Image Processing using FPGA

Shrikant Subhash Warghade
Manu Vyas
Mani Kunnathettu Rajee
Prof Alex Noel Joseph Raj (Project Guide)

School of Electronics Engineering (SENSE), Vellore Institute of Technology,
VIT University, Vellore
Tamil Nadu, India

Abstract—Field Programmable Gate Arrays (FPGA) have become a staple of the current innovation trend because of their flexibility and potential. A fast-growing area of FPGA implementation is Image Processing. In this paper, a ZYBO (ZYnq BOard) Zynq-7000 series has been used for image processing. The paper elaborates on the process of developing the system functionality through VHDL using the Xilinx Vivado 2015.1 software. The paper discusses the working of the system in detail. Furthermore the advantages of this method, over a fully software-based implementation i.e. using MATLAB for the same operations, have been discussed. The paper concludes by summarizing the important results.

Index Terms— ZYBO(ZYnq BOard), FPGA, real time image processing, IP,ASIC,HDMI, VGA

1 Introduction

In the recent years, image processing is a field that has garnered the interest of researchers. The reason behind it is that it has wide area of applications; ranging from home automation to security services and extending some military applications.

Parallel architecture and pipelined processors have been used for image processing traditionally. However they have proven ineffective for processing high resolution video with minimum delay. Of course, the delay involved in image and video processing has to as minimum as possible.

ASICs(Application Specific Integrated Circuit) provide a hardware solution. An ASIC is a microchip intended for a well-specified application. For example: A Digital Signal Processor (DSP) or a hand-held PC. ASICs are utilized as a part of an extensive variety of utilizations, including automation, control, ecological observation, Personnel Digital Assistant(PDAs) and so on.

Although they are very efficient for their intended purpose, there is a disadvantage to ASICs: they are hard-wired. Hence the flexibility is very low meaning that if an ASIC is required to perform any task other than what it is originally designed to perform, it will suffer from technical limitations. That is why Field-Programmable Gate Arrays (FPGAs) have been adopted in many applications, with increasing popularity. FPGAs are having Programmable Logic Blocks (PLBs) which provide high level of flexibility and have a configurable architecture in the form Lookup Tables (LUTs). This essentially means that the FPGA can be programmed to perform a wider variety of tasks more efficiently as compared to an ASIC[1]. The associated overhead is the complexity involved in using FPGAs; particularly the coding complexity.

For the purpose of image processing, we adopted an FPGA in this paper. In the implementation of the proposed system, the Zynq-7000 processor is used.

2 Proposed System Architecture

2.1 Architecture Overview

The Zynq SoC is divided into two separate subsystems: The processing system (PS), and the Programmable Logic(PL). Fig.1 shows an overview of the Zynq SoC architecture.

We use the HDMI source/sink port available on the board as the input. The VGA port is used as the output.

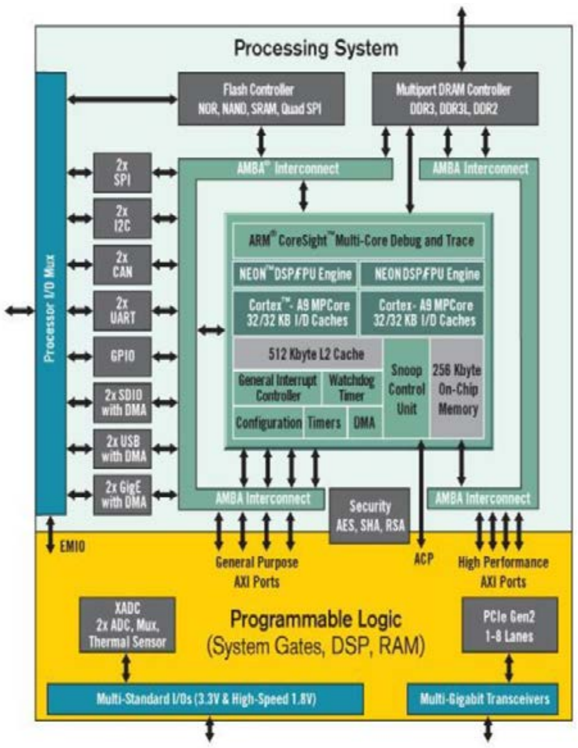
Processing System
Flash Controller NOR, NAND, SRAM, Quad SPI
Multiport DRAM Controller DDR3, DDR3L, DDR2
AMBA Interconnect
AMBA Interconnect
Processor I/O Mux
2x SPI
2x I2C
2x CAN
2x UART
GPIO
2x SDIO with DMA
2x USB with DMA
2x GigE with DMA
ARM CoreSight Multi-Core Debug and Trace
NEON DSP/FPU Engine
NEON DSP/FPU Engine
Cortex- A9 MPCore 32/32 KB I/D Caches
Cortex- A9 MPCore 32/32 KB I/D Caches
512 Kbyte L2 Cache
General Interrupt Controller
Watchdog Timer
Configuration
Timers
DMA
Snoop Control Unit
256 Kbyte On-Chip Memory
AMBA Interconnect
Security AES, SHA, RSA
AMBA Interconnect
EMIO
General Purpose AXI Ports
ACP
High Performance AXI Ports
XADC 2x ADC, Mux, Thermal Sensor
Programmable Logic (System Gates, DSP, RAM)
PCIe Gen2 1-8 Lanes
Multi-Standard I/Os (3.3V & High-Speed 1.8V)
Multi-Gigabit Transceivers

Fig.1. Zynq AP SoC Architecture

The Fig 2. shows the basic block diagram for an image acquisition and processing system. It includes a camera as an image sensor , ZYBO board(FPGA) as the processing element and a generic monitor as the output (fed by the VGA output of the ZYBO). The camera captures the image and that captured image will be given to ZYBO board through HDMI cable and the HDMI ink port on the board.

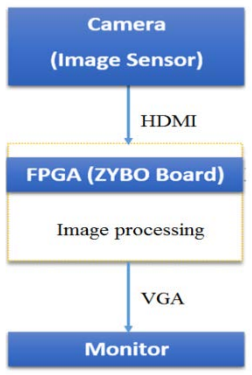

Fig.2. Image acquisition and processing block diagram

The ZYBO board will process the image and the processed image will be displayed on the monitor through VGA.

2.2 The Block Design

1) HDMI(High Definition Multimedia Interface)

HDMI (High-Definition Multimedia Interface) is an exclusive audio/video interface for exchanging uncompressed video information and compacted or uncompressed sound information from a HDMI-consistent source gadget. Almost all modern devices that deal with audio/video, use HDMI. Examples include video projectors, modern TVs, computerized sound devices such as home theatres etc.

The maximum pixel clock rate for HDMI 1.0 was 165 MHz, which is sufficient to allow image resolution up to1080p and WUXGA (1920×1200) at 60 Hz. In the proposed system we are performing image processing for 720p, by using the pre-defined settings available for the Intellectual Property (IP) blocks available in Vivado[2].

2)DDC(Display Data Channel)

The Display Data Channel(DDC) is a protocol that is used for the digital communication between the computer display and the graphics adapter. This enables the display to communicate its supported display modes to the adapter.

3)DVI to RGB[3]

This IP is used to interface raw Transition-Minimized Differential Signaling (TMDS) clock and data channel inputs as defined in DVI 1.0 specs for Sink devices. The IP decodes the incoming video stream and gives 24-bit RGB video data along with the pixel clock and synchronization signals recovered from the TMDS link as the output for further use.

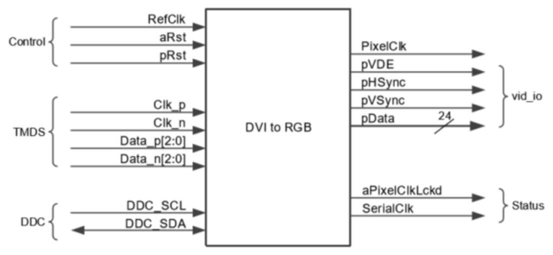

Fig.3. DVI to RGB

4) RGB to VGA[4]

The input for this IP is Xilinx vid_io input. The output is an independently customizable color depth. The output is correctly blanked for the RGB pixels so that it may be connected to the VGA DAC.

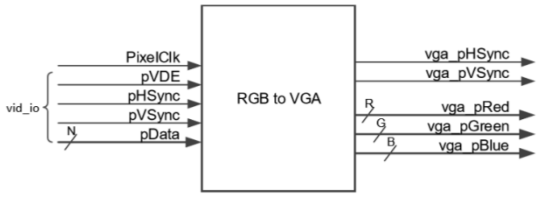

Fig.4. RGB to VGA

5) AXI Bus

Xilinx is using Advanced eXtensible Interface(AXI) protocol for implementation Intellectual Property(IP) cores. AXI is a part of ARM AMBA.

Currently there are three types of AXI4 interfaces available :

1. AXI4
2. AXI4_Lite
3.AXI4-Stream

We have used the AXI-4 interfacing the project.

3 Methodology

The video stream incoming from the HDMI is decoded into bits. Each of the color channels i.e. R, G and B are represented by 8 bits per pixel. So in total, for every pixel, there are 24 bits of data to be processed.

In the VHDL code, we are able to manipulate each of these 24 bits to achieve desired effects. Such bit-level access enables us to exercise a high degree of freedom with the code. For our current application, the following points are of importance –

i. The input and the output of the IP has to be 24 bits, to maintain the compatibility with the rest of the system. So both the input and output ports have been defined as STD_LOGIC_VECTOR (23 downto 0).

ii. In between the input and output, we implement the MUX-based design. The interfacing of the switches enables the user to select the desired output.

iii. The 24 bits of the pixel are divided into 8 bits per color channel, before any processing is done. This is to facilitate color-specific manipulations.

iv. For the grayscale and thresholding functions, the data type is converted into INTEGER type before further processing. This is so because averaging of the RGB values is required for grayscale and subsequent thresholding; and division cannot be performed on STD_LOGIC_VECTOR. After the processing is done, the data is converted back to STD_LOGIC_VECTOR.

We intended to implement the following types of image processing capabilities with the ZYBO- Only Red, Only Green, Only Blue, Grayscale, Thresholding and Inverted image.

In order to make the system user-friendly and intuitive, we adopted a multiplexer-based design. The user may use 3 switches on the board to toggle between the various outputs. The fig.5 shows the block diagram of the 8:1 MUX for image conversion.

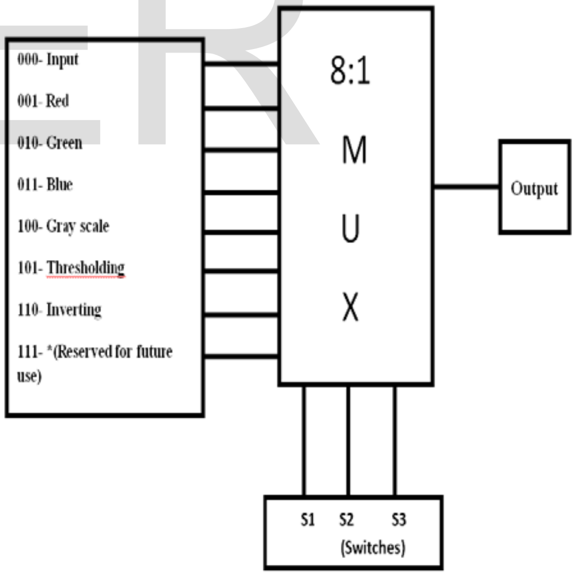

Fig.5. MUX for image conversion

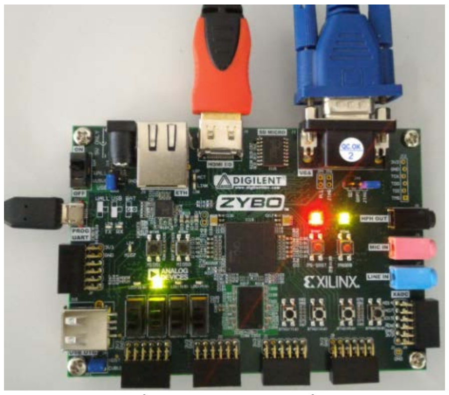
Fig. 6. ZYBO Board

The following table elaborates which type of output will be observed for which particular switch combination.

TABLE 1
FUNCTIONALITY OF THE MUX

FUNCTIONALITY	SWITCH COMBINATIONS		
	S1	S2	S3
Original Image	**0**	**0**	**0**
Only Red	**0**	**0**	**1**
Only Green	**0**	**1**	**0**
Only Blue	**0**	**1**	**1**
Grayscale	**1**	**0**	**0**
Thresholding	**1**	**0**	**1**
Inverted	**1**	**1**	**0**

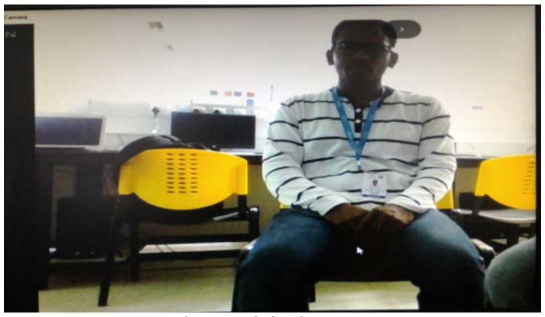
Fig.7. Original Image

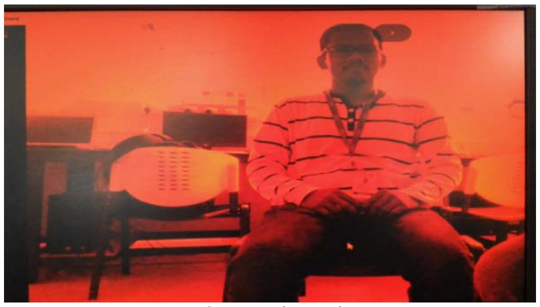
Fig.8. Only Red

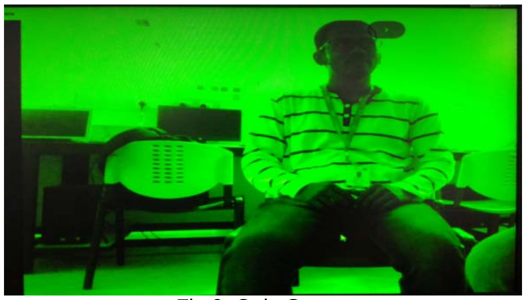
Fig.9. Only Green

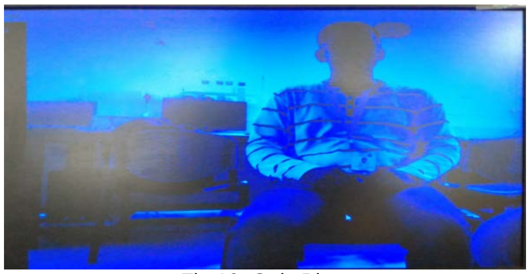
Fig.10. Only Blue

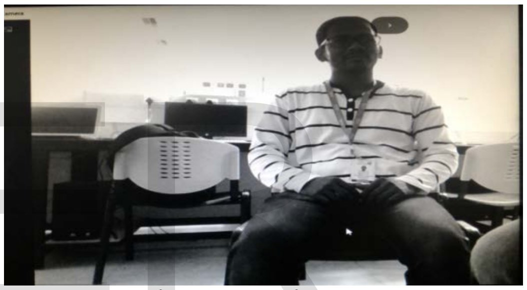
Fig.11. Grayscale Image

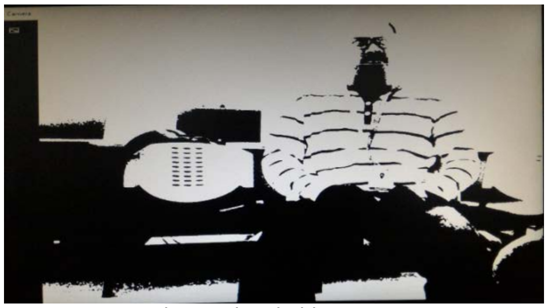
Fig.12. Threshold Image

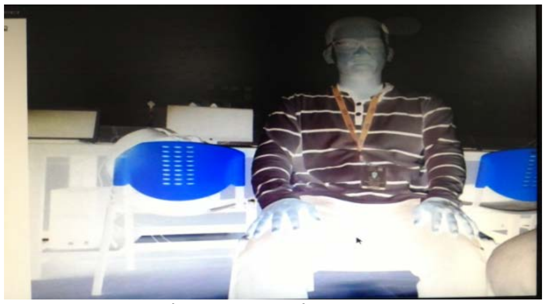
Fig.13. Inverted Image

4 Comparison with Software Implementation

In order to objectively compare the implementation of the same video processing functions in software and hardware, we used MATLAB. In particular, we wrote a simple MATLAB code to take the video from the webcam of a laptop. Then we used the rgb2gray() function to convert the video to grayscale.

For the same function, the VHDL code takes the input stream, averages the RGB color channel values, then puts the averaged value back into the RGB streams.

The frame rate is of prime importance for such an application. The camera feed was set to a resolution of 1280*720 i.e. 720p. The refresh rate of the monitor was 60Hz. For these conditions, we observed the following –

i. Frame rate for MATLAB implementation – 7 to 10 FPS (Frames per second)

ii. Frame rate for FPGA implementation – 30 FPS

4.1 Important highlights of hardware v/s software approaches

1)Speed – The speeds achieved in hardware approach are much faster than those achieved in the software approach.

2)Complexity – Hardware approach is much more complex to implement. Software approach is relatively simple to implement.

3)Suitability – The type of approach to be adopted is eventually guided by the type of application. For development and testing of a new algorithm or design, it is suitable that the software approach be used. For the final implementation, hardware approach can be used. If the output/response is required to be very fast, it is better to use dedicated hardware i.e. ASICs. Example – Hard real-time systems generally will preferably be developed using hardware, particularly ASICs. This is not to say that 100% of the functionality resides on the hardware; some functionality also resides on software. However the hardware counterpart dominates the software.

4)Cost – Hardware implementation is more expensive.

5 Conclusion

We have tried to implement simple and basic image processing functionality with the ZYBO FPGA in this project. In essence, this project serves as a base for further developments. The primary conclusion of the endeavors is that FPGAs can work as a great development tool, especially for high-speed image processing implementations. The latency observed in our application is very low.

The other important outcome is that such hardware implementation is better suited for deployment of the final design, rather than as a development tool for a new application, if the criticality of the application in question is high.

References

[1] FPGA based Multiprocessor Embedded System for Real-Time Image Processing, Nauman Masud, lahanzeb Nasir, Muhammad Shahid Nazir, Muhammad Aqil, 15th International Conference on Control, Automation and Systems (ICCAS 2015) Oct. 13-16,2015 in BEXCO, Busan, Korea

[2] www.xilinx.com/support/.../XUPZYBO/.../ZYBO_RM_B_V6.pdf

[3] www.digilentinc.com DVI-to-RGB (Sink) 1.6 IP Core User Guide Revised January 21, 2016; Author Elod Gyorgy

[4] www.digilentinc.com RGB-to-VGA (Sink) 1.6 IP Core User Guide Revised January 21, 2016; Author Elod Gyorgy

[5] Efficient Smart CMOS Camera Based on FPGAs Oriented to Embedded Image Processing from Sensors 2011, 11, 2282-2303; doi:10.3390/s110302282

[6] Real Time Image Processing based on Reconfigurable Hardware Acceleration by Steffen Klupsch, Markus Ernst

[7] FPGA Based ASM implementation for CCD CameraController R. Srinivasan, K. Anupama, Suneeta, S. K. Saha and Aditya Rao, Indian Institute of Astrophysics, Bangalore, 2009 International Conference on Emerging Trends in Electronic and Photonic Devices & Systems (ELECTRO-2009)

[8] An Embedded Architecture for Implementation of a Video Acquisition Module of a Smart Camera System, Jai Gopa\ Pandey*, Member, IEEE, Abhijit Kannakar, Member, IEEE, and Chandra Shekhar CSIR - Central Electronics Engineering Research Institute, Pilani, Rajasthan, India-333031