Reconfigurable Hardware for face detection

A Project Report

submitted by

Abinav Ramesh S (12EC04)

Mohan Rao D K (12EC59)

Pramod Kumar B (12EC73)

under the guidance of

Prof. Sumam David

in partial fulfilment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA

SURATHKAL, MANGALORE - 575025

April 20, 2016

DECLARATION

We hereby declare that the Project Work Report entitled **"Reconfigurable Hardware for face detection"** which is being submitted to the National Institute of Technology Karnataka, Surathkal for the award of the Degree of **Bachelor of Technology in Electronics and Communication Engineering** is a bonafide report of the work carried out by us. The material contained in this Project Work Report has not been submitted to any University or Institution for the award of any degree

Name of the Student	Register No.	Signature with Date
Abinav Ramesh S	12EC04	
Mohan Rao D K	12EC59	
Pramod Kumar B	12EC73	

Department of **Electronics and Communication Engineering**

Place : National Institute of Technology Karnataka, Surathkal

Date : April 2016

Certificate

This is to certify that the project entitled **"Reconfigurable Hardware for face detection"** submitted by :

(1) Abinav Ramesh S (12EC04),

(2) Mohan Rao D K (12EC759),

(3) Pramod Kumar B (12EC73)

as the record of the work carried out by them, is *accepted as the B. Tech Project Work Report Submission* in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Electronics and Communication Engineering**.

Guide

Prof. Sumam David

Professor,

Department of Electronics and Communication Engineering.

National Institute of Technology Karnataka, Surathkal

Chairman-DUGC

(Signature with Date and Seal)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

Major Project -II

End Semester Evaluation Report, April 2016

Course code : EC498

Course Title: Major Project -II

Title: *Reconfigurable Hardware for face detection*

Project Group:

Name of the Student	Register No.	Signature with Date
Abinav Ramesh S	12EC04	
Mohan Rao D K	12EC59	
Pramod Kumar B	12EC73	

Place: NITK Surathkal

Date: 21-04-2016 Name and Signature of Project Guide

ACKNOWLEDGEMENTS

We would like to thank Prof. Sumam David, for all her support throughout the project. We thank our examiner Dr. M.R. Arulalan for his support throughout the project. We also thank all the Department of Electronics and Communication Engineering, National Institute of Technology Karnataka, Surathkal for their support.

ABSTRACT

Modern digital systems are becoming more and more complex and it is required of designers to do more with less resources. In this regard FPGAs have become a very critical asset. FPGAs have provided the flexibility to perform reprogramming, but the cost, board space, and power consumption constraints demand a more efficient design strategy. Partial reconfiguration on FPGAs provides us an opportunity to meet these demands by allowing specific regions of the FPGA to be reprogrammed with new functionality while applications continue to run on other parts of the device.

Reconfiguration of FPGA's during run-time is a concept that has been known to researchers for many years, but this paradigm in FPGAs is seldom applied in mainstream applications. The main reason for this being the fact that not many algorithms readily render themselves to reconfigurability. But it is important for future work on FPGA's to exploit this capability as the early research carried out in this area have shown that reconfigurable architectures are capable of achieving 10 – 100 x computational density and reduced latency over conventional solutions.

We propose to use the power of reconfigurability coupled with an efficient hardware to perform face detection which will be able to complete the task with considerable speedup with respect to conventional pipelined implementation for the same.

Contents

List of Figures

Chapter 1
Introduction

In this chapter we define the problem statement of our project, give the motivation for choosing the given topic and present the previous work done in this field by others.

1.1 Problem definition

Reconfigurable architectures can bring unique capabilities to computational tasks. They offer the performance and energy efficiency of hardware with the flexibility of software. In some domains, they are the only way to achieve the required, real-time performance without fabricating custom integrated circuits. Their functionality can be upgraded and repaired during their operational lifecycle and specialized to the particular instance of a task.

The aim of the current project is to exihibit the superior performance of the FPGAs by providing hardware solution to a computational task. In particular, optimized hardware would be developed for the Viola−Jones Face Detection algorithm.

The algorithm would be implemented on an FPGA. The partial reconfiguration feature provided in Xilinx FPGAs would be exploited for developing area efficient Hardware. Partial reconfiguration feature would be used to time multiplex the FPGA resources between two stages of the Face Detection Unit.

1.2 Previous work

Face Detection is an interdisciplinary field that integrates techniques from pattern recognition, image processing, and computer graphics and vision and plays an important role in the areas of security and surveillance systems and human computer interfacing. A number of promising face detection algorithms have been developed and published. Among

the following stand out because they are often referred to when performance figures etc. are compared.

1. Neural Network-Based Face Detection[4]
2. A Statistical Method for 3D Object Detection Applied to Faces and Cars[5]
3. Robust Real-Time Objection Detection[6]

One of the most popular implementation and efficient of face detection is the Viola Jones [6] approach which uses asset of weak classifiers. The key elements of this method are the Adaboost learning and Haar Features which makes it rapid and robust to real world challenges. But at the same time this method is computationally expensive because of the need of applying Haar features calculation over the whole image.

Algorithms for face detection have also been implemented on HandelC[7], a C-based HDL. A multi-GPU implementation of the Viola-Jones face detection algorithm that meets the performance of the fastest known FPGA implementation has also been done.[8].

1.3 Motivation

Our motivation behind using reconfiguration is to leverage the flexibility of the FPGA by allowing specific regions of the FPGA to be reprogrammed while applications continue to run in the remainder of the device. This feature of FPGAs prove to be beneficial when compared to commercial ASICs.

The common bottlenecks while designing any ASIC are

1. Cost
2. Area
3. Power

Partial reconfiguration addresses these three fundamental needs by enabling the designer to:

1. Reduce cost and/or board space
2. Change a design in the field
3. Reduce power consumption

Thereby enabling the designer to design a system that is more cost effective and re-programmable when compared to an ASIC.

Partial reconfiguration enables designers to reduce the size of their designs by dynamically time-multiplexing portions of the available hardware resources. The ability to load functions as per requirement reduces the amount of idle logic, thereby saving space. In this project, a hardware architecture is implemented which lessens the resource usage of the FPGA and speeds up the process.

1.4 Overview

1.4.1 Software Training

The Software Training of the face detector was mainly carried out on MATLAB, it can be divided into two major phases :

1. Feature Extraction
2. Initial Testing

Feature Extraction

This phase consists of extracting the relevant haar features and grouping the extracted features into cascade stages to obtain the required level of accuracy.

Initial Testing

In this phase the accuracy of the cascade classifier was tested to ensure its proper working. The MATLAB function which carries out this function takes an input image and runs the

fixed 24*24 detector across the image looking for faces. If a face is detected, its presence is indicated with red dot at the centre of the corresponding face. Once the detector has finished parsing through the entire image, the image with marked faces is given as output.

1.4.2 Hardware Implementation

The Hardware detector was designed using C++ in Vivado High Level Synthesis tool, which performs all the tasks required for face detection, including pre-processing and post-processing of images.The detector was designed to take in an input image and provide an output image with faces detected and marked with a rectangle around it.

Chapter 2

Description

2.1 Viola-Jones Face Detector

The Viola Jones Face detection algorithm provides a robust framework for face detection that is capable of processing images at an extremely rapid rate.

There are three key elements to the implementation of this algorithm.

1. The transformation of an image into an Integral Image which allows quick computation of Haar features used by the detector.
2. A simple and efficient classifier built using the Adaboost algorithm to select a small number of critical features from a very large set of potential features.
3. A sequence of classifiers in cascade which allows an image to be quickly discarded if it is not a face.

Following is a brief description of the algorithm:

The transformation of an image into an Integral Image

The first step of the Viola-Jones face detection algorithm is to turn the input image into an integral image. This is done by making each pixel equal to the entire sum of all pixels above and to the left of the concerned pixel.

1	1	1
1	1	1
1	1	1

Input image

1	2	3
2	4	6
3	6	9

Integral image

Figure 2.1: Integral image

This allows for the calculation of the sum of all pixels inside any given rectangle using only four values,thus facilitating the sum of pixels within rectangles of arbitrary size to

be calculated in constant time. These values are the pixels in the integral image that coincide with the corners of the rectangle in the input image. This concept is illustrated in the following figure.

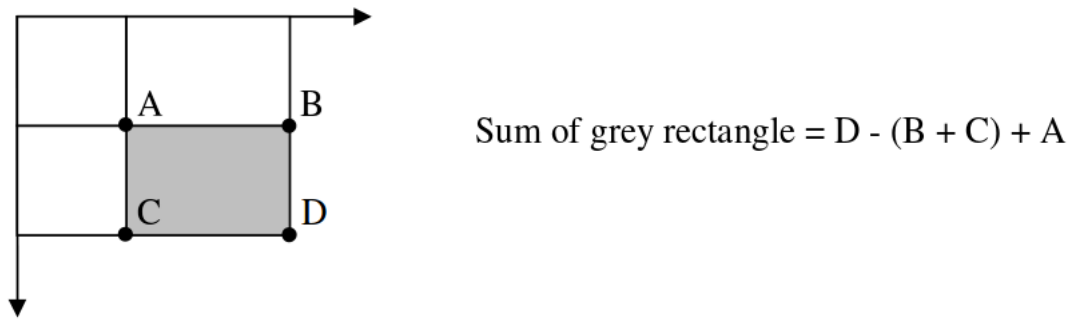

Figure 2.2: Sum Calculation

Adaboost Algorithm

The Viola-Jones face detector analyzes a given sub-window using features consisting of two or more rectangles.The different types of features are shown in the following figure .

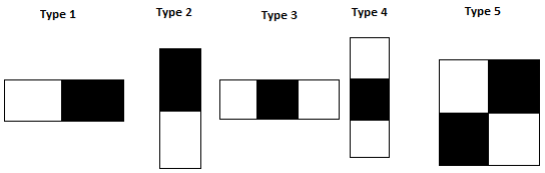

Figure 2.3: Haar Features

Each feature results in a single value which is calculated by subtracting the sum of the white rectangle(s) from the sum of the black rectangle(s). When allowing for all possible sizes and positions of the features in a 24*24 image, a total of approximately 160,000 different features can be constructed. Among all these features some few are expected to give almost consistently high values when on top of a face. In order to find these features ,the AdaBoost algorithm is used.

AdaBoost is a machine learning boosting algorithm capable of constru cting a strong classifier through a weighted combination of weak classifiers. Viola-Jones' AdaBoost algorithm is presented in pseudo code as follows.

- Given examples images $(x_1,y_1),...,(x_n,y_n)$ where y_1=0,1 for negative and positive examples.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for y_1=0,1, where m and l are the numbers of positive and negative examples.
- For t=1,...,T:

1) Normalize the weights, $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$

2) Select the best weak classifier with respect to the weighted error:

$$\varepsilon_t = \min_{f,p,\theta} \sum_i w_i \left| h(x_i, f, p, \theta) - y_i \right|$$

3) Define $h_t(x) = h(x, f_t, p_t, \theta_t)$ where f_t, p_t and θ_t are the minimizers of ε_t.

4) Update the weights:

$$w_{t+1,i} = w_{t,i} \beta^{1-e_i}$$

where $e_i = 0$ if example x_i is classified correctly and $e_i = 1$ otherwise, and $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$

- The final strong classifier is:

$$C(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where $\alpha_t = \log \frac{1}{\beta_t}$

Figure 2.4: Pseudo code of Adaboost algorithm

Cascade Classifier

The cascaded classifier is composed of stages each containing a strong classifier. The job of each stage is to determine whether a given sub-window is definitely not a face or maybe a face. When a sub-window is classified to be a non-face by a given stage it is immediately discarded. Conversely a sub-window classified as a maybe-face is passed on to the next stage in the cascade. It follows that the more stages a given sub-window passes, the higher the chance the sub-window actually contains a face.This concept is illustrated as shown in the following figure.

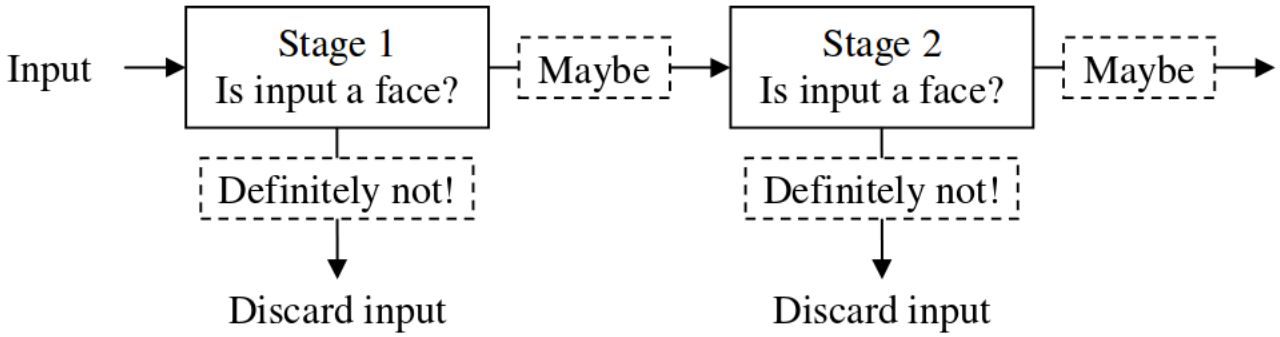

Figure 2.5: Cascade Classifier

2.2 Implementation

2.2.1 Feature Extraction

As planned, the Viola-Jones algorithm was implemented on MATLAB and the relevant Haar features were extracted from it for implementing the detection of faces on the FPGA.

As pointed out in the previous section the MATLAB program followed this sequence of steps:

1. Generation of all possible Haar features for images of size 24 x 24.
2. Generation of the integral images from the collected data set of faces and non-faces.
3. Calculation of feature value for all the images.
4. Choosing the appropriate threshold for each feature so that it can classify an image with least possible error.
5. Implementation of the Adaboost algorithm so that 256 optimal features may be chosen from the 162336 available features.
6. Choosing optimal set of features which gives maximum F_1 score

These extracted features were then used to synthesize the hardware detector in Vivado HLS.

2.2.2 Initial Hardware Implementation

In order to implement the face recognition system, it was initially decided to implement the entire project as a stand alone system without any operating system running on the ARM cores of the Zedboard.

This implementation of the system used the OV7670 camera to capture video on the input side and display the same on a VGA monitor.

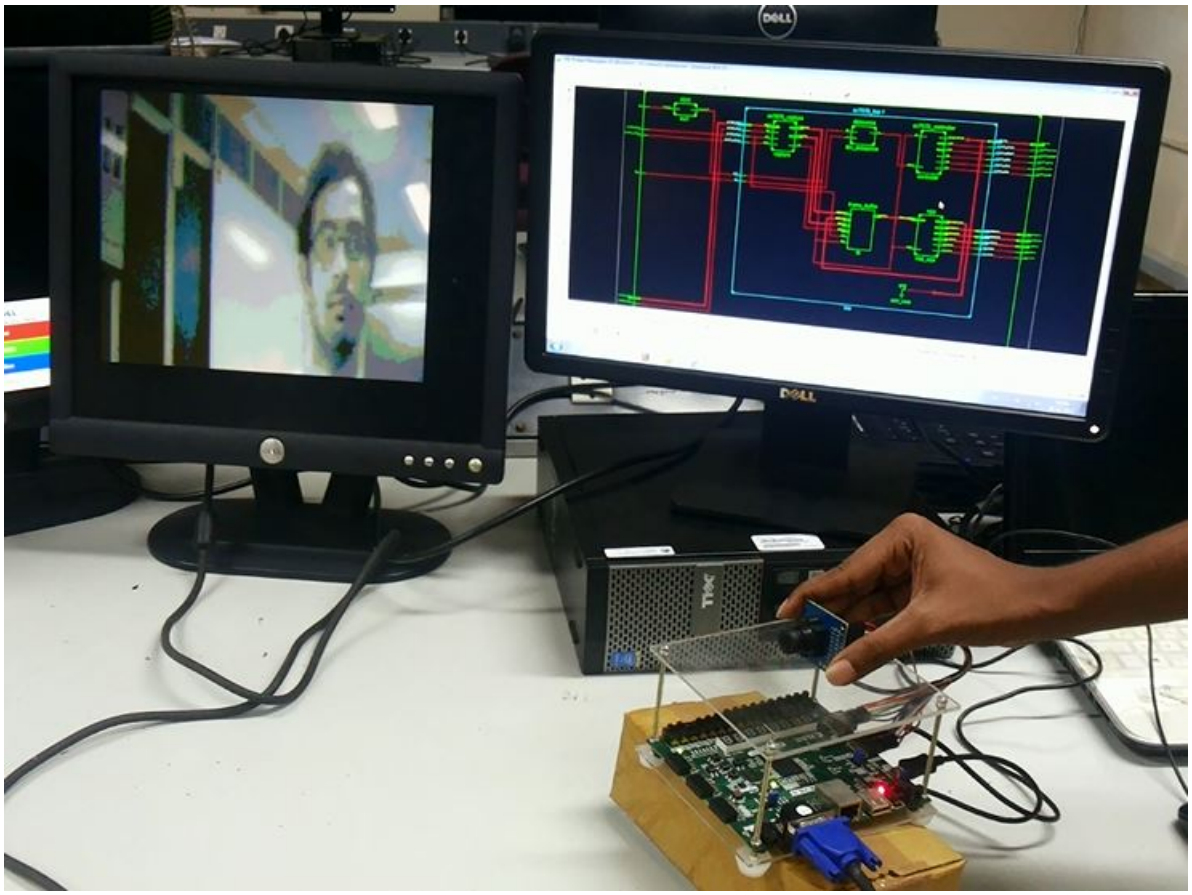

Figure 2.6: Bare Metal Implementation

This version of the hardware system consisted of the following blocks:

1. A VGA controller to display the video on the monitor
2. A frame buffer to store incoming data from the camera
3. An OV7670 controller to give appropriate signals to drive the camera
4. A capture module to decode the data from the camera and store it in the frame buffer

While implementing the face detection system in this fashion, certain disadvantages were encountered which made it unfeasible to implement such a system on bare metal.Some of the hurdles faced in implementing this version are outlined below:

1. The video output from the OV7670 camera has only 4 bit resolution.
2. Colour representation obtained from this camera is very poor.
3. Time required to interface the programmable logic with the processing system is large.

Due to the above mentioned disadvantages and to ensure the feasibility of implementing a hardware detector, Xillybus implementation was adhered to.

2.2.3 Revised Implementation

Due to the limitations of the Bare Metal Implementation outlined in the previous section the Xillybus IP Core was chosen to interface the detector and the I/O devices to a host running Xillinux on the ARM cores of the Zedboard. Our hardware detector was synthesized in C++ using Vivado HLS. A brief description of Vivado HLS is given below.

The revised implementation consists of the following three steps :

1. Synthesizing hardware detector in Vivado HLS
2. Combining the synthesized detector with the Xillybus IPCORE and Zynq processor in Vivado
3. Writing the Host program

Synthesizing hardware detector in Vivado High Level Synthesis

Vivado High-Level Synthesis (HLS) is a technique for utilizing programmable logic without using hardware description languages like Verilog or VHDL . It compiles a C/C++ function into logic elements, which can be implemented on FPGA.

The top level detector function (The inputs and outputs of that function is the interface between the PS and the PL) was written and some optimizations were performed to efficiently use the hardware on Zedboard.

These optimizations were leveraged by adding directives,which instructs the compiler how to synthesize that block of code to HDL. The directives are usually specified using pragmas.

The different directives used to optimize loops are as follows.

1. INTERFACE : This directive controls how the function arguments would be synthesized into RTL port.
2. PIPELINE : Reduces the initiation interval of the function by allowing the concurrent execution of operations within a function.
3. DEPENDENCE : Used to provide additional information that can overcome loop-carry dependencies.
4. RESOURCE : Specify which hardware resource (RAM component) an array maps to (if used on Array's)

Interfacing in Vivado

The synthesized detector was imported to VIVADO as an IPCORE ,which was interfaced with the rest of the Processing System (ZYNQ processor and other required I/O and Memory peripherals) and bitstream of the entire system was generated to be loaded on the FPGA.

Host program

The host program is a C++ program which takes care of the interface between the I/O devices and the FPGA logic(hardware detector in our case). Operations such as sending the input image from camera to the hardware and reception of output image from hardware is done using the host program.

In order to run the host program, Xillinux had to be installed on Zedboard. This required us to create an image of Xillinux on an SD Card and then the following four files were created and copied onto the SD card's first partition (the boot partition):

1. Linux kernel binary.
2. A .bit file which was created using Xilinx ISE tool to program the Programmable Logic on the Zedboard.
3. An initial bootloader, the file that contains the initial processor initializations.
4. A .dtb file or Device Tree Blob file, which contains hardware information for the Linux kernel.

Once the above files were copied onto the SD card, it could successfully be used to boot Xillinux on the Zedboard. After the installation of Xillinux, the OpenCV C++ libraries were installed inorder to run the hardware detector to perform the required tasks. Thus, a video input can be taken from a camera through an OpenCV function running on the host program and passed to the programmable logic for further processing and the output can be displayed on the monitor.

Working of Xillybus IP Core

Xillybus IP core is an integral part of the design as it is used for communicating between the Programmable Logic and the host program through a set of read and write FIFOs which can be accessed in the form of files from the host program. The following paragraph describes the functionality of Xillybus IP core and the steps followed to include Xillybus into the design.

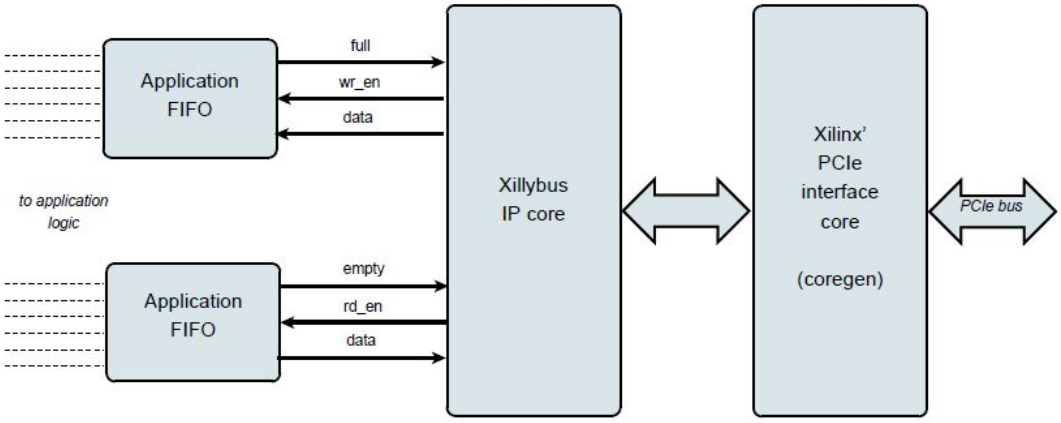

Figure 2.7: Xillybus Interface

The above figure depicts a simplified block diagram of the Xillybus IP, showing the connection of one data stream in each direction (to and from the detector). Writing data to the lower FIFO makes the Xillybus IP core sense that data is available for transmission in the FIFO's other end. The Xillybus reads the data from the FIFO and sends it to the host, making it readable by the userspace software. The data exchange mechanism is transparent to the application logic in the FPGA, which interacts with the FIFO.

Finally, the application on the computer interacts with device files that behave like named pipes. The Xillybus IP core and driver stream data efficiently between the FIFOs in the FPGAs and their respective device files on the host.

Chapter 3
Results

3.1 Software

The evaluation set is formed by using a set of faces and non-faces which were not used in the training phase(A total of 1000 faces and 1000 non-faces were included in the evaluation set).This set is used to test the classifier for different feature set and the one which minimizes the F_1 score is chosen. The output of the classifier with the chosen optimal set of features is shown below.

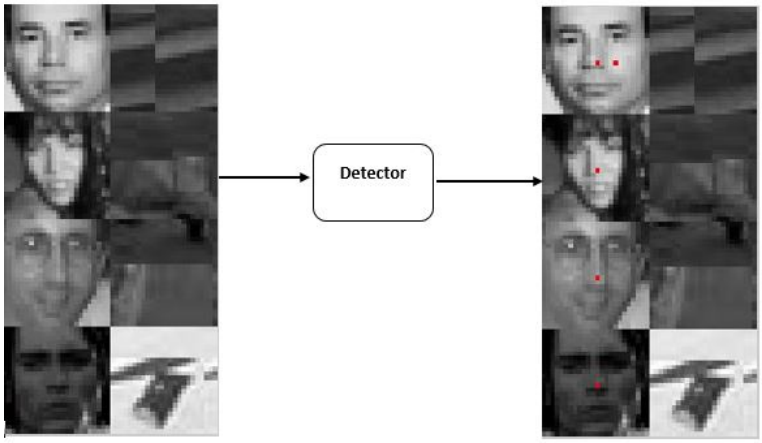

Figure 3.1: Output of face detector

Number of Test Cases = TC = 2000

Number of Negative Cases = P = 1000

Number of Positive Cases = N = 1000

		Actual	Class
		1	0
Predicted	1	True Positive = 924	False Positive = 191
Class	0	False Negative = 76	True Negative = 809

Precision = 0.8287

Recall = 0.9240

F_1 Score = 0.8738

Accuracy = 86.65%

3.2 Hardware

Below is the output of the hardware detector for a saved image and the frame is captured from a webcam.

Figure 3.2: Output of the hardware face detector

Although the hardware face detector detected most of the faces, the accuracy of the detector was not 100%.

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.74	1.25

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	2	-	-
Expression	-	3	0	6184
FIFO	-	-	-	-
Instance	-	33	2714	5411
Memory	144	-	21	6
Multiplexer	-	-	-	1736
Register	-	-	2950	-
Total	144	38	5685	13337
Available	280	220	106400	53200
Utilization (%)	51	17	5	25

Figure 3.3: Synthesis Report of hardware detector

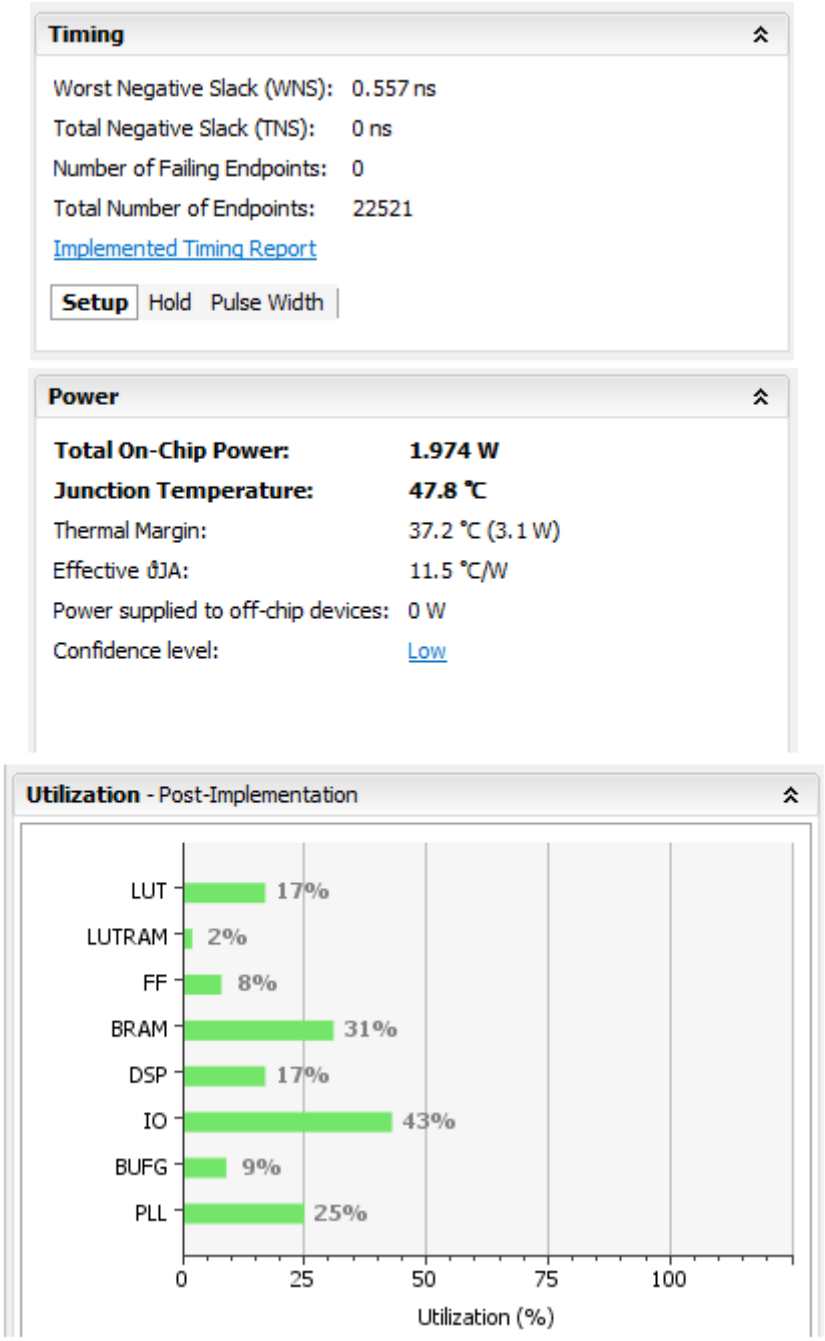

Figure 3.4: Overall Device Utilization

Chapter 4

Conclusions

1. A face detection system was implemented on the FPGA hardware without leveraging the concept of reconfigurability.
2. To overcome the difficulties of interfacing the I/O devices to the FPGA, an embedded version of Linux called Xillinux was installed on the Zynq processor present on the Zedboard.
3. Xillinux also includes an IPCORE called Xillybus which establishes a connection between Zynq processor and the FPGA Fabric.
4. The face detection system was found to work reasonably well for saved images.However due to the limited resources available on FPGA, we could only implement the face detection system for a low-resolution image.
5. Higher resolution images requires the use of the hardware multiple times for a single image.

4.1 Future Work

The following could be implemented to improve the currently implemented face-detection system.

1. Extend the hardware face detector to work in real-time.
2. Implement reconfigurability on FPGA.
3. Optimize the synthesized hardware detector inorder to process the input image faster.
4. Effectively utilize unused resources of FPGA inorder to support higher resolution images.

Bibliography

[1] Vivado High Level Synthesis Reference Manual, http://www.xilinx.com/support/documentation/sw_manuals/xilinx2014_3/ug902-vivado-high-level-synthesis.pdf.

[2] Xillybus for Zynq Reference Manual, http://xillybus.com/downloads/doc/xillybus_getting_started_zynq.pdf.

[3] Xillybus Reference Manual, http://xillybus.com/downloads/doc/xillybus_getting_started_linux.pdf.

[4] Shumeet Baluja Henry A. Rowley and Takeo Kanade. Neural Network-Based Face Detection. *International Journal of Computer Vision*, 1998.

[5] Henry Schneiderman and Takeo Kanade. A Statistical Method for 3D Object Detection Applied to Faces and Cars. *International Journal of Computer Vision*, 2000.

[6] Paul Viola and Michael J. Jones. Robust Real-Time Face Detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[7] V Muthukumar and Daggu Venkateshwar Rao. Image Processing Algorithms on Reconfigurable Architecture using HandelC. *Proceedings of the EUROMICRO Systems on Digital System Design*, 2004.

[8] Nhat Tan Nguyen Thanh Ryan Kastner Scott B. Baden Daniel Hefenbrock, Jason Oberg. Accelerating Viola-Jones Face Detection to FPGA-Level using GPUs. *18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines*, 2010.

[9] C. Huang and F. Vahid. Scalable Object Detection Accelerators on FPGAs Using Custom Design Space Exploration. *Proceeding of the IEEE 9th Symposium on Application Specific Processors*, pages 115–121, 2011.

[10] P.O. Laprise V. Nair and J. J. Clark. An fpga-based people detection system. *Appl. Signal Process.*, 2005:1047–1061, 2005.

[11] C. Chareonsak Y. Wei and X. Bing. "Fpga implementation of adaboost algorithm for detection of face biometrics. *Biomedical Circuits and Systems*, 2004.

[12] Christian Beckoff Daniel Ziener Christopher Dennl Volker Breuer Jürgen Teich Michael Feilen Walter Stechele Dirk Koch, Jim Torresen. Partial Reconfiguration on FPGAs in Practice – Tools and Applications. *ARCS Workshops*, 2012.

[13] Ov7670 camera,
http://hamsterworks.co.nz/mediawiki/index.php/ov7670_camera, 2013.

[14] HLS Directives,
http://www.xilinx.com/support/answers/55279.html, 2013.

[15] Implementing Face detection algorithm,
http://etd.dtu.dk/thesis/223656/ep08_93.pdf.

[16] Mohammad Sadri Blog,
www.googoolia.com/wp/category/zynqtraining/.

[17] Xillybench github project,
https://github.com/troore/xillybench.

[18] OpenCV Documentation and Tools
http://opencv.org/.

[19] UG998 Xilinx documentation
http://www.xilinx.com/support/documentation/sw_manuals/ug998vivado-intro-fpga-design-hls.pdf

[20] VDMA reference design
http://www.xilinx.com/support/documentation/application_notes/xapp742-axi-vdma-reference-design.pdf

[21] Evaluating Vivado High-Level Synthesis on OpenCV Functions for the Zynq-7000 FPGA

http://www.idt.mdh.se/utbildning/exjobb/files/TR1803.pdf

[22] Vivado HLS Design Flow Lab

http://users.ece.utexas.edu/g̃erstl/ee382v_f14/soc/vivado_hls/VivadoHLS_labs.pdf

[23] Improving area and resources

users.ece.utexas.edu/g̃erstl/ee382v_f14/soc/vivado_hls/VivadoHLS_Improving_Resources.pdf