Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 35 (2012) 223 – 229

Procedia Engineering

www.elsevier.com/locate/procedia

International Meeting of Electrical Engineering Research
ENIINVIE 2012

Digital filter implementation over FPGA platform with LINUX OS

Jorge I. Galvan-Tejada[a], Carlos E. Galvan T.[b], Jose M. Celaya-Padilla[a], J. Ruben Delgado C.[c]

[a]*Instituto Tecnológico de Estudios Superiores de Monterrey*
BioInformatics, Monterrey, Mex.
[b]*Universidad Autonoma de Zacatecas*
UAIE, Zacatecas, Mex.
[c]*Instituto Tecnológico de Estudios Superiores de Monterrey*
Ambience Intelligence, Monterrey, Mex.

Abstract

The embedded processors on FPGA's are a good tool to specific propose works. In this work we present how the FPGA is used to apply a Sobel filter to a set of images, also the step needed to set-up the entire system is described. An embedded processor, with a Linux distribution implemented is used to run a special compilation of C filter program, the filter is compared with the results obtained with a PC running the same filter, in the embedded system all the process runs in the FPGA and the exit file can be accessed by ftp or http server embedded into the Linux system.

Keywords: C, Digital filter, FPGA, Image process.

1. Introduction

The embedded systems complexity is growing fast, the capability of reconfiguration makes it very interesting. The FPGAs are now capable to implement whole OS, some linux distributions can be stored usually in a flash card and run on a processor embedded into the FPGA [1].

This systems have some advantages being reconfigurable, this is great to implement processors and make test over them before a solid state production, also the relative low cost allow to use them in the academic ambience, the research groups are capable of develop tools for the industry at lower prices [2, 3].

The goal of this work is to develop a different alternative to the traditional technique for image processing in embedded systems [4] Our proposal consist in create a soft core processor embedded in to the FPGA, once we have done this, a classical distribution that is used in FPGAs was modified to allow running image processing algorithms in standard "C" based language [5].

Email address: `gatejo@uaz.edu.mx` (Jorge I. Galvan-Tejada)

1877-7058 © 2012 Published by Elsevier Ltd. Open access under CC BY-NC-ND license.
doi:10.1016/j.proeng.2012.04.184

This work is intended to offer a faster alternative for prototyping or testing image digital processing, without need of recompile the kernel of linux distribution or in a worse case develop the VHDL/VERILOG application language equivalent.

The relevance is in the academic area, where the students have limited time to implement the filters, due the complexity and time consuming of coding such algorithms in HDL, as we can see in [4], programming an algorithm can take hundreds of hours, mean while the same algorithm in plain c can take just a fraction of that time.

Nomenclature

FPGA	Field-Programmable Gate Arrays
OS	Operating system
HDL	Hardware design language
DICOM	Digital image and communication in medicine

2. Background

2.1. FPGA

FPGAs are reconfigurable devices [6], which contain programmable logic components called ”logic blocks”, and a hierarchy of reconfigurable interconnects that allow the blocks to be ”wired together” in different configurations. Logic blocks can be configured to perform complex combinational functions, or merely simple logic gates like AND and XOR digital operators. In most FPGAs, the logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory, FPGAs are capable to perform complex techniques such as parallelism and pipe-lining.

FPGAs have traditionally been configured by hardware engineers using a Hardware Design Language (HDL). The two principal languages used are Verilog HDL (Verilog) and Very High Speed Integrated Circuits (VHSIC) HDL (VHDL) which allows designers to design at various levels of abstraction. Verilog and VHDL are specialized design techniques that are not immediately accessible to software engineers, who have often been trained using imperative programming languages. Consequently, over the last few years there have been several attempts at translating algorithmic oriented programming languages directly into hardware descriptions.

2.2. Microblaze

The MicroBlaze embedded processor soft core is a reduced instruction set computer (RISC) optimized for implementation in Xilinx Field Programmable Gate Arrays (FPGAs).

The MicroBlaze soft core processor is highly configurable, allowing you to select a specific set of features required by your design.

The fixed feature set of the processor includes:

- Thirty-two 32-bit general purpose registers
- 32-bit instruction word with three operands and two addressing modes
- 32-bit address bus
- Single issue pipeline

In addition to these fixed features, the MicroBlaze processor is parametrized to allow selective enabling of additional functionality.

2.3. Sobel filter

As [7] mention one of the biggest part of Image Processing is edge detection, due that they represent valuable information of objects frontiers, and can be used to segment the image, detect objects, etc. and edge can be defined as transition of one gray level to and other, theoretically this transaction must be and step function, but this condition never happen due blur effect, blooming effect, noise, soften image etc., in order to over come this problems, an edge detector algorithm has to be implemented, such as sobel, morphology, Prewitt, Laplace, Kirsh, etc.

The majority of this algorithms are based in the first and second derived, in bi dimensional functions $f(x, y)$, the derived is the vector that points to the maxima variation of $f(x, y)$, and the module is proportional to the variation, this vector is called gradient and can be defined as:

$$\nabla f(x,y) = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix} \tag{1}$$

$$Mag[\nabla f(x,y)] = \sqrt{\left(\frac{\partial f(x,y)}{\partial x}\right)^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2} \tag{2}$$

From equation 2 we can obtain the derived approximation, witch will indicate the abrupt changes in the image:

$$\nabla_x f(x,y) = f(x,y) - f(x-1,y) \tag{3}$$

$$\nabla_y f(x,y) = f(x,y) - f(x,y-1) \tag{4}$$

The Sobel operator 5 compute the gradient of the intensity at each point, the direction of the highest possible increase 7 , and the value of the change in the direction, represented by a vector 6, this can be interpreted as the area with a high gradient as a edge, and the area with lower gradient, as a background of the image.

$$\mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \tag{5}$$

$$\mathbf{G} = \sqrt{\mathbf{G_x}^2 + \mathbf{G_y}^2} \tag{6}$$

$$\mathbf{\Theta} = \arctan \frac{G_y}{G_x} \tag{7}$$

2.4. Xillybus

In a recent years, some projects have remarked interest using linux operating system (OS), this projects includes some serious agencies as NASA, that has sponsored the FlightLinux Project, with the intention of demonstrate the reliable linux OS on the UoSAT12 Satellite [8]. Linux is one of the most prominent examples of free and open source software collaboration: the underlying source code may be used, modified, and distributed by anyone under licenses such as the GNU General Public License. Typically Linux runs on a wide variety of computer hardware, including mobile phones, tablet computers, network routers, televisions, video game consoles, desktop computers, mainframes and supercomputers.

Typically Linux is packaged in a format known as a Linux distribution for a specific purpose such as desktop and server use, Xillybus distribution is a FPGA oriented distribution kit, which allows you to run a functional Linux system on the Microblaze FPGA soft processor, using the Xilinx SP605 hardware evaluation kit for Spartan-6. All components necessary to build it are available for download at no cost, partly from this site, and partly from Xilinx.

The steps to reaching a functional system include installing pieces of software, and perform specific operations as described in detail. No prior knowledge on either FPGA nor Linux is necessary. Your computer may run either Windows or Linux for running this through.

This distribution allows you to

- Run a Linux system which is ready for working with, including a DHCP client, telnet, ftp, a small web server, mounting of NFS as well as Windows (SMB) shares. The Compact Flash card will also serve as the local disk which can be written to.
- Easily compile your own user-space applications in with GNU make and gcc. Compilation is possibly dynamic against libraries which are part of the distro.
- Easily develop your own Linux-reachable peripherals on the FPGA using the Xillybus IP core, and transport data between the FPGA and Linux user space applications with minimal effort.

3. Methodology

In order to make this work two different process were prepared, one was the FPGA set-up, this included the installation of the Linux OS and the and FAT configuration. On the other hand we develop, the Sobel filter in plain ”c” and a set of images to test the functionality.

3.1. FPGA setup

First we download the Xillybus Linux distribution, this distribution is specially designed to run over FPGA microblaze embedded processor. The Xillybus distribution has a project to create the *.bit* file, this file configure and create the microblaze with the necessary resources to run the Linux version (This version runs only in SPARTAN 6 by Xilinx).

Once the processor is embedded, in a compact flash memory (Sandisk or Kingston are recommended)we create a 2 Gb partition (The partition can't be larger because the FAT system required), in order to the FPGA can read this card, the partition must have a FAT 16 file system. The ISO of the Xillybus was loaded into the compact flash partition. When the iso is loaded some other partitions are created in the flash memory, with Xilinx design tools the FPGA design (.bit) file and the Xillybus kernel were merged to create the ACE file in order to boot from the FPGA, this file must be copied to the special FAT 16 partition created during the ISO expansion.

Using a PC with LINUX system, we connect to the FPGA over SSH (Secure shell) using PUTTY terminal, with the PC the permissions were modified to allow write in the flash memory, this is necessary in order store the output file, the figure 1, shows the Microblaze running on the FPGA.

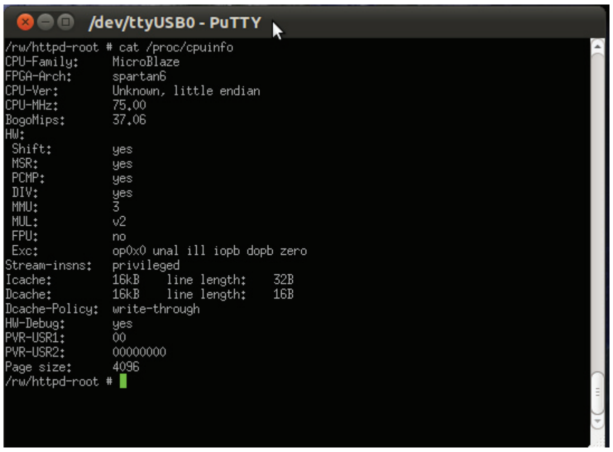
3.2. Sobel filter and images

The Sobel filter is a well known filter to border detection, this filter uses a convolution of a 3x3 kernel along the image, witch is a higly computer task, this filter was programmed in C, compiled over the Linux platform using crossover compiler to be able to run the program in the FPGA linux system.

The images files were transferred using the ftp embedded in the linux kernel running in the FPGA. The set of 4 images with a size of 256 x 256 pixels and a gray scale format of 8 bits.

First, this images were filtered with the same C code over a PC platform with Windows OS, 2.4 Ghz Processor and 1 Gb RAM memory, this process to check the computing time and check if the filter do a good job detecting the borders on the image.

Then, on FPGA all the images were filtered and the final exit files were accessed by the HTTP server included into the linux on the FPGA .



```
/rw/httpd-root # cat /proc/cpuinfo
CPU-Family:     MicroBlaze
FPGA-Arch:      spartan6
CPU-Ver:        Unknown, little endian
CPU-MHz:        75.00
BogoMips:       37.06
HW:
 Shift:         yes
 MSR:           yes
 PCMP:          yes
 DIV:           yes
 MMU:           3
 MUL:           v2
 FPU:           no
 Exc:           op0x0 unal ill iopb dopb zero
Stream-insns:   privileged
Icache:         16kB    line length:    32B
Dcache:         16kB    line length:    16B
Dcache-Policy:  write-through
HW-Debug:       yes
PVR-USR1:       00
PVR-USR2:       00000000
Page size:      4096
/rw/httpd-root #
```

Fig. 1. a) MicroBlaze processor runing in a FPGA

4. Results

After the experimentation the FPGA filtered images show a good edge detection, the times of computing are shown in the table 1. Also in the figure 2 we present the final images used in the filter, and the exit file obtained from the FPGA process filter.

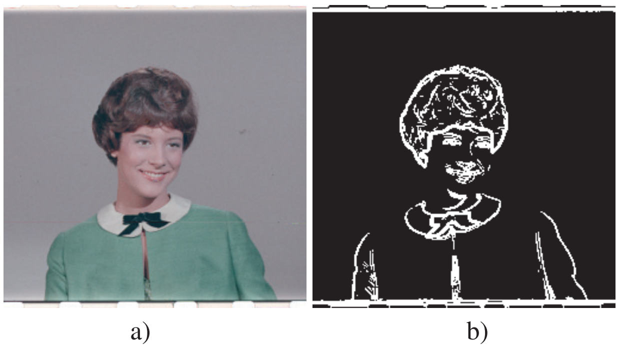


Fig. 2. a) Original image b) Filtered image

All the images present the expected results after filtering in both pc and FPGA, with this we prove that is indeed possible run complex filter with a zero or minimum change from the original plain c program.

5. Conclusions and future work

As we presented in this paper is possible to use a FPGA for a very complex tasks like running linux OS which is a x86/x64 optimized system that takes advantages of a embedded instructions like SS1,SS2,MMX. On the processor itself, and even multicore optimizations. However we demonstrated that even with the limitations of the FPGA, has a very good performance on a highly computer algorithm like a image processing.

As we demonstrate is less complicated implement Linux OS to the FPGA and this way to test algorithms for image processing, this is very helpful in the academic area, this reduces the time to the students to go from the theoretical filter to the implementation and test, also is faster to change algorithm programming it in C language than VHDL or Verilog.

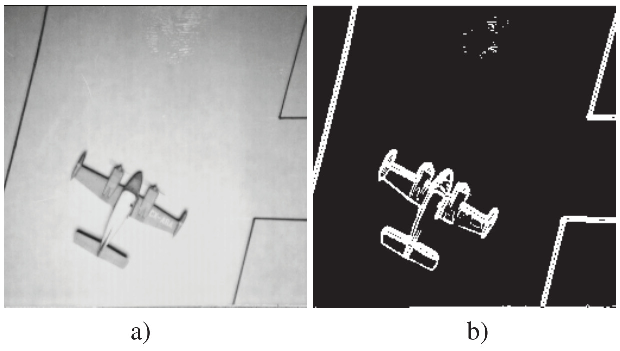


Fig. 3. a) Original image b) Filtered image

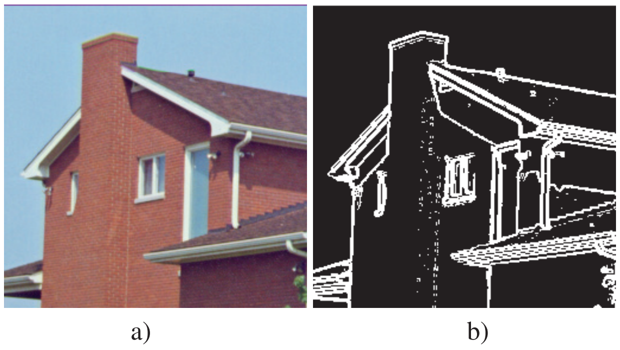


Fig. 4. a) Original image b) Filtered image

In table 1 we present some computing time results, the FPGA appears to be very slow, but if we consider the total Mhz in the processor and the total memory amount in the FPGA, it is not to slow, in fact it could be improved with some parallel computer work.

In order to implement a big cluster of medical images processing, this work is very handful use C language to create the filters. The medical images DICOM format is not compatible with the most of the regular image software.

The DICOM is a complex multi layer format, use this kind of images with VHDL or Verilog could take days of work to read it, using this method is possible to work with it in the FPGA, and the computing time could be improved using parallel computing with several FPGAs.

6. Acknowledgement

The authors thank Movie.com for partially financing the research. Jorge I. Galvan-Tejada, Carlos E. T. Galvan, thank PROMEP for the support of the project.

Table 1. Results of computed times

Processed image	Time in seconds FPGA	Time in seconds PC
Image 1	176.93	0.015
Image 2	178.88	0.0156
Image 3	180.03	0.0161

References

[1] F. Devic, L. Torres, B. Badrignans, Securing boot of an embedded linux on fpga, in: Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on, 2011, pp. 189 –195. doi:10.1109/IPDPS.2011.141.

[2] T. Hall, J. Hamblen, Using an fpga processor core and embedded linux for senior design projects, in: Microelectronic Systems Education, 2007. MSE '07. IEEE International Conference on, 2007, pp. 33 –34. doi:10.1109/MSE.2007.89.

[3] T. Hall, J. Hamblen, Using an fpga processor core and embedded linux for senior design projects, in: Microelectronic Systems Education, 2007. MSE '07. IEEE International Conference on, 2007, pp. 33 –34. doi:10.1109/MSE.2007.89.

[4] N. A. B. V. M. JDaggu Venkateshwar Rao*, Shruti Patil, Implementation and evaluation of image processing algorithms on reconfigurable architecture using c-based hardware descriptive languages, International Journal of Theoretical and Applied Computer Sciences 1 (1) (2006) 9–36.

[5] M. Riva, B. Esposito, D. Marocco, F. Belli, Embedding linux on fpga: An application on a real-time neutron/gamma discrimination system for fusion devices, in: Fusion Engineering, 2009. SOFE 2009. 23rd IEEE/NPSS Symposium on, 2009, pp. 1 –4. doi:10.1109/FUSION.2009.5226463.

[6] G. Afonso, R. Ben Atitallah, A. Loyer, J.-L. Dekeyser, N. Belanger, M. Rubio, A prototyping environment for high performance reconfigurable computing, in: Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2011 6th International Workshop on, 2011, pp. 1 –8. doi:10.1109/ReCoSoC.2011.5981497.

[7] J. Celaya-Padilla, Plate detection using morphological algorithms and edge statics, Universidad Autonoma de Zacatecas, 2011.

[8] J. Monson, M. Wirthlin, B. Hutchings, Fault injection results of linux operating on an fpga embedded platform, in: Reconfigurable Computing and FPGAs (ReConFig), 2010 International Conference on, 2010, pp. 37 –42. doi:10.1109/ReConFig.2010.79.