **Pixel Clock:** This clock signal is given with the acquisition board and represents the system clock for our smoothing architecture. For a video flow of 25 images per second, the clock has 10 MHz for a 512^2 image size and reaches 40 MHz for 1024^2 image size.
* **Blank:** When it is high, this signal indicates that the received pixels belong to the same line. In the other case, all computing operators can be deactivated until the arrival of the first pixel of the second line in the same image where blank is going on (high).
* **SYNH:** This signal gives periodically a negative pulse during the dead time between two consecutive lines belonging to the same image.
* **SYNV:** In this case, the negative pulse is generated between two consecutive images. This pulse width is equal to the inter-image time.

The observation of the validations signals described below related to digital video reveals the existance of two types of dead time. The first one is situated between two consecutive lines where the second one is dedicated for the inter-frame delay related to the video stream.

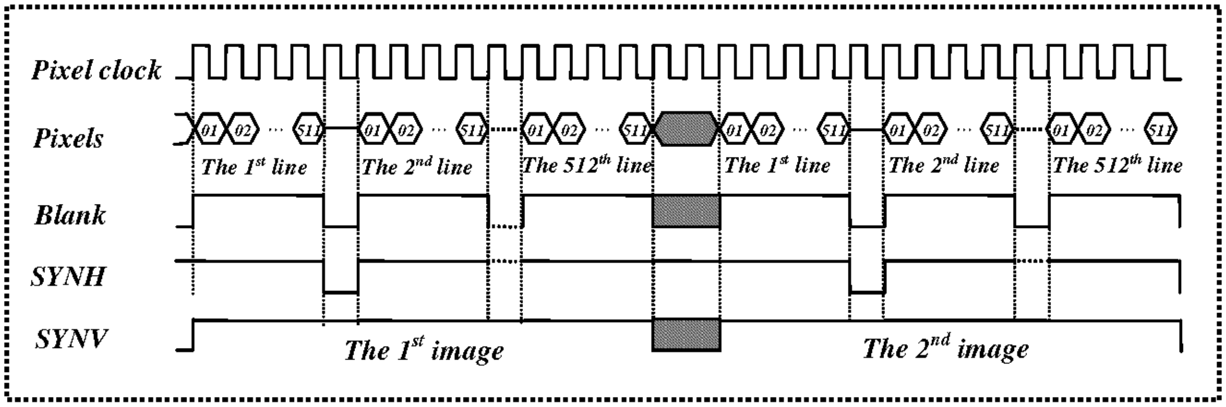


Fig. 7: Synchronization for Non Iinterleaved Video Representation

These delays are imposed with the video standard which allow for a cathodic screan a delay for a line feed (right to left) and for a frame return (from top to down). For a 512x512 image, the sum of these delays is a bout 13.78 ms which represents 34 % of the total time required for displaying the total image. These delays introduce an additionnal constraints which we have to consider during the hardware implmentation in order to meet the real time smoothing using the FPGA technologie.

FPGA Implementation: Our architecture is described with the VHDL language in Register Level Transfer (RTL) according to the second solution presented below in order to be checked onto the FPGA VIRTEX technology. The architecture style relates with a combined data flow with a memory dominated [11,12] where the control dominated model is not suitable for this application [13]. In order to meet real-time constraints, it is important to define the target of the image/video smoothing specification. If conventional TV and some MPEG-2 standards are considered, the proposed architecture must be able to reach a processing rate of 25 frames per second. The processing should include smoothing of such images with respect to real time constraints. A 720x576 pixel frame must be processed within 40 Ms. Therefore; the processing of one pixel should not exceed 96 Ns (a minimum frequency of 10.3 MHz).

In order to satisfy these constraints, we have designed the Nagamod architecture incorporating parallelism and pipelining by integrating four fixed hardware components with an external memory bank as shown in Fig. 8.

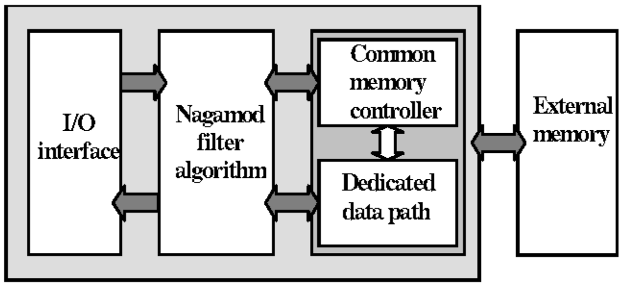


Fig. 8: Functional Block Diagram Architecture

The proposed architecture consists on three main components such as:

- The I/O interface which interacts with the commercial synhroniastion borad (cyclope).
- Nagamod Filter Algorithm: This component is responsible for all pixels computations related to 5x5 neighbourhoods with logic and arithmetic operators like min, max, add, sub, extent, etc.
- Common Memory Controller (CMC): The main function of the CMC is to store the incoming video traffic in certain individual queues to the external data memory, retrieve a part of stored pixels in order to be computed and make a new rearrangement of data organization according to the need of the Nagamod algorithm (Fig. 9). Using four FIFOs line-based storage linked together, the CMC is capable of updating the logical organization in one clock cycle in order to satisfy the real-time caracter. This component has its dedicated data path including all needed operators like counters, address generation, etc.

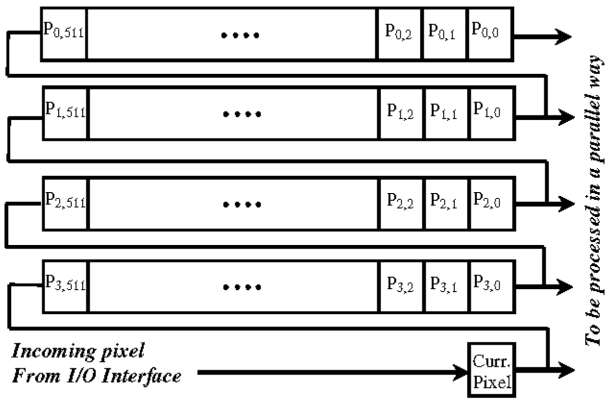

$P_{0,511}$ •••• $P_{0,2}$ $P_{0,1}$ $P_{0,0}$
$P_{1,511}$ •••• $P_{1,2}$ $P_{1,1}$ $P_{1,0}$
$P_{2,511}$ •••• $P_{2,2}$ $P_{2,1}$ $P_{2,0}$
$P_{3,511}$ •••• $P_{3,2}$ $P_{3,1}$ $P_{3,0}$
Incoming pixel From I/O Interface
Curr. Pixel
To be processed in a parallel way

Fig. 9: Logical Organization of the External Memory

We remark that this logical organization architecture requires 4 FIFOs based-line storage which are implemented using an external memory device. Remember that implementing FIFO inside the FPGA requires a large number consumption of CLBs [14].

Our architecture is synthesized to be checked onto the RC1000P-P, which is a PCI Option Card with an on-board secondary Bus [15]. This prototyping board can be connected to the PCI Bridge through the PLX PCI908 (EEPROM) PCI Bridge, and runs all of these devices at host-PC PCI speed. It has a XILINX FPGA in BG560 package and Virtex 1000 device. The RC1000P-P is clocked at 33.33 MHz and can be programmed from the host PC over the PCI bus, or using theXilinx Xchecker download cable. In addition, there are four memory banks, as shown in Fig. 10, where the FPGA has four 32-bit memory ports, one for each memory bank. It can therefore access all four banks simultaneously and independently. Each bank of SRAM on the RC1000-PP can be accessed by the host via the PLX PCI9080 or by the FPGA according to the local switch positions and only the SRAM bank0 is used for the Nagamod hardware implementation.

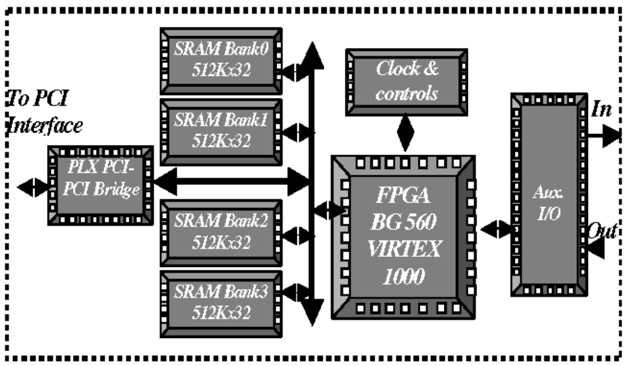
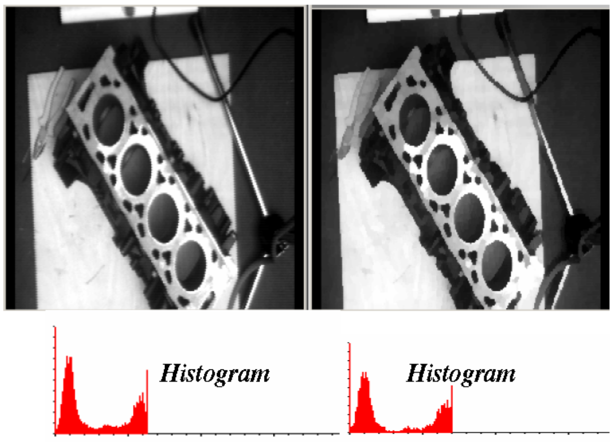


Fig. 10: Block Diagram of the RC1000P-P Board

This memory has implemented the four FIFOs where the access time is about 17 Ns. In order to suit the real time constraints, we have to make a read and write operations related to the two consecutive pixels intensities in the same cycle.

EXPERIMENTAL RESULTS

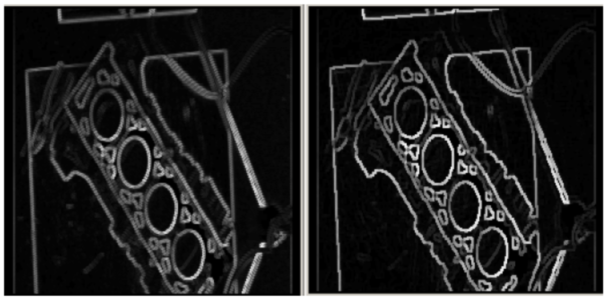
Qualitative Evaluation of the Nagamod Algorithm: The first stage of the experimental part consists on simulating the hardware implementation with ModelSim on the RTL level. We have comapred the smoothing image to the original one captured by a camera. Indeed, we have simulated a 256x256 binary image, containing 256 lines where each line contains 256 pixels with 256-gray scale. The Fig. 11 shows two image configurations before and after the simulation. By comparing the reconstructed image after smoothing operations with the original one shown in Fig. 11, we can appreciate the quality of filtered image. This image has a better contrast and the noise represented by small regions is eliminated. The difference between the two images is well illustrated with the histogram representation.



(a) Original image *(b) Nagamod image*

Fig. 11: RTL Simulation Results using a 256x256 Binary Motor Block Image

In order to prove the efficiency of the nagamod technique and illustrate the smoothing results as a pre-processing phase, we have computed gradient the original image and the Nagamod smoothing one. The Fig. 12 demonstrates the efficiency of the Nagamod algorithm for eliminating noises and improuving the edge detection. We remark that the edge detection is better performed after applying the smoothing technique end an important part of existing noise has been eliminated. Furthermore, the gradient related of filtered image is computed using an image generated with our RTL architecture before synthesis.



(a) Gradient of original *(b) Gradient of the filtered Image*

Fig. 12: Gradient Analysis of Filtered Image

Figure 13 presents the RTL simulation related to the memory controller. It shows the manner in reading and buffering data. It also demonstrates the synchronization aspect with *enable(Blank)* and *gnt0(SYNH)* signals which are responsible for activating memory operations. In fact, during the dead time between two consecutive lines and two consecutive frames, our memories do not perform read and write operations inducing an optimisation of the power consumption for our architecture. In addition, the design can be used for many video standards with different frequencies.

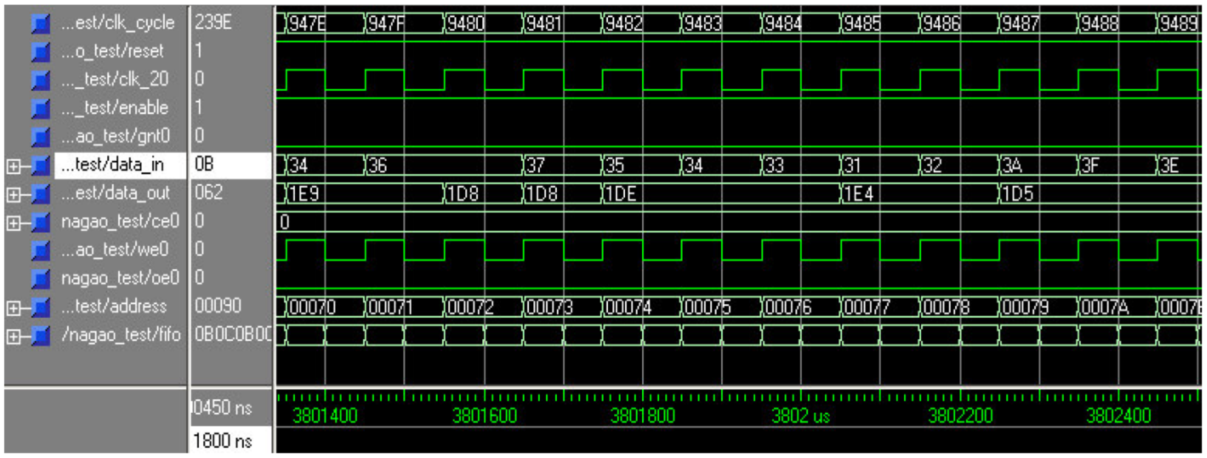

...est/clk_cycle 239E
947E 947F 9480 9481 9482 9483 9484 9485 9486 9487 9488 9489
...o_test/reset 1
..._test/clk_20 0
..._test/enable 1
...ao_test/gnt0 0
...test/data_in 0B
34 36 37 35 34 33 31 32 3A 3F 3E
...est/data_out 062
1E9 1D8 1D8 1DE 1E4 1D5
nagao_test/ce0 0
0
...ao_test/we0 0
nagao_test/oe0 0
...test/address 00090
00070 00071 00072 00073 00074 00075 00076 00077 00078 00079 0007A 0007B
/nagao_test/fifo 0B0C0B0C
0450 ns
1800 ns
3801400 3801600 3801800 3802 us 3802200 3802400

Fig. 13: RTL Architecture Simulation Results with ModelSim Tool

Quantitative Evaluation of the Nagamod Algorithm: In order to quantify the visual quality of the image after smoothing, we have used the PSNR metric, more appreciated for the video applications than the SNR one. The PSNR is usually defined as:

$$PSNR = 10\log_{10}\frac{255^2}{MSE}$$

Where, MSE is the mean square error given by:

$$MSE = \sum_m \sum_n [x(m,n) - \hat{x}(m,n)]$$

The PSNR is measured in decibels (dB) where $x(m,n)$ represents the pixels intensity related to the original image and $\hat{x}(m,n)$ the rebuilt one after smoothing operations. The PSNR remains a good measure for comparing restoration results for the same sequence of image. We have applied our technique onto another image (desk) with a weak contrast. The following table gives the PSNR values related to the two test images.

Table 2: Architecture Requirements

Image	PSNR in dB
Desk image	6.89
Motor block image	16.42

The PSNR measure quantifies the difference between the original image and the smoothed one. It shows the contribution of our technique to enhance the image visual quality. According to the Table 2, the second image with 16.42 dB PSNR may look much better than the other one with 6.89 dB because it has a better contrast and the Nagamod technique suits well with this kind of image.
In the rest of this section, we have to represent the design effort and resources in terms of complexity involved in the generation of the RTL architecture. In the advanced validation process, the RTL description is mapped onto the XILINX bg560 library related to the VIRTEX 1000 family [16] in order to be downloaded onto the RC1000P-P board. In the rest of this section we present the FPGA synthesis results for the main modules of the architecture. The FPGA technology was chosen since it provides some implementation advantages such as : (i) FPGAs provides a high-density for logic arithmetic at relatively short design cycles, (ii) in FPGAs, it is possible to control operations at bit level to build specialized datapaths. The primary results of the synthesis (Table 3) represent the number of configurable logic bloc (CLBs) after optimization for each component of the block diagram presented below. In addition, the table presents the amount of Flip Flop buffers related to each component.

Table 3: Nagamod Synthesis Results

Component	CLB number	Flop number
B_1	104	52
B_2	32	0
Memory controller	55	54
Dedicated data path	39	77
Optimized Nagamod	435	361

Timing Analysis: From a temporal view point, the traverse of a data flow through an operator introduces a latency delay between the input and the output. So, the critical path related to the glue logic part of the Nagamod filter has a critical path equal to 25,1 Ns corresponding to the crossing time related to the B_1 and B_2 components. When we add the memory controller part, we can go beyond the limited delay according to the video frequency 100 ns per period. If the propagation time is higher than the sampling period related to input pixels (100 Ns),), it would be mandatory to introduce some registers (pipeline) through this path. Note that the association of two combinatory operators A and B pipelined with a respective latencies t_A and t_B conduct to a minimal period:
$T_{min} = \max(t_A,t_B)$. On the other hand, without pipeline this delay has to be: $T_{min} \leq (t_A + t_B)$. In our implementation the processing time exceeds 100 Ns. In order to overcome this difficulty, we have introduced the pipeline organization with 2 stages in order to divide the critical path onto two cycles and reach the timing requirements.

CONCLUSION

In this study, we have attempted to show design exploration architecture for real time environment. The detailed architectures for the hardware implementation of the Nagamod technique with all functions and operators have been carried out in a suitable manner. We have discussed the different design solutions. We showed that our proposed solution provides interesting features and suits well with our design and prototyping environment. The architecture has been implemented on a RC1000P-P FPGA of XILINX. The detailed architecture of different block elements of the whole design has been shown. Satisfactory results have been obtained with the FPGA Virtex technology from complexity and speed point of view. Our choice is well justified and gave good results with an optimized memory controller architecture. The proposed architecture reduces the noise in the video sequence and makes the smoothing operation in real-time manner. We reviewed in particular the improvement obtained with the Nagamod algorithm for edge detection and for video visual quality. Future works will extend the current study to apply them for controlling the production quality of heavy industries.

REFERENCES

1. Canny, J., 1986. A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 8: 679-698.
2. Meribout, M. and K.M. Hou, 1999. Real-time image segmentation on a massively parallel architecture. J. Real-Time Imaging, 5: 279-291.
3. Kang, L.W. and J.J. Leou, 2004. An error resilient coding scheme for H.264/AVC video transmission based on data. J. Visual Communication and Image Representation, (In press).
4. Taubman, D. and M.W. Marcellin, 2002. JPEG 2000 Image Compression Fundamental Standards and Practice. Kluwer Academic Publishers.
5. Dantu, R.V., N.J. Dimopoulos, K.F. Li, R.V. Patel and A. J. Al-Khalili, 1994. Parallel algorithms for low-level vision on the homogeneous multiprocessor. Computers and Elect. Engg., 20: 51-60.
6. Demigny, D., 2001. Méthodes et Architectures pour le TSI en Temps réel. Hermes Editor-Lavoisier.
7. Tomita, F. and S. Tsuji, 1977. Extraction of multiple regions by smoothing in selected neighbourhoods. IEEE Transaction on Systems Man and Cybernetics, pp: 107-109.
8. Nagao, M. and T. Matsuyama, 1979. Edge preserving smoothing. Computer Graphics and Image Processing, pp: 394-407.
9. Demigny, D., J. Devars, L. Kessal and J.F. Quesne, 1993. Implantation temps réel du filtre de lissage d'images de nagao. Revue de Traitement de Signal, 10: 319-329.
10. Bouguiba, R., 2000. Conception d'une architecture matérielle reconfigurable dynamiquement dédiée au traitement d'images en temps réel. Ph. D. Thesis. Cergy-Pontoise University, France.
11. Vahid, F. and D. Gajski, 1994. Specification and Design of Embedded Systems. Printice Hall Editor, Englewood Cliffs, NJ.
12. Fisher, G.E., 1994. Rapid system prototyping in open system environment. In Proceedings of International Workshop on Rapid System Prototyping (RSP), Grenoble France.
13. Djemal, R., G. Mazaré and R. Tourki, 2000. Rapid prototyping of an ATM programmable associative operator. J. Systems Architectures, 46: 1159-1173.
14. Andraka, R., 1998. A Survey of CORDIC Algorithm for FPGA Based Computers, pp: 191-200.
15. RC1000P-P. RC1000P-P Hardware Reference manual. INRIA.
16. APS-X84. VHDL/FPGA Synthsis Tutorial-FPGA Basics. http://users.erols.com/aaps/x84lab/FPGA. htm