Reconfigurable Image Processor using an Fpga-Raspberry Pi Interface

Zalak Dave
Eduvance, Mumbai, India

Shivank Dhote
Vidyalankar Institue of Technology, MumbaiIndia

Pranav Charjan
Vidyalankar Institue of Technology, Mumbai India

Jonathan Joshi
Eduvance, Mumbai, India

Ganesh Gore
Eduvane, Mumbai, India

ABSTRACT
Image processing (I.P.) systems, involving multiple processing functionalities, use standard software tools to manipulate pixel values. The load on the system is high when these software tools are used for real time I.P. applications as the system they are running on are systems that are not specific to a particular application. This would require either high end hardware systems or an application specific hardware. Field Programmable Gate Arrays (FPGA) provide a cost effective custmizable solution. The Command Controlled Image Processor proposed in this paper provides a specific hardware based solution which is designed only for certain specific image processing tasks. This paper deals with the design and implementation of a multifunction processor with different modules using a FPGA. The design has been prototyped on a Xilinx Spartan 3E FPGA. The expected and achieved outputs have been given with comparison to standard MATLAB outputs. The hardware occupancies and delays have also been reported for different FPGA devices.

General Terms
MATLAB, Image Processing, FPGA

Keywords
Basys-2, Raspberry Pi, FPGA, Image Processing.

1. INTRODUCTION
Over 10 years ago, Xilinx Corporation introduced the first generation of Field Programmable Gate Arrays, or FPGAs [1][2]. These chips were designed to allow hardware manufacturers to include simple control logic in their products without having to resort to custom circuits. Essentially, the technology allows engineers to use software tools to specify hardware circuits. Although the technology was originally developed as an alternative to PALs and used for glue logic, there were early visionaries who perceived that the potential for FPGA technology was much greater than that. Even in the early stages of FPGA development, Papers were published that suggested this technology could be used for complex applications such as imaging. The key to FPGA technology is that it is reconfigurable. At any time, new software can be loaded into the chip that completely changes its character and function. Although the original FPGAs were relatively simple devices, this class of chip has grown in size and complexity to the point that today, complex algorithms can be implemented using FPGAs. The programming tools for these products, however, have not advanced to the same level as other, more mature technologies. As a result, creating software to run in an FPGA environment requires a high level of skill. Developers create schematics or a Hardware Description Language (HDL) representation of a design. The design is then compiled into a bit-stream which is loaded into the chip, rather than building a physical circuit. Advancements in FPGA technology have allowed it to become a viable alternative to other general purpose and specialized processors. FPGA represents the next step in computer design and control. For real-time computation, FPGA technology provided even more specialization and power. FPGAs continue to advance this process. For many applications, the use of FPGAs offers a faster, less expensive solution that is easier to upgrade as technology continues to move forward. In addition to higher speed and lower costs, the implementation of an FPGA solution requires fewer chips on a board. This allows a smaller footprint to be achieved as well as creating a highly customizable product. In addition, FPGA based system can be upgraded in the field by simply sending new code to run on the chips.

2. RELATED WORK
Traditionally, most real time image processing and machine vision systems used DSP based image processing boards. These products provided the horsepower necessary to process large amounts of data in real-time. General-purpose DSPs tend to support the largest common factor in all algorithms, with no regards for specific needs. As a result of this tendency, DSPs have largest required word widths, the most common memory addressing schemes, and generic arithmetic operations. For specific needs of image processing in defense applications the requirement of data width is 8 to 14 bit wide, the need for 32 bit wide data width is not there. Larger data width brings additional requirements of packing two or more adjacent bits into in word. Image processing rarely require higher data widths like floating point computations. Field Programmable Gate Arrays, or FPGAs, provide a programmable, high- speed solution that is both less expensive and more flexible than DSPs. It has been shown by Bosi [3], that FPGA can be used for specific needs of convolution where DSP has its limitations. The hardware implemented still uses DSP for data control and display. This approach brings in multiple hardware platforms where know how of each hardware is a must. With this criterion in mind, this paper explores the performance and architectural tradeoffs involved in the various image processing applications have been implemented on a FPGA. Work done in [4] and [5] are examples of such implementations. The aim of this work is to explore at a student level the aspect of implementing multiple image processing applications on a FPGA.

3. SYSTEM ARCHITECTURE
3.1 Overview
According to the block diagram in Figure 1 below there is a command interface which will be used to send necessary

commands to the processor depending on the output required. These commands are sent to the Raspberry Pi which is often called a credit card sized computer. The advantage is that Raspberry pi has GPIOs(general purpose Input Output)[6][7] pins which can be easily interfaced with the processor.

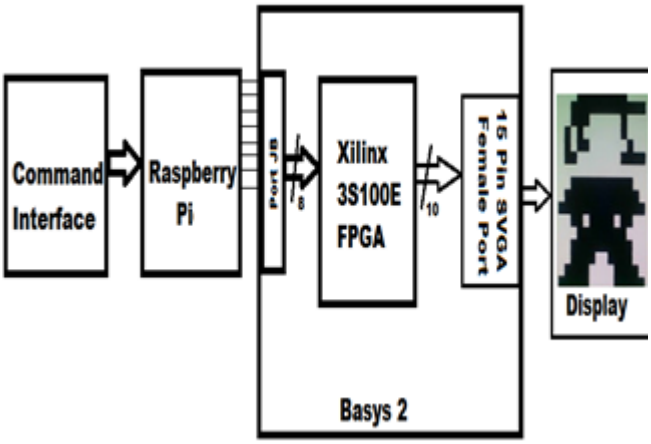

Figure1: The System Block Diagram.

The image processor is prototyped on a Basys-2[8] board by Digilent[8] which has the Xilinx Spartan 3E [9][10] FPGA on it. As shown in Figure 2 the board has ports which can be interfaced with the FPGA and required inputs can be sent to the FPGA. In the designing of the processor various algorithms are followed to achieve the various effects required on the image. Also the output from the processor is given to a VGA driver which is also developed on the same FPGA. The Basys -2 has a SVGA port[9][11]. The VGA driver is linked to the SVGA port on the board and this port can be connected to any supported VGA display device like a monitor or a projector.

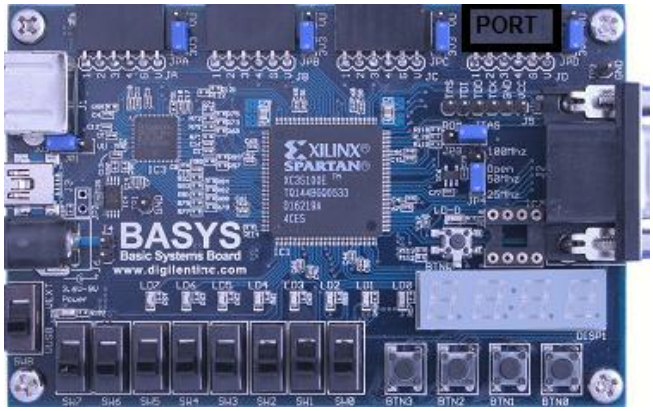

Figure 2: BASYS-2 Development Board.

3.2 The input output interface

The command interface and the R-pi are used to give a multifaceted approach to controlling the image processor. The aim is to give the user a more user friendly system as commands are given in spoken English. These are decoded by the R-Pi and hence signals are sent to the FPGA on the Basys 2 board using a binary coded system. The processor on the FPGA will perform certain operations on an image and will display the output on a monitor using the VGA driver which is developed on the FPGA as well. The R-Pi runs a python script which takes string input from the command line and drives the GPIO pins which are connected to the Basys 2. The script makes use of Python GPIO module to operate 4 GPIO pins as output pins. User interface used the Command Line Interface of the RPi. The user must run a .bat file which starts the script. Then user must enter the keyword of the operation to be performed on the FPGA in the command line. The script reads the keyword and provides the corresponding 4 bit value to the Basys 2 by means of output GPIO pins.

3.3 The Image Processor

The FPGA on the board is designed in a switch case pattern and hence depending on the signal received from the R-pi a specific operation would be performed on the already present image. According to Figure 2 above, shows the FPGA has basically been divided into 6 logic blocks and 1 VGA driver block. The outputs of every module is compared with the same effect reproduced using MATLAB.

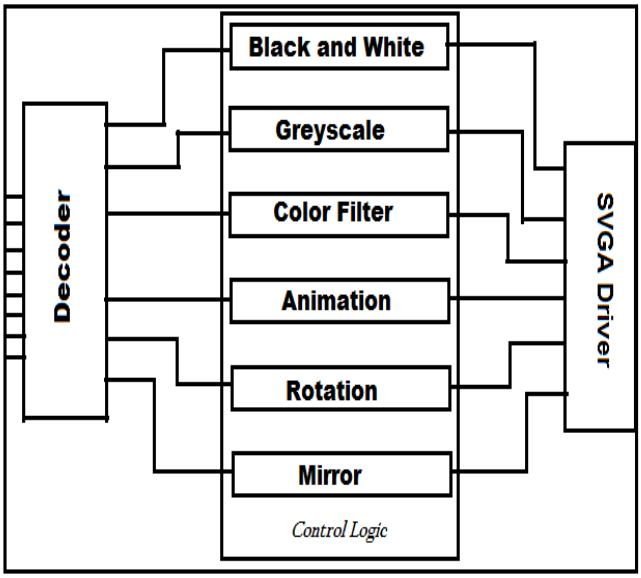

Figure 3 : The Block Diagram of the FPGA Design

3.4 The VGA Driver

The VGA driver is also written in Verilog which takes care of the Horizontal and vertical synchronization of the display with the FPGA. The VGA driver is written for a display of 640x480. Also as we have processed only 256 bit image the zoom of the display is also taken care by manipulating the input to the display.

4. RESULTS

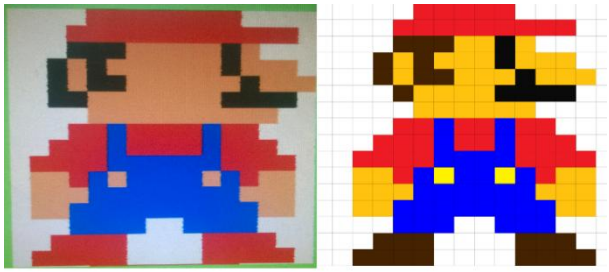

Figure 4:The Original Image On FPGA (left) and MATLAB

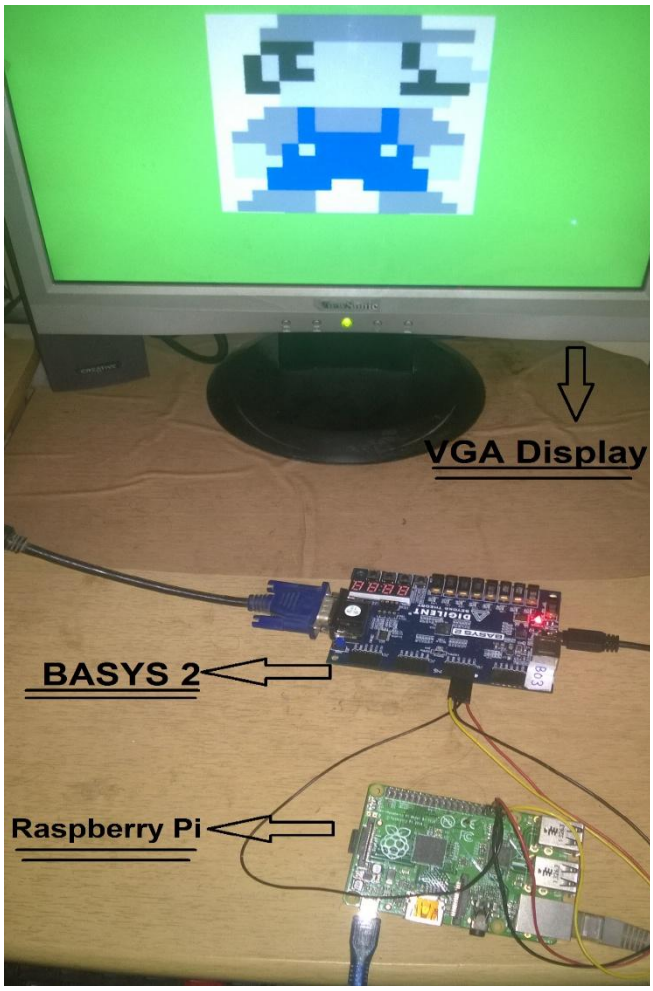

Figure 5: Physical Hardware Setup

As shown in Fig 5, the hardware setup includes a Rpi connected to the Basys 2 FPGA board with the result displayed on the monitor via the VGA cable. The setup is shown as proof of working design. Individual results are discussed in subsequent sections.

4.1 Output

4.1.1 Color to Black and White

In this block the concept of thresholding is used. Pixel values are compared with a specific value and then using a decision statement either the pixel is converted to black or white. The threshold value can be decided by the user. Fig. 4 shows the binary image.

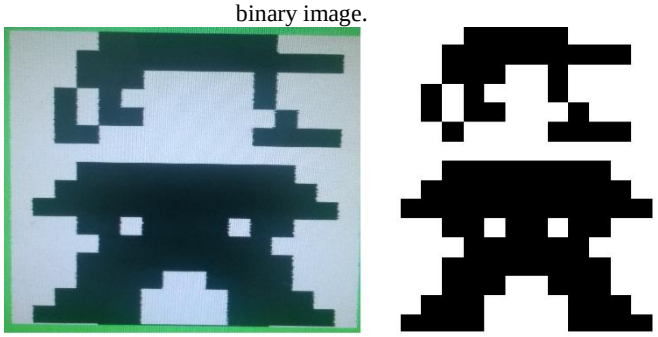

Figure 6: The Binary Image On FPGA(left) and MATLAB(right)

4.1.2 Color to grayscale

This module also uses thresholding but has different thresholds. The pixel value lying between two consecutive highest thresholds is assigned a shade lighter than black and hence a greyscale image is obtained by assigned different levels to every pixel value. Figure 5 shows a greyscale image produced by the processor (left) and MATLAB.

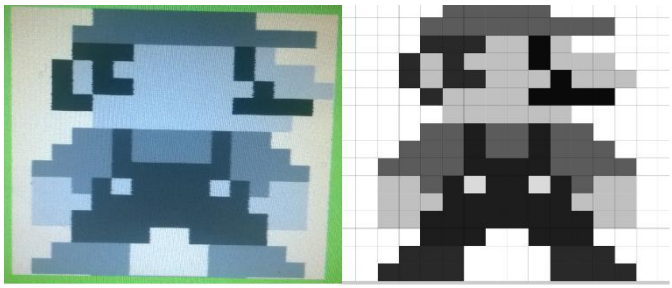

Figure 7: The Greyscale Image On FPGA(left) and MATLAB(right)

4.1.3 Color Filter

Pixels having one particular value are not processed and rest of the image is converted to greyscale. In Figure 5 it is seen that the blue color is left as it is and all other pixels have been converted to their respective greyscale values.

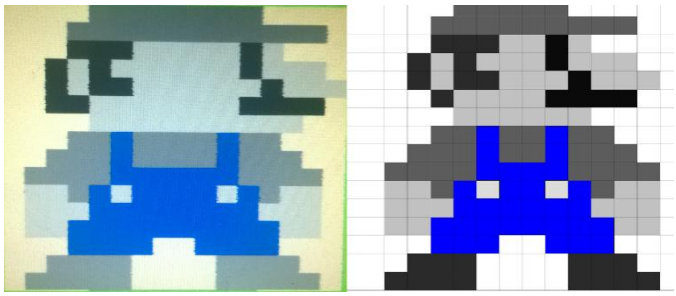

Figure 8: Color Filtered Image On FPGA (left) and MATLAB(right)

4.1.4 Rotation

The image is rotated by changing the order of bits that are being sent to the VGA driver. As the ROM is accessed from the start in normal display, we let the horizontal access be the same but change the order in which the vertical bits are accessed. In Fig 7. the image is rotated by 90 degrees and in Figure 8 shows the image is having a 180 degree rotation.

Figure 9: 180° Rotation On FPGA (left) and MATLAB(right)

4.1.5 Mirroring

The horizontal and the vertical addresses of the ROM are exchanged and then fed to the VGA driver circuit. This completely creates a mirrored image of the original. Figure 9 shows the mirrored effect on the display and on MATLAB.

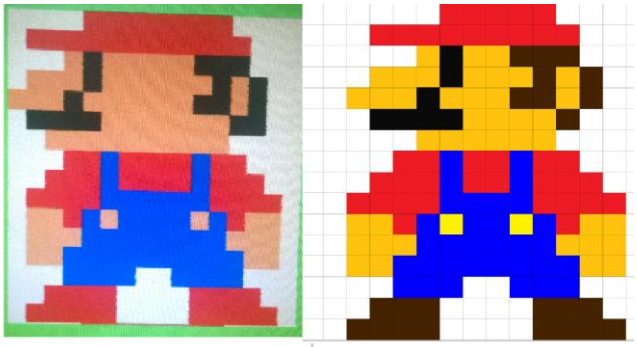

Figure 10: Mirroring on an FPGA (left) and MATLAB(right)

4.1.6 Animation

In this module the value of the ROM is shifted into another ROM such that the last 191 values are copied first and the rest of the memory is left empty. Then using a counter we feed the values of the two ROMs to the VGA driver one after the other. This is at more than 15 frames per second and hence gives a perception of motion. Figure 10 shows the two frames.

Figure 11: The Two Frames in the Animation

4.2 Results Discussion

Based on the results shown above it is seen that the processing done by the FPGA produces results that are at par with MTLAB. The results do show that the algorithm though simple can be efficiently implemented and deployed on a portable and fast platform like the FPGA. Also given below are the processing times and hardware occupancies for the given implementation.

Table 1: Processing times for algorithm implementation using MATLAB

Sr. Number	Process	Time(sec) 2.2GHz(i3)	Time(sec) 3.1 GHz(i5)
1.	Gray Scale	0.033192	0.019982
2.	Black and White	0.027514	0.019653
3.	Color Filter	0.028218	0.019826
4.	Image Mirror	0.030529	0.018255
5.	Image Rotation	0.25615	0.018200

Table 1 presents the processing times for the various algorithms when implemented on a general purpose computer running MATLAB. This is to give a general idea on the implementation times that software would take. In comparison to Table 2 shows the implantation time (delay) for the algorithms. Delays have been reported for various FPGA devices for a more general understanding of the performance of the given algorithms. The FPGA shows a delay of **5.12 microseconds**

Table 2: Algorithm path delays for various FPGA devices

Device Name	Delay (nanoseconds)
Spartan 3	19.297
Spartan 6	5.468
Virtex 4	9.663
Virtex 5	4.474
Virtex 6	3.188

Table 3 shows the device utilization of the implementation across various FPGA devices. It is known that FPGA utilization is measured in terms of number of Configurable Logic Blocks (CLBs) used on the chip. Results show that the design comfortably fits on all devices as shown.

Table 3: Device utilization across various FPGA devices

Device Name	Consumed CLBs (%)
Spartan 3	9
Spartan 6	0.35
Virtex 4	13
Virtex 5	0.51
Virtex 6	0.068

5. CONCLUSION AND FUTURE WORK

In this paper the authors have discussed the design and implementation of a multi-function image processing system on a FPGA. It has been successfully demonstrated that the results are at par with a commercial software like MATLAB. Results have also been reported about hardware (FPGA CLB) utilization of our design on different FPGAs. The device delays reported also match up to real time processing speeds as shown in comparison of different FPGAs.

6. REFERENCES

[1] Trimberger S. M, "Field – Programmable Gate Array Technology", Kluwer Academic Publishers, 1995.

[2] R. C. Gonzalez and P. Wintz, Digital Image Processing - Second Edition,Addison- Pearson Education Publishing Company, 2005.

[3] B. Bosi, G. Bois and Y. Savaria, "Reconfigurable Pipelined 2D Convolver for Fast Digital Signal Processing", IEEE Trans. On VLSI Systems, Vol. 7, No.3, Sept. 1999.

[4] Jonathan Joshi, Kedar Karandikar, Sharad Bade, Mandar Bodke,Rohan Adyanthaya, "Multi-core Image Processing System using Network on Chip Interconnect".

[5] Jonathan Joshi, Nisseem Nabar, "Reconfigurable Implementation of Wavelet based Image Denoising"

[6] Available Raspberry Pi B+ Data Sheet: http://www.element14.com/community/servlet/JiveServlet/previewBody/65470-102-1-287848/Raspberry-Pi%20Technical%20Data%20Sheet.pdf.

[7] Available Raspberry pi B+ pin diagram: http://www.element14.com/community/docs/DOC-68203/l/raspberry-pi-b-gpio-40-pin-block-pinout

[8] Available BASYS2, Digilent Website: www.digilentinc.com/basys2

[9] Available Xilinx website: www.xilinx.com/support/index.html/content/xilinx/en/supportNav/silicon_devices/fpga/spartan-3e.html

[10] Xilinx DataSource™ CD-ROM, Rev 10 second quarter 2004.

[11] Xilinx Website: http://www.xilinx.com/training/fpga/fpga-field-programmable-gate-array.html.