DESIGN AND HW/SW IMPLEMENTATION OF A NONLINEAR INTERPOLATOR FOR BORDER PRESERVING

ANIS Ridha BOUDABBOUS[1], Marwa Jomaa GRAJA[2]

[1,2]Jouf University, College of Computer and Information Sciences, Kingdom of Saudi Arabia

---***---

Abstract - *In this paper, a new design and HW/SW implementation of a nonlinear interpolator for border preserving is presented. The objective of this proposed work is to achieve significant run time performance using a hardware/software development board. It also demonstrates consistent image quality performance among a variety of images. We showe that our HW/SW solution improves considerably the Interpolation speed (258 times faster) compared to software solution as well as preserving a high image quality.*

Key Words: **Nonlinear interpolation, color image, HW/SW, FPGA, Reconfigurable Architecture.**

1. Introduction

In numerical analysis, interpolation is a mathematical operation to construct a curve from the data of a finite number of points, or a function from the data of a given finite number of values. The solution of the interpolation problem passes through the prescribed points, and, according to the type of interpolation. In the case of images, the interpolation consists to "add pixels where there are none". The degradation of image quality appear when you resize a digital image, or resize it with a distortion, change the perspective, etc ...So, It is important to understand how interpolation works, to know how to resize your photos by losing as little quality as possible. In fact, In the image processing applications, image interpolation is a process which estimates a set of unknown pixels from a set of known pixels in the image. High quality interpolated images are obtained when the pixel values are interpolated according to the edges of the original images [1]. In the literature, some interpolation methods based on multi-directional filters [2] and the statistical approach [3], and polyphase weighted median filters [4] are used. Accordingly, the development of image interpolator for multidimensional data processing has a great importance for various applications. The interpolator is designed to soften the image and sharpen while preserving image details. The simplest solution is to use a bilinear interpolation and interpolate each plan independently [5]. This solution is very effective, but it produces a smoothing of the luminance and the appearance of false colors.

The interpolation technique presented in this paper is a Border-Preserving Interpolator of color image. The complete methodology for developing the Border-Preserving Interpolator is described in [6] but without any hardware realizations. In this paper, a HW/SW methodology is used to implement this Interpolator using Altera Stratix II EP2S60F60 FPGA. Indeed, this paper focuses on FPGA implementation of Border-Preserving Interpolator using a HW/SW context validation. In this way, our work contribution is mainly to develop a new hardware implementation approach of Border-Preserving Interpolator based on HW/SW codesign context in order to speed up the Interpolation process compared to the software solution.

So, this paper is structured as follows: Section 2 presents a brief overview of Border-Preserving Interpolator. Section 3 proposes reconfigurable hardware architecture of the Border-Preserving Interpolator. In section 4, we present the validation results of the Border-Preserving Interpolator in HW/SW context using Altera Platform. Finally, conclusions are drawn in section 5.

2. Overview of Border-Preserving Interpolator

Border-Preserving Interpolator operates on four samples of decimated multivariate data, a, b, c and d (mask 3x3) to reconstruct the missing sample x in the central position. The masks that can be used in the interpolation method are given in Figure 1.

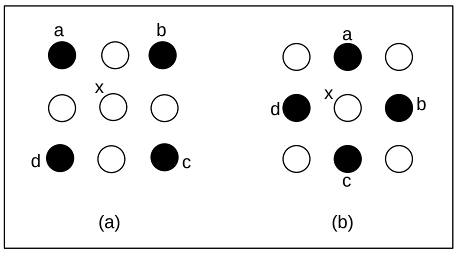


Fig. 1. A 3x3 mask to interpolate the sample x using the known data a, b, c and d. (Border-preserving interpolator)

The normalized value of x verifies two conditions. First, each weight is a positive number. Second, in a flat area (where we assume that the four neighbors of the pixel under consideration have the same value), the sum of the weights is equal to 1, which ensures that the output is unbiased. The value of interpolator output is calculated using this equation:

$$x = \frac{\sum_{\{u \neq v; u,v \in (a,b,c,d)\}} w_{uv}.(u+v)}{2\sum_{\{u \neq v; u,v \in (a,b,c,d)\}} w_{uv}}$$

$$= \frac{1}{W}\left(\frac{w_{ab} + w_{ac} + w_{da}}{2} a + \frac{w_{ab} + w_{bc} + w_{bd}}{2} b \right. \qquad (1)$$

$$\left. + \frac{w_{ac} + w_{bc} + w_{cd}}{2} c + \frac{w_{bd} + w_{cd} + w_{da}}{2} d \right)$$

Where $W = w_{ab} + w_{bc} + w_{cd} + w_{da} + w_{ac} + w_{bd}$. The weights are computed based on vector- rational function, as follows:

$$w_{uv} = \frac{1}{12 + k\|u - v\|} \qquad (2)$$

Where $u, v \in \{a, b, c, d\}$ and $\|.\|$ denotes l_1 norm

The above equation describes how to calculate the value of pixel 'x' from known pixels a, b, c and d. The parameter k is a user-specified depending on the application. Referred to [6], we choose k = 0.025. In this method the interpolation uses weighted average of known pixels. As can be seen the weights W_{uv} are calculated using the vector- rational function $f(u,v) = \|u - v\|$. This use of the function leads to the conclusion that if one pixel different from other pixel only in one component, it will not be used in the calculation of pixel x. This vector- rational function causes interpolation to use color-edges in the original image as barriers to color spots, so they won't attack other part of the image. We recall that the main goal of this work is to reduce the interpolation processing time. Therefore, the Border-preserving interpolator requires much computation. This is due to the expensive computation of the vector-rational function given by equation (2). In the following paragraphs, we propose hardware architecture to accelerate the SW solution of this type of Border-preserving interpolator.

3. Proposed Reconfigurable Hardware Architecture of the Border-Preserving Interpolator

In this part, we propose a new design and HW/SW implementation based on SOPC system (System on Programmable Chip). The implementation steps are the following :

– C/C++ programming of the interpolator
– Choose HW or SW implementation (Partitioning)
– HW part : Develop the FPGA design and code using VHDL language.
– Program the FPGA
– SW part : Develop the interpolator C/C++ code and validate the code using CPU instruction
– Integrate HW and SW parts in the same FPGA board and check the final functional test.

In fact, the amount of hardware required for an accurate implementation is quite large. This fact is due to the use of the l_1 norm, which requires the computation of the rational function in (2).

$$Nuv = \|u - v\| = (Ru - Rv)^2 + (Vu - Vv)^2 + (Bu - Bv)^2 \quad (3)$$

The Architecture shown in Figure 2 has been used to perform the l_1 norm (i.e. *Nuv*). In this architecture, we use three subtractions corresponding to R, V, and B components, to calculate (Ru-Rv), (Vu-Vv) and (Bu-Bv). Also, we use three multipliers to calculate the square. Finally one adder delivers the value of the l_1 norm.
The whole interpolator architecture is shown in Figure 3. The architecture described in VHDL consists of some adders and multipliers to calculate the value of x after calculating *W* and its inverse. Also, all division by 2 is replaced by right shifts.

This architecture of Border-preserving interpolator was described in VHDL language, validated using Mentor Graphics ModelSim and synthesized for Altera Stratix II EP2S60F60 FPGA circuit [7] with speed 3 grades. Using ModelSim simulations, the good functionality of the designed architecture has been verified

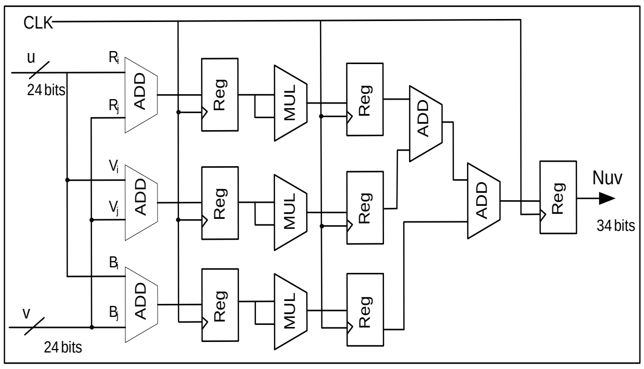


Fig. 2. Architecture of l_1 norm (Nuv)

4. Validation Results of the Border-Preserving Interpolator in HW/SW Context

Using the HW/SW design, we can have several potential advantages. First of all, we minimize overall system cost, development time and the cost per unit. In addition, we implement only the hardware and software needed for a particular design. Also, we have the ability to exploit the parallelism and the pipeline in architectures to design. Finally, and Using the HW/SW design, it is possible to make optimizations in hardware resources (FPGA) and processing time.

4.1 NIOS-II Development Board

The ALTERA NIOS-II softcore processor (Fast version) [8] is a 32-bits scalar RISC with Harvard architecture, 6 pipeline stages, 1-way direct-mapped 64KB data cache, 1-way direct-mapped 64KB instruction cache and can execute up to 150 MIPS. The main interest of this softcore processor is its extensibility and adaptability. Indeed, users can incorporate custom logic directly into the NIOS-II Arithmetic Logic Unit (ALU). Furthermore, users can connect into the FPGA the on-chip processor and custom peripherals to a dedicated bus (Avalon Bus). Thus, users can define their instructions and processor peripherals to optimize the system for a specific application.

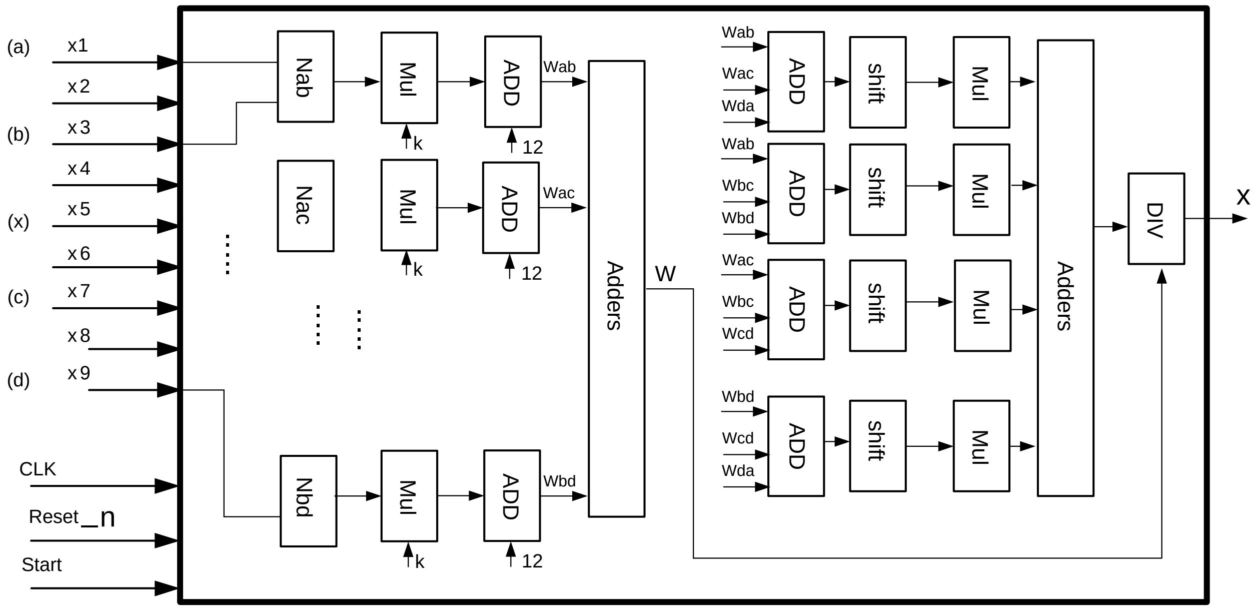


Fig. 3. Parallel architecture of Border-Preserving Interpolator

4.2 HW/SW interpolator System

In this part, we developed a connection interface between the Nios II embedded processor and the hardware architecture across the Avalon bus. The interface entity is given by the figure 4.

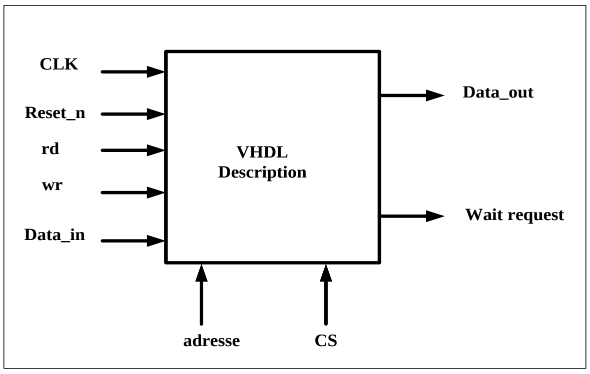


Fig. 4. Hardware interface entity

This entity describes the various inputs/outputs of the hardware system HW/SW *(or SoPC: System in Programmable Chip)*. The different signals *(data_in, Data_out etc.)* are connected to the Avalon bus and allow the reading, writing and data synchronization. Table II summarizes the size and function of the signals used in this entity

TABLE II
SIZE AND SIGNAL DESCRIPTION

Signals	Size (bits)	Description
CLK	1	Clock signal
Reset_n	1	Initialization signal
rd	1	Read Signal
wr	1	Write signal
Data_in	32	Signal Data Input
Adresse	2	addressing
CS	1	Chip Select
Data_out	32	Signal Data Output
waitrequest	1	Wait signal

Figure 5 shows the different parts of the HW/SW implemented system. They are developed in the SoPC Altera environment and validated using an Altera Nios II development board [9].

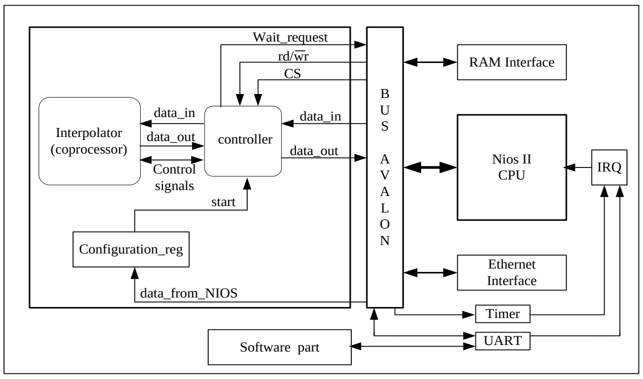


Fig. 5. The SoPC Interpolator system

The system designed consists of the Nios II processor, the Avalon bus, peripherals (memory controller, UART, timer etc.) and hardware accelerator (interpolator coprocessor). In our system, the Treatment is mainly based on the Nios II processor, which is connected to hardware devices via the Avalon bus. This bus is configured in master/slave. It is automatically generated to fit the needs of the design especially for the interconnection of devices.
The synthesis targeted the Altera FPGA was made using the Altera Quartus II tool. Table I, illustrates the hardware cost in terms of ALUTs (Adaptive Look-Up Tables), and DSP block and Input output pins in Stratix EP2S60F672C3 FPGA.

TABLE I
THE IMPLEMENTATION RESULTS IN STRATIX II FPGA

Border-preserving interpolator	HW coprocessor	Whole HW/SW design
ALUTs	967 / 48,352 (2 %)	5,802 / 48,352 (12 %)
Pins	148 / 493 (30 %)	284 / 493 (58 %)
DSP block	129 / 288 (45 %)	138 / 288 (48 %)

The architecture exploits 48% of DSP and 58% of Pins. We use 12% of ALUTs. The whole design process at 60 MHz system clock.

The software part is used, basically, to optimize the reading of decimated image, the reconstruction of interpolated image and the loading pixels to and from hardware core (i and j are respectively the length and the width of the image). The used idea [10] of loading pixels is described by:

- The first interpolator window is formed by sending nine pixels to the hard core.
- The second interpolator window is obtained by exploiting the last six pixels from the previous window and sending only three new pixels to the hard core.
- The j_{th} interpolator window is obtained by exploiting the last six pixels from the $(j\text{-}1)_{th}$ window and sending only three new pixels to the hard core.

In addition, in this work, we use the µClinux as an operating system to control the functionality of the design. Linux for embedded systems (or embedded Linux) gives us several benefits: It is ported to most of processors with or without Memory Management Unit (MMU). A Linux port is available for the Nios II softcore. Most of classical peripherals are ported to Linux. A file system is available for data storage. A network connectivity based on Ethernet protocols is well suited for data recovering. To control the interpolation coprocessor, the technique of state machine is used. Synchronization is performed also by using control signals such as start, done, etc.

5. Experimental Results

The cycle number and the image quality are two main metrics to compare the hardware implementation and software on a single FPGA platform for experimental validation. In this experiment, the ideal Border-Preserving Interpolator (SW solution) is compared to its hardware implemented version (HW/SW version) by using a set of standard images. For this, we use the metric MSU Blocking metric [11]. This metric also contains heuristic method for detecting objects edges, which are placed to the edge of the block. Knowing that the original image (not interpolated) the MSU is equal to 7.33, table III illustrates a comparison between ideal Border-Preserving Interpolator and its HW/SW version using different QCIF color images (176x144, this size is used by 3G phone applications).

TABLE III
COMPARISON BETWEEN SW AND HW/SW IMPLEMENTATION

	SW	HW/SW
Number of cycles	512110054	1982153
Total time (ms)	8535	33
MSU	6.10	6.21

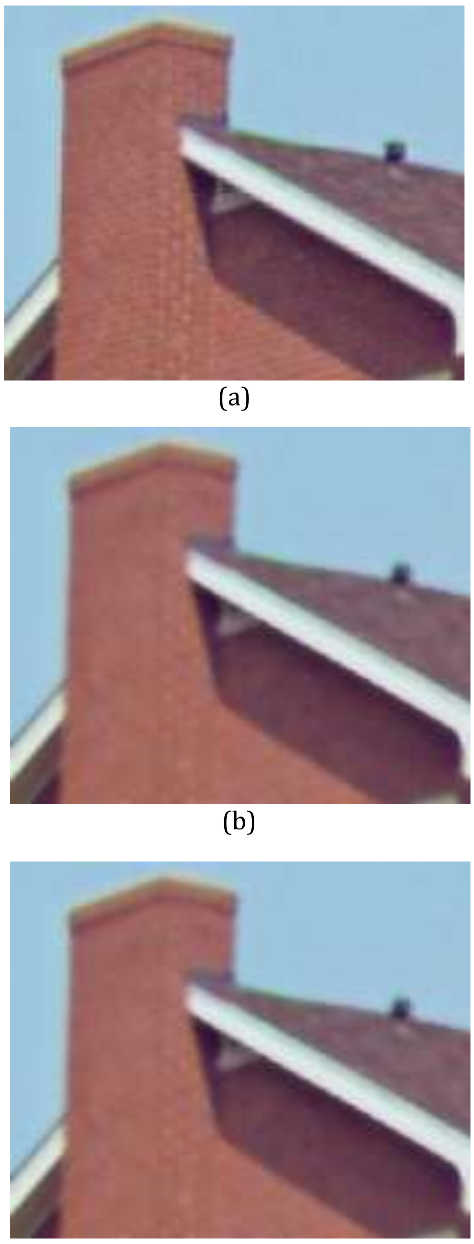
The HW/SW version of Border-Preserving Interpolator slightly outperforms its respective SW version. This was expected due to the rounding effects in VHDL description. Nevertheless, the difference between the HW/SW and SW version of Border-Preserving Interpolator is negligible for the most of cases, which demonstrates the accuracy of the presented architecture.
It is clear that the implemented Border-Preserving Interpolator preserves the chromaticity components and fine details of color images. We can note that the HW/SW implementation of the Border-Preserving Interpolator provides a good time improvement (interpolation speed) compared to the software solution. It is clear that the performance using hardware accelerator (HW/SW solution), is much faster than the execution of algorithms using software solution with approximately same image quality. This difference is caused by the introduction of the rational function in the Border-Preserving Interpolator. In reality, the division blocks (used in the architecture) implemented in the FPGA increase unfortunately area occupation and execution time. This work can be improved by looking for an approximation to calculate rational function (or l_1 norm) without Altera divider. Also, this work can be extended to Angular Interpolator and a complete comparison between different interpolators. (Image quality and FPGA implementation)

6. Conclusion

In this paper, HW/SW implementation is performed to speed up Border-Preserving Interpolator. So, the actual co-design based implementation constitutes a balance between the two well known requirements: time and area. Our design is working at 60 MHz system clock. The execution time using hardware acceleration is acceptable and can be applied to image processing applications that do not need fast processing (33 ms correspond to 31 frames/s). Also, better improvement in processing time is possible via different FPGA platforms having higher operating frequency. Finally, we showed that our HW/SW solution improves considerably the interpolation speed (258 times faster) compared to the software solution.

7. References

[1] Wing-Shan Tam1,2, Chi-Wah Kok1, Wan-Chi Siu1, A MODIFIED EDGE DIRECTED INTERPOLATION FOR IMAGES, 17th European Signal Processing Conference (EUSIPCO 2009) Glasgow, Scotland, August 24-28, 2009.
[2] Ortwin Franzen*a, Christian Tuschenb and Hartmut Schrödera "Intermediate image interpolation using polyphase weighted median filters" Proc. SPIE 4304, 306 Nonlinear Image Processing and Pattern Analysis XII USA 2001.
[3] HOMEM, M. R. P. ; MASCARENHAS, N. D. D. . A Statistical Approach for Image Interpolation. In: XX Brazilian Symposium on Computer Graphics and Image Processing, SIBGRAPI 2007.
[4] Zhongmou Wu, Yun He Combined adaptive-fixed interpolation with multi-directional filters International Journal, Signal Processing : Image Communication Vol 24 277–286 Elsevier 2009.
[5] K.T. Gribbon and D.G. Bailey "A Novel Approach to Real-time Bilinear Interpolation" Proceedings of the Second IEEE International Workshop on Electronic Design, Test and Applications DELTA 2004.
[6] Lazhar Khriji, Faouzi Alaya Cheikh, Moncef Gabbouj "High-resolution digital resampling using vector rational filters" Optical Engineering. 38(5) 893–901 May 1999.
[7] Stratix II device, http://www.altera.com/products/devices/stratix-fpgas/stratix-ii/stratix-ii/st2-index.jsp
[8] Nios II Processor http://www.altera.com/devices/processor/nios2/ni2-index.html
[9] Altera Development Board
http://www.altera.com/products/devkits/altera/kit-niosii-2S60.html
[10] A. Boudabous, A. Ben Atitallah, P. Kadionik, L. Khriji and N. Masmoudi "FPGA implementation of vector directional distance filter based on HW/SW environment validation" AEU International Journal of Electronics and Communications, Elsevier. Vol 65, N° 3, Pages 250-257, March 2011.
[11] http://compression.ru/video/quality_measure/video_measurement_tool_en.htm


(a)

(b)

(c)

Fig. 6. (a) Original decimeted images, (b) Interpolated image using ideal Border-Preserving Interpolator (c) Interpolated image using HW/SW Border-Preserving Interpolator implementation