INTEGRATION OF OPENCV AND CYCLONE V HYBRID ARM AND FPGA SYSTEM ON A CHIP FOR FACE DETECTION APPLICATION

an

UNDERGRADUATE THESIS PROPOSAL

Presented to
The Faculty of Electrical
Electronics and Communications Engineering
MSU – Iligan Institute of Technology
Iligan City

In partial fulfillment
Of the requirements for the Degree
BACHELOR OF SCIENCE IN ELECTRONICS AND COMMUNICATIONS ENGINEERING

Submitted by:
GIL MICHAEL E. REGALADO

Adviser
PROF. JEFFERSON A. HORA

March 14, 2014

APPROVAL SHEET

The special project attached hereto, entitled "INTEGRATION OF OPENCV AND CYCLONE V HYBRID ARM AND FPGA SYSTEM ON A CHIP FOR FACE DETECTION APPLICATION" prepared and submitted by **GIL MICHAEL ECHAVEZ REGALADO** in partial fulfillment for the degree of BACHELOR OF SCIENCE IN ELECTRONICS AND COMMUNICATIONS ENGINEERING, is hereby recommended for approval:

ENGR. MERIAM V. BAUTISTA
Member

____
Date

ENGR. ALLEN LOWATON
Member

____
Date

ENGR. JEFFERSON A. HORA
Adviser

____
Date

This thesis is accepted and approved in partial fulfillment of the requirements for the Degree of Bachelor of Science in Electronics and Communications Engineering.

ENGR. OLGA JOY L. LABAJO
Chairman, EECE Department

____
Date

ATTY. ALLAN A. DONASCO
Dean, College of Engineering

____
Date

ABSTRACT

This research focused on the Integration Design of a Hybrid ARM and FPGA based Face Detection System powered by the OpenCV computer vision library and the SoCKit Altera Cyclone V System-on-Chip FPGA Development Board. A hardware and software integration system was designed for compatibility with the Cyclone V SoC SoCKit Development Board using Altera QSys and Altera Quartus. A custom version of the Linux Operating System Kernel from Github was then developed to support the Development Board's specification and the System Requirements such as USB Video Class Kernel Modules for USB Web Cam Support of the Integrated Face Detection System which was compiled using the Linaro tool-chain. OpenCV was then compiled within the Linux System and ~~the Face Detection System was then configured~~a face detection program using OpenCV face detection functions was developed that would be compatible to the integrated system~~to support the board~~. The Integrated Face Detection System was compared to a CISC based setup with Intel(R) Core(TM) i7-2670QM CPU @ 2.20GHz.~~Intel Core i7~~ The results showed that the SoC is slower by 43% compared to the Intel Core i7 setup in detecting a face from the standard Lena.jpg input file.

~~This research investigates that the Hybrid ARM and FPGA System on a Chip driven by Linux the open source operating system and the OpenCV Computer Vision Library is a potential platform for future Computer Vision Applications.~~

DEDICATED TO

FRIENDS

MENTORS

LOVED ONES

FAMILY

AND

THE OPEN SOURCE

HARDWARE AND SOFTWARE COMMUNITY

ACKNOWLEDGMENT

I would like to thank everyone who extended a hand in the completion of this undergraduate thesis. Without their kind help, understanding, encouragement and support, things would have been harder.

To my parents, Papa Gil T. Regalado and Mama Victoria E. Regalado for their extension in both encouragement and financial support during the duration of my research, I am very grateful for them. Their constant follow up and reminder has helped me finish this thesis.

To the many people who I have talked to and discussed with in real life and from the internet and whose help, views and insight have been very important to the development and completion of my study. To name a few, Sir Kit amd Sir Marven of the EE Department for some inspiration and Sir Howard Mao whom I've met from Github whose work in providing basic tutorial for the use of the Altera Cyclone V SoCKit Development Board and whose replies to my questions even how basic and annoying my questions are, he has been instrumental in inspiring me to proceed with this study.

Additionally, to the community at Rocket Boards Org and to those who are working with Linux Kernel and the Altera fork of the Linux Kernel whose discussions abound the rocket-boards request-for-information mailing list and Github, their continued work on fixing Kernel issues for board support on several FPGA hybrids and for responding to my questions have been a huge help to my development of this thesis. They are too many to mention but all their work and discussions in the mailing lists and forums have been of great help.

To my thesis adviser, Prof Jefferson Hora for encouraging me to proceed with this thesis and for giving me insights during the development of the project, his input and support has been instrumental in the completion of this thesis.

And above all, to the Almighty God who has given me strength and wisdom in pursuing this research study despite the several challenges, limitations and obstructions that abound during its course, thank you very much.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Background of the Study

One of the most important sensory ability of humans with the highest information density is vision.[2] The filtration methods of the human scene understanding capability is able to operate even in the high abundance of information by focusing on some elements while suppressing the rest. Artificial visual attention has been one of the key methodologies taken from nature that inspire researchers to develop robust and efficient machine vision systems for visual search applications.

As a scientific discipline Computer Vision collects the theory for building artificial systems that obtain information from images. Image data can either be a video frame, view from multiple cameras, or a multi-dimensional data from a medical scanner. Modern computer vision systems are applied in fields of process control, event detection, information organization, modeling of objects and man-machine interaction. The mentioned applications are often found applied in a wide array of industrial commercial, home and office applications. [1]

The study of computer vision describes the artificial vision system implemented in either software or hardware or the combination of both. One such software implementation is the open source computer vision library more commonly called as OpenCV. This library of programming function mainly aimed at real-time computer vision is free for use under the Berkeley Software (BSD) license. Released around 1999, OpenCV was a project from an Intel Research initiative to advance CPU-intensive applications. [1]

In this study the researcher will focus on investigating the potential of Altera's Cyclone V System on a Chip (SoC) with built in ARM Hard Processor component and FPGA Fabric for the application of Face Detection using the Open Source Computer Vision Library OpenCV. The SoCKit Development board will be used as the hardware platform for this study. The study will initially go through Development of the SoC Hardware and Software Integration, adaption of the Linux Operating System for running on the CycloneV SoC, and the compilation and to Integrate OpenCV for a Face Detection System utilizing the SoC.

1.2 Statement of the Problem

This paper had aimed to develop an Integrated Face Detection system using the OpenCV Library that operates on the CycloneV ARM and FPGA SoC Development Board called SoCKit.

1.3 Objectives of the Study

The general objectives of this study were:

1. developed the Hardware and Software Integration system required to run Linux on the CycloneV SoC SoCKit Development Board;
2. developed a custom version of the Linux Open Source Operating System that will be compatible with the system CycloneV SoC SoCKit Development Board;
3. compiled and installed the OpenCV library on to the Linux System;
4. integrated the Face Detection System using the Installed OpenCV Library with the rest of the subsystems;
5. and compared the performance parameters of the Face Detection system on an CISC Intel based platform.

1.4 Significance of the Study

This study aims to develop solutions for allowing the OpenCV Library to run on a Hybrid ARM and FPGA hardware. This will open opportunities for future research on accelerated performance of the OpenCV library for computer vision on ARM based devices using the FPGA fabric of the Cyclone V SoC. Considering the wide array of industries OpenCV is currently being implemented, and the prevalence of ARM on commercial and industrial applications, the future potential acceleration will provide a more efficient and scalable use of the OpenCV library in different fields of its application by different industries.

1.5 Scope and Limitations

In this study:

1. only the SoCKit Cyclone V Development Board was used for the prototype;
2. for the software and hardware integration, in order to reduce code complexity and build time to reasonable levels, only the necessary hardware components required for the research was integrated in to the system;
3. only a selected number of Linux Operating System subsystems was adopted to run on the SoCKit Cyclone V Development Board;
4. and to reduce cost, the primary image input method was through USB Video Class support which was compatible only with limited models of off-the-shelf USB Web Cameras. For this study, the Logitech C525 Camera was used.

1.6 Definition of Terms

ARM Architecture - is a family of instruction set architectures for computer processors based on a reduced instruction set computing (RISC) architecture developed by British company ARM Holdings.[7]

Complex Instruction Set Computer (CISC) - is a computer where single instructions can execute several low-level operations (such as a load from memory, an arithmetic operation, and a memory store) and/or are capable of multi-step operations or addressing modes within single instructions.[9]

Development Board - A microprocessor development board is a printed circuit board containing a microprocessor and the minimal support logic needed for an engineer to become acquainted with the microprocessor on the board and to learn to program it.[16]

Direct memory access (DMA) - Is a feature of modern computers that allows certain hardware subsystems within the computer to access system memory independently of the central processing unit (CPU). [4]

Ethernet - is a family of computer networking technologies for local area networks (LANs).[12]

Field-Programmable Gate Array (FPGA) - is an integrated circuit designed to be configured by a customer or a designer after manufacturing—hence "field-programmable". The FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC) (circuit diagrams were previously used to specify the configuration, as they were for ASICs, but this is increasingly rare). [5]

IP Core - In electronic design a semiconductor intellectual property core, IP core, or IP block is a reusable unit of logic, cell, or chip layout design that is the intellectual property of one party. IP cores may be licensed to another party or can be owned and used by a single party alone. The term is derived from the licensing of the patent and/or source code copyright that exist in the design. IP cores can be used as building blocks within ASIC chip designs or FPGA logic designs. [20]

Network on chip (NoC) - is a communication subsystem on an integrated circuit (commonly called a "chip"), typically between IP cores in a system on a chip (SoC). NoCs can span synchronous and asynchronous clock domains or use unclocked asynchronous logic. NoC technology applies networking theory and methods to on-chip communication and brings notable improvements over conventional bus and crossbar interconnections. NoC improves the scalability of SoCs, and the power efficiency of complex SoCs compared to other designs. [21]

PHY - An instantiation of PHY connects a link layer device (often called MAC as an abbreviation for Media Access Control) to a physical medium such as an optical fiber or copper cable. A PHY device typically includes a Physical Coding Sub layer (PCS) and a Physical Medium Dependent (PMD) layer.[18]

Pixels - a physical point in a raster image, or the smallest addressable element in an all points addressable display device; so it is the smallest controllable element of a picture represented on the screen. [15]

Processor - is the hardware within a computer that carries out the instructions of a computer program by performing the basic arithmetical, logical, and input/output operations of the system.[6]

Reduced Instruction Set Computing (RISC) - is a CPU design strategy based on the insight that simplified (as opposed to complex) instructions can provide higher performance if this simplicity enables much faster execution of each instruction.[8]

System on a Chip (SOC) - A system on a chip or system on chip (SoC or SOC) is an integrated circuit (IC) that integrates all components of a computer or other electronic system into a single chip. [3]

ULPI - is an interface standard for high-speed USB 2.0 IP systems. It defines an interface between USB IP link controllers (such as MUSBHDRC) and the PHYs or transceivers that drive the actual bus. ULPI stands for UTMI+ low pin interface and is designed specifically to reduce the pin count of discrete high-speed USB PHYs.[17]

Universal Serial Bus (USB) - is an industry standard developed in the mid-1990s that defines the cables, connectors and communications protocols used in a bus for connection, communication, and power supply between computers and electronic devices.[13]

Unshielded twisted pair (UTP) - cables are found in many Ethernet networks and telephone systems.[19]

USB On-The-Go (USB OTG) - is a specification that allows USB devices such as digital audio players or mobile phones to act as a host, allowing other USB devices like a USB flash drive, digital camera, mouse, or keyboard to be attached to them. Unlike conventional USB systems, USB OTG systems can drop the hosting role and act as normal USB devices when attached to another host. This can be used to allow a mobile phone to act as host for a flash drive and read its contents, downloading music for instance, but then act as a flash drive when plugged into a host computer and allow the host to read data from the device. [14]

Video Graphics Array (VGA) - refers specifically to the display hardware first introduced with the IBM PS/2 line of computers in 1987, but through its widespread adoption has also come to mean either an analog computer display standard, the 15-pin D-subminiature VGA connector or the 640x480 resolution itself. Today, the VGA analog interface is used for high definition video including 1080p and higher. While the VGA transmission bandwidth is high enough to support even higher resolution playback, there can be picture quality degradation depending on cable quality and length.[10]

Webcam - A webcam is a video camera that feeds its image in real time to a computer or computer network. Unlike an IP camera (which uses a direct connection using Ethernet or Wi-Fi), a webcam is generally connected by a USB cable, FireWire cable, or similar cable.[11]

1.7 Theoretical Framework

1.7.1 Hardware

This study requires the use of the SoCKit Evaluation Board and its on-board peripherals. In this section the components, their relevance to the study and associated technical data are introduced.

1.7.1.1 SoCKit Development Board

The board combines Cortex-A9 embedded cores with the FPGA fabric using a high-bandwidth interconnect backbone. Specifically, it contains Altera Cyclone V SoC with Dual ARM® Cortex®-A9 processors and 110K Les, High Speed Mezzanine Connector (HSMC) including transceivers, two banks of low-power DDR3 memory a MicroSD card and Ethernet 10/100/1000 interfaces, Adjustable clock output by Silicon Labs, Graphic LCD: 128 x 64 (SPI Interface), VGA and Audio connections, USB 2.0 OTG (Full Speed) and USB to UART connections with 3-Axis digital accelerometer and temperature sensor. [22]

Of particular interest in this study is the Cyclone V SoC inside the Development board, the board's USB OTG interface Support, VGA interface Support, Memory ICs and Ethernet interface support. Refer to Appendix B for the complete specification of the SoCKit Evaluation Board. The **Figure 1** below shows the system block diagram of the SoCKit Evaluation Board.

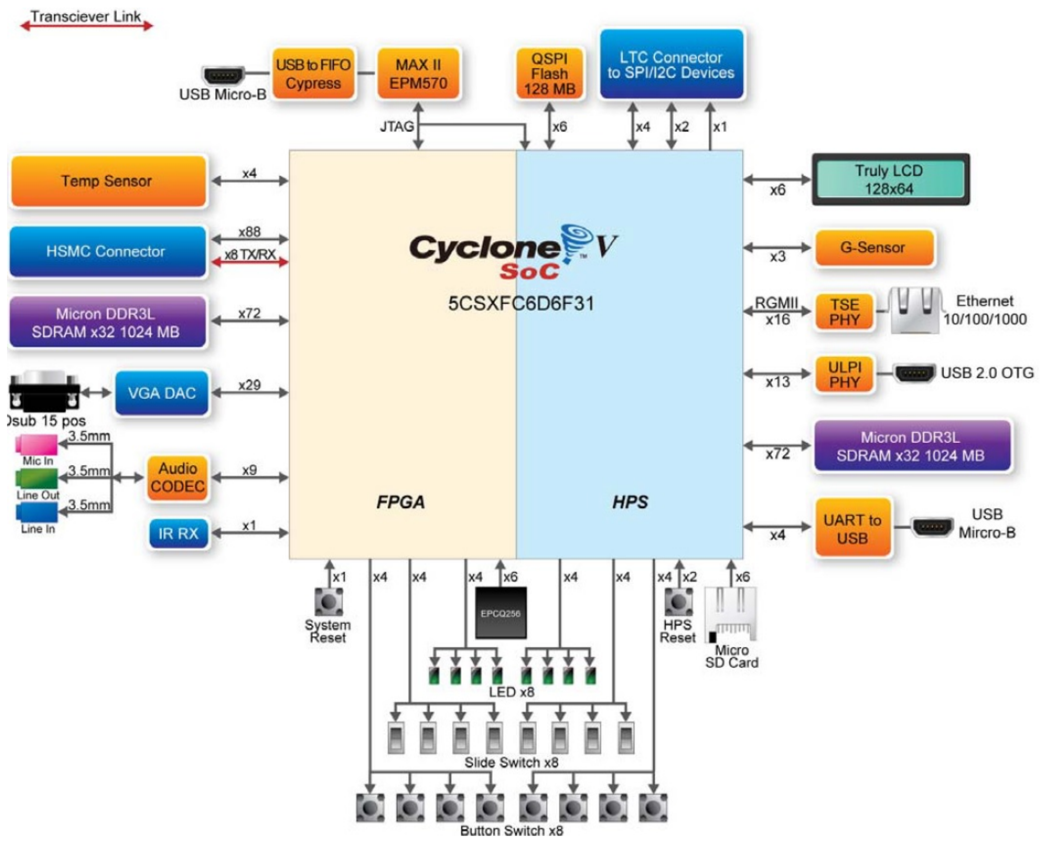


Figure 1. System Block Diagram of the SoCKit Evaluation Board

1.7.1.1.1 Cyclone V SoC

Quoted from the Altera Website, “the Altera SoCs integrate an ARM-based hard processor system (HPS) consisting of processor, peripherals, and memory interfaces with the FPGA fabric using a high-bandwidth interconnect backbone. The Cyclone® V SoCs reduce system power, system cost, and board size while increasing system performance by integrating discrete processor, FPGA, and digital signal processing (DSP) functions into a single, user customizable ARM-based system on a chip (SoC)”. [23]

For this Study a number of selected subsystems of the Cyclone V SoC will be used to develop the Face Detection System. Shown below is the block diagram of the Cyclone V SoC integrated circuit. Shown in **Figure 2** is the Cyclone V IC (Integrated Circuit) Block Diagram.

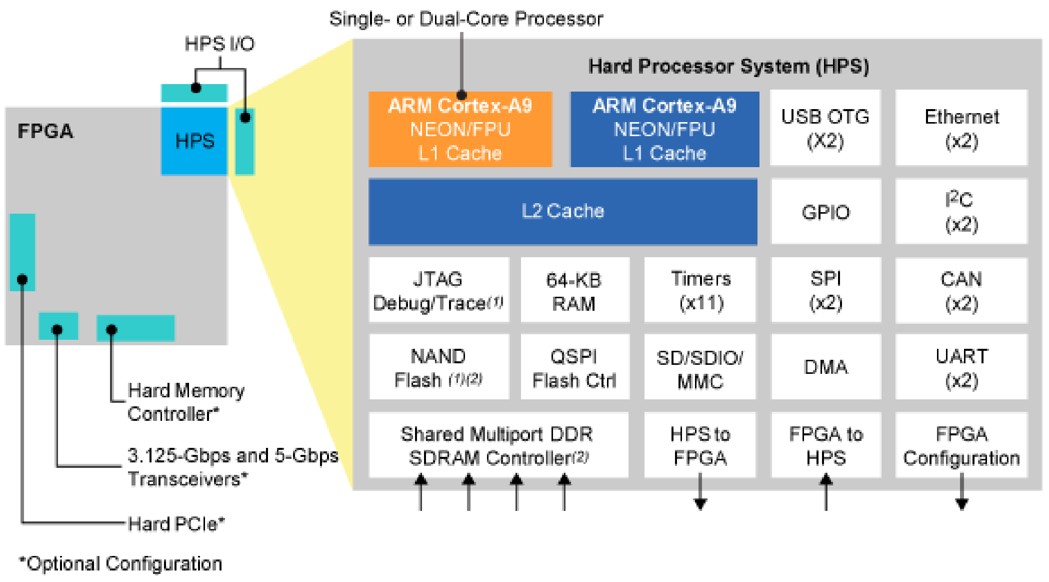


Figure 2. Cyclone V SoC Integrated Circuit Block Diagram

It features a 925 MHz, dual-core ARM® Cortex™-A9 MPCore™ processor with each processor having 32 KB of L1 instruction cache, 32 KB of L1 data cache, Single- and double-precision floating-point unit and NEON^{TM} media engine, CoreSight^{TM} debug and trace technology. And of particular interest, the IC also contains Multiport SDRAM controller with support for DDR2, DDR3, and LPDDR2 and optional error correction code (ECC) support, SD/SDIO/MMC controller with DMA, 2x 10/100/1000 Ethernet media access control (MAC) with DMA, and 2x USB On-The-Go (OTG) controller with DMA. [23]

1.7.1.1.1.1 Dual-Core ARM Cortex-A9 MPCore Processor

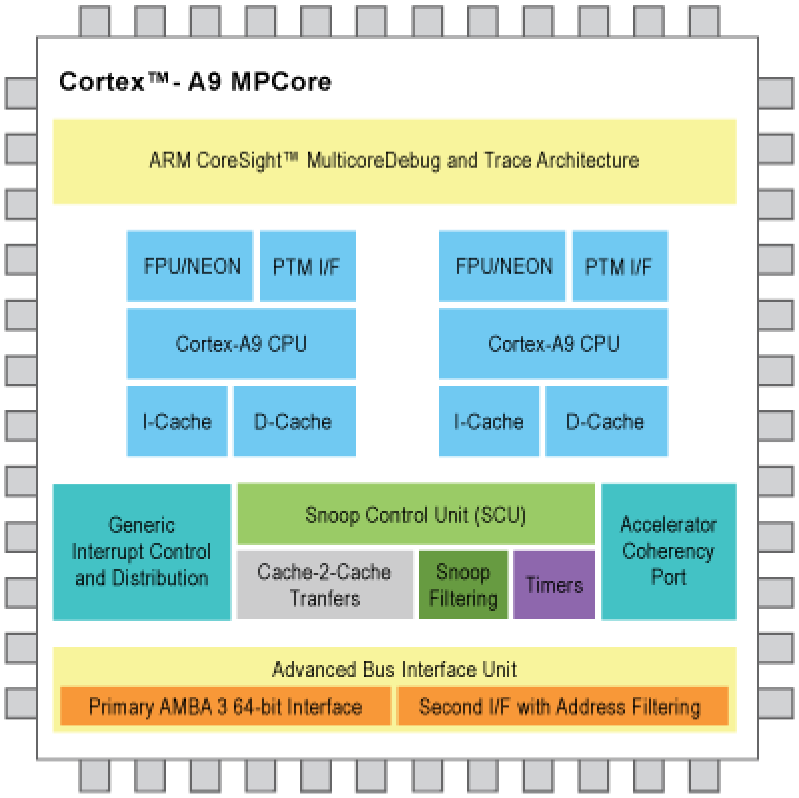

Cortex™- A9 MPCore
ARM CoreSight™ MulticoreDebug and Trace Architecture
FPU/NEON
PTM I/F
Cortex-A9 CPU
I-Cache
D-Cache
FPU/NEON
PTM I/F
Cortex-A9 CPU
I-Cache
D-Cache
Generic Interrupt Control and Distribution
Snoop Control Unit (SCU)
Cache-2-Cache Tranfers
Snoop Filtering
Timers
Accelerator Coherency Port
Advanced Bus Interface Unit
Primary AMBA 3 64-bit Interface
Second I/F with Address Filtering

Figure 3. Cortex A9 MPCore

The ARM Cortex-A9 processor with its Block Diagram shown in **Figure 3** is combined with a rich set of embedded peripherals, interfaces, and on-chip memories to create a complete hard processor system (HPS). The high-bandwidth on-chip backbone connecting the HPS and FPGA fabric provides over 100 Gbps peak bandwidth, ideal for sharing data between the ARM processor and hardware accelerators within the FPGA fabric. Full specification listing is available on

Appendix C as listed on the ARM Cortex Portion of the Altera Company Website.[24]

1.7.1.1.2 USB 3300 Hi-Speed USB Host, Device or OTG PHY

The built-in USB Controller shown in **Figure 4** of the board is the SMSC USB3300 Hi Speed USB Host, Device or OTG PHY with ULPI Low Pin Interface. It supports USB Specification Rev 2.0. In addition it supports OTG Monitoring of VBUS levels with internal comparators. This controller will become the input interface of the shelf USB Webcam to be used for the input image of the Face Detection system. [25]

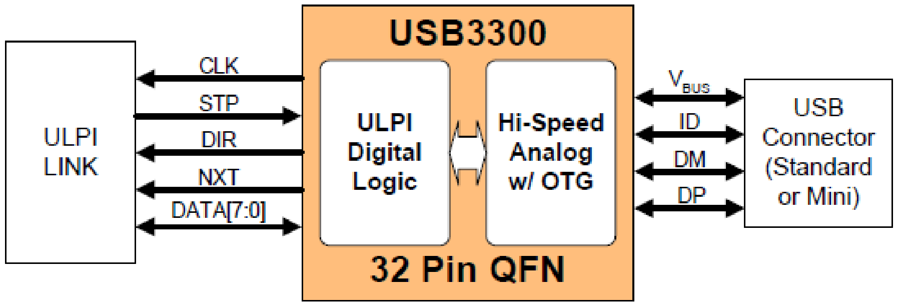


Figure 4. Basic ULPI USB Device Block Diagram

1.7.1.1.3 KSZ9021RL/RN Gigabit Ethernet Transceiver

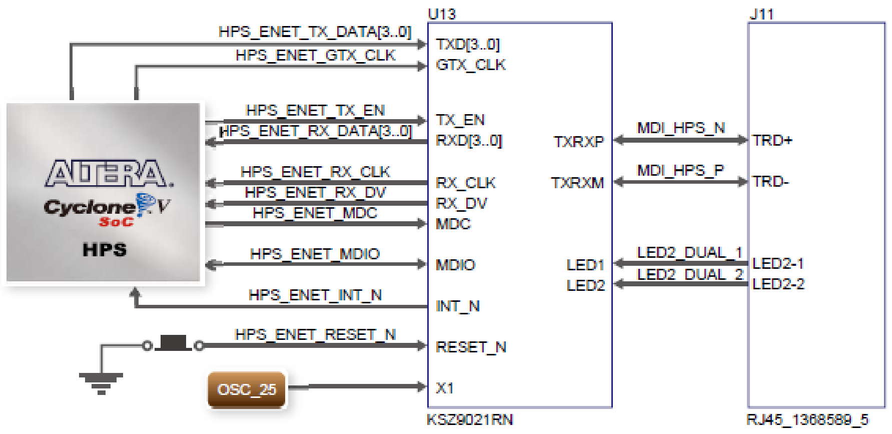


Figure 5. Connections between Cyclone V SOC and FPGA and Ethernet

The KSZ9021RL/RN is a completely integrated triple speed Ethernet Physical Layer Transceiver for transmission and reception of Data over standard CAT-5 unshielded twisted pair (UTP) cable. This subsystem is of particular interest to this research because it will be used as the Network Connection in download, compilation and installation of important Linux, OpenCV and other Software's Source Code and Associated Libraries. **Figure 6** shows the Functional Block Diagram of the Ethernet PHY and **Figure 5** shows the Connections between FPGA and Ethernet. [26]

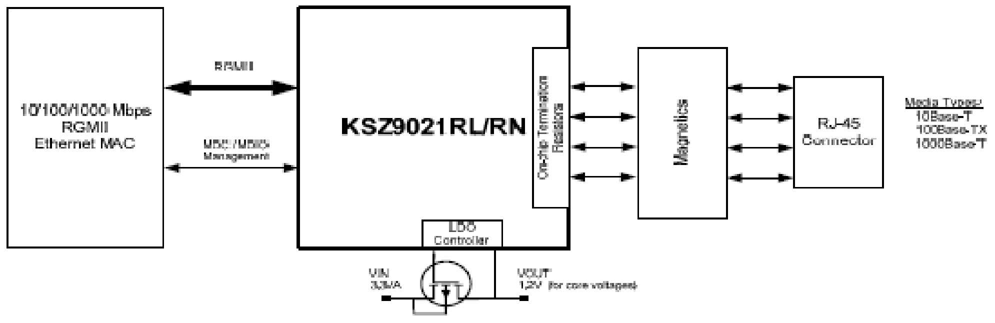


Figure 6. Functional Block Diagram of the KSZ9021RL/RN

1.7.1.1.4 VGA

The board includes a 15-pin D-SUB connector for VGA output. The VGA synchronization signals are provided directly from the Cyclone V SoC FPGA, and the Analog Devices ADV7123 triple 10-bit high-speed video DAC (only the higher 8-bits are used) is used to produce the analog data signals (red, green, and blue). It could support the SXGA standard (1280*1024) with a bandwidth of 100MHz. **Figure 7** shows the associated block Diagram. The VGA will be the interface used to connect the FPGA to the Display Monitor for display of output. [27]

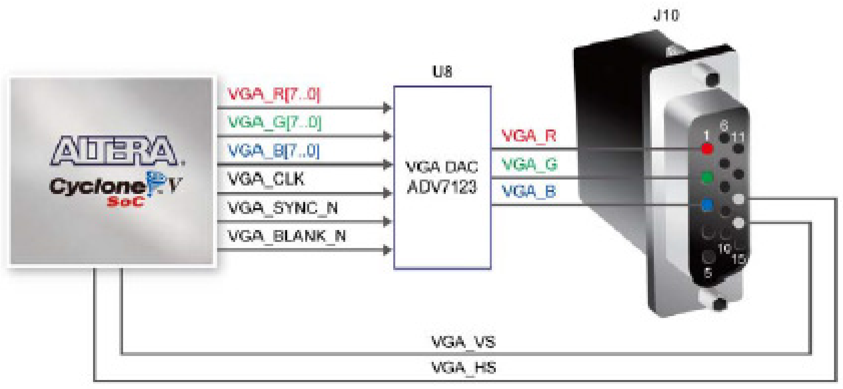


Figure 7. VGA Block Diagram

1.7.1.1.5 Memory

The board supports 1GB of DDR3 SDRAM comprising of two x16 bit DDR3 devices on FPGA side. The DDR3 devices shipped with this board are running at 400MHz if the hard external memory interface is enabled, and at 300MHz if the hard external memory interface if not enabled. **Figure 8** shows the connections between the DDR3 and Cyclone V SoC FPGA. The HPS memory must be considered during the system integration phase of the system. [27]

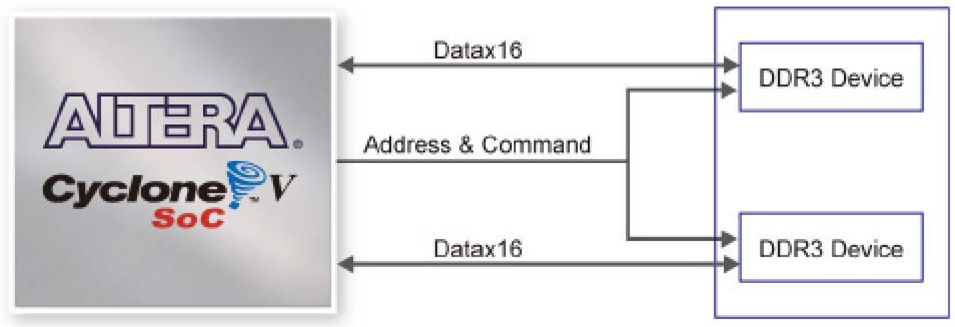


Figure 8. Connections between FPGA and DDR3

1.7.1.2 Logitech C525 Webcam

For the image input, an off the shelf USB Web Cam will be utilized. For this research the Logitech C525 Webcam is the model to be used. The Logitech C525 is capable of up to 1280 x 720 pixels for video resolutions and has 8 megapixels for still images. In addition, the researcher also intends to use this as a Mouse and Keyboard interface in conjunction with the Synergy Open Source Virtual KVM Software. [28]

1.7.1.3 Micro SD Card

The Micro Secure Digital (SD) is a non-volatile memory card format for use in portable devices, such as mobile phones, digital cameras, GPS navigation devices, and tablet computers. This Memory Card acts as the NAND Flash Memory for the Linux Operating System Files, and Device Tree structure. [29]

1.7.2 Software

In this paper two sets of categories for software will be made. First is the In-System software will be the software to be included inside the System in development. The second category will be the Development Software which will be software used in the development of the System.

1.7.2.1 In-System

Subsystems found inside the System being developed is referred to as In-System Software in this document. Some of the relevant In-System components, their relevance to the study and their technical details are discussed in this section.

1.7.2.1.1 Linux Kernel

The Linux Kernel will be the main Kernel used for the Linux Operating System to be deployed in the CycloneV SoC. The Linux kernel is a Unix-like operating system kernel used by a variety of operating systems based on it, which are usually in the form of Linux distributions. The Linux kernel is released under the GNU General Public License version 2 (GPLv2) (plus some firmware images with various non-free licenses), and is developed by contributors worldwide. [30]

1.7.2.1.2 Debian Operating System

Debian will be the operating System used in conjunction with the Linux Kernel. It is composed of free software mostly carrying the GNU General Public

License. The operating system is developed by an internet collaboration of volunteers aligned with The Debian Project. On top of Debian is where the OpenCV Library will be installed.

1.7.2.1.3 OpenCV Library

OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision, developed by Intel, and now supported by Willow Garage and Itseez. It is free for use under the open source BSD license. The library is cross-platform. It focuses mainly on real-time image processing. The OpenCV libraries functions and modules will be used for the development of the Face Detection System for the Altera Cyclone V Development Board. [1]

1.7.2.1.4 Lightweight X11 Desktop Environment (LXDE)

The project utilizes the LXDE for the Modified Debian Based Linux Operating System. LXDE is an energy saving and extremely fast performing desktop solution. It works well with computers on the low end of the performance spectrum such as new generation netbooks and other small mobile computers. LXDE is designed for cloud networks such as local freifunk clouds or the global Internet cloud. It can be built on top of various Linux distributions such as Ubuntu, Debian or Fedora. It supports a wealth of programs that can be installed with Linux systems locally. LXDE already supports many computer processor architectures including Intel, MIPS and ARM. In this case, we're using the LXDE ARM support for processor compatibility. [32]

1.7.2.2 Development Software

1.7.2.2.1 Altera Quartus

In order to design and compile the Hardware Description Code and IP Blocks of the System, the Altera Quartus II Complete Design Suite is used. Quartus II is a software tool produced by Altera for analysis and synthesis of HDL designs, which enables the developer to compile their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer. The version utilized for this project is 13sp1 which is a service pack of version 13. [33]

1.7.2.2.2 Qsys - Altera's System Integration Tool

The researcher also made use of the Qsys system integration for the FPGA design process by automatically generating interconnect logic to connect intellectual property (IP) functions and subsystems. Qsys is the next-generation SOPC Builder tool powered by a new FPGA-optimized network-on-a-chip (NoC) technology delivering higher performance, improved design reuse, and faster verification compared to SOPC Builder. [34]

1.7.2.2.3 SoC Embedded Design Suite

The SoC EDS is a comprehensive tool suite for embedded software development on Altera SoC devices. It contains development tools, utility programs, run-time software, and application examples to expedite firmware and application software of SoC embedded systems. The SoC EDS includes an exclusive offering of the ARM Development Studio™ 5 (DS-5™) Altera Edition Toolkit. The SoC EDS is used by the researcher to compile and use pre-built U-Boot and Linux build environments. [35]

1.7.2.2.4 Linaro Toolchain

The Linaro Toolchain is maintained by the Linaro Toolchain Working Group. The Linaro Toolchain deals with aspects of system-level tools - the core development toolchain compiler, assembler, linker, debugger, emulation, profiling and analysis using oprofile, and performance events and instrumentation with ftrace. Of particular interest in this research is the Linaro GCC Tool Chain. [36]

1.7.2.2.4.1 Linaro GCC

The GNU Compiler Collection (GCC) is a compiler system produced by the GNU Project supporting various programming languages. GCC is a key component of the GNU toolchain. The Free Software Foundation (FSF) distributes GCC under the GNU General Public License (GNU GPL). Linaro GCC is a fork of the GNU

GCC, and is a performance focused branch of the current GCC stable release and includes back ports of the improvements and bug fixes that Linaro and others have done upstream. The researcher used the Linaro GCC in compiling ported code for both the Linux Kernel, and the Linux Kernel Modules. [37]

1.7.2.2.5 Github

Github is not a software per se but a web-based hosting service for software development projects that use the Git revision control system. This research has a heavy use of this web service due to complexities in managing software versioning and the wide array of libraries, software, and configuration files necessary for this project. The real software behind it is Git. Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. [38]

1.7.2.2.6 Win32 Disk Imager

This is a Windows program for saving and restoring images from removable drives (USB drives, SD Memory cards, etc). It can be used to write boot images (i.e. ubuntu-12.04-preinstalled-desktop-armhf+omap4.img) to a SD Flash device or USB flash device, making it bootable. It currently does not support writing an ISO image to USB. The researcher has a heavy use of this software to backup and restore images that are compiled using the Linaro Tool Chain. Compilation and reconfiguration takes a huge amount of time in the development process and this tool is most helpful in making sure data are saved on the PC and restored to the MicroSD Cards. [39]

1.7.2.2.7 PuTTY

The researcher utilizes PuTTY for communication with the HPS Serial UART. PuTTY is a free and open-source terminal emulator, serial console and network file transfer application. It supports several network protocols, including SCP, SSH, Telnet, rlogin, and raw socket connection. The name "PuTTY" has no definitive meaning, though "tty" is the name for a terminal in the UNIX tradition, usually held to be short for Teletype. [40]

1.7.2.2.8 Others

There are many more software tools and utilities being used to develop this system. They are documented in APPENDIX I for reference.

CHAPTER II

REVIEW OF RELATED LITERATURE

A survey of related studies was undertaken by the researchers to get an insight from comparable studies to get suggestion regarding the ways and means for the collection of relevant data and interpretation of results.

2.1 OpenCV

OpenCV the open source computer vision library is released under a BSD license and hence it's free for both academic and commercial use. lt has a C++, C, Python and Java language support and supports Windows, Linux, Mac OS, iOS and Android operating systems. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can also take advantage of multi-core processing. [1]

2.2.1 OpenCV Face Detection

A recognition process can be much more efficient if it is based on the detection of features that encode some information about the class to be detected. This is the case of Haar-like features that encode the existence of oriented contrasts between regions in the image. A set of these features can be used to encode the contrasts exhibited by a human face and their spacial relationships. Haar-like features are so called because they are computed similar to the coefficients in Haar wavelet transforms. [1]

The object detector of OpenCV has been initially proposed by Paul Viola and improved by Rainer Lienhart. First, a classifier, namely a cascade of boosted classifiers working with haar-like features is trained with a few hundreds of sample views of a particular object, and negative examples which are arbitrary images of the same size. [41]

After a classifier is trained, it can be applied to a region of interest in an input image. The classifier outputs a "1" if the region is likely to show the object, and "0"othenrise. To search for the object in the whole image one can move the search window across the image and check every location using the classifier. The classifier is designed so that it can be easily "resized" in order to be able to find the objects of interest at different sizes, which is more efficient than resizing the image itself. So, to find an object of an unknown size in the image the scan procedure should be done several times at different scales. [41]

The process of cascading means that the resultant classifier consists of several simpler classifiers stages that are applied subsequently to a region of interest until at some stage the candidate is rejected or all the stages are passed. The word "boosted" means that the classifiers at every stage of the cascade are complex themselves and they are built out of basic classifiers using one of four different boosting techniques called weighted voting. [1]

Currently Discrete AdaBoost, Real AdaBoost, Gentle AdaBoost and Logitboost are supported. The basic classifiers are decision-tree classifiers with at least two leaves. Haar-like features are the input to the basic classifiers. The feature used in a particular classifier is specified by its shape, position within the region of interest and the scale. [1]

2.2.1.1 Rapid Object Detection using a Boosted Cascade of Simple Features

A research papers presented on the Conference on Computer Vision and Pattern Recognition (2001) describes an approach for visual object detection capable of processing images rapidly along with high detection rates using Machine Learning. An integral image is introduced in their study which allows for the detector to compute features very quickly. A learning algorithm based on AdaBoost was also developed in their study which selects a small portion of critical visual features from a larger set which results as they called an extremely efficient set of classifiers. Then finally, they cascade this classifiers to get more detail and more complexity allowing background regions to be quickly discarded and the region of interests identified. [48]

2.2.1.2 AdaBoost

In machine learning, boosting is the approach of creating a highly accurate rules by combining the results of many weak and inaccurate rules. One of the first practical boosting algorithm was of Freund and Schapire which is mostly used and studied today. The algorithm assumes that each weak hypothesis has accuracy that is better than random guessing. This assumption is sometimes called a weak learning condition. Given the weak learning condition it is then possible to prove that the training error according to Schapire can be reduced to zero very rapidly.[49]

2.2.1.3 FPGA-Based Face Detection System Using Haar Classifiers

A comparable study presents a hardware architecture for face detection designed using the AdaBoost algorithm using Haar features. The proponents of this study designed an image scaling, integral image generation, pipelined processing as well as a classifier and parallel processing multiple classifiers to accelerate the processing speed of their face detection system. They designed their system using Verilog HDL and implemented it in Xilinx Virtex 5 FPGA. They were able to show 35 times increase in performance compared to equivalent software implementation.[50]

2.2.1.4 Object Detection Using the Statistics of Parts

A research study by Schnedierman and Kanade proposes an object detector and its instantiations for detecting faces and cards at any size, location and pose. Their implementation uses multiple classifiers from different orientations. Each of such classifiers determines the presence of an object on a specified image window. The classifiers scan the image exhaustively such that the location and size of the target can be determined. [51]

2.2.1.5 A Parallel Architecture for Hardware Face Detection

Theocharides and his group also studied a scalable parallel architecture for face detection using AdaBoost algorithm. Their experimental results show that their proposed architecture can detect faces at the same accuracy as the software implementation on a real-time video at 52 frames per second. [52]

2.2.2 OpenCV Face Recognition

Presently, OpenCV supports three different algorithms for Face Recognition namely, Eigenfaces; Fisherfaces; and Local Binary Patterns Histograms. Face recognition is an easy task for humans. lt was shown by David Hubel and Torsten Wiesel, that our brain has specialized nerve cells responding to specific local features of a scene, such as lines, edges, angles or movement. Since humans don't see the world as scattered pieces, our visual cortex must somehow combine the different sources of information into useful patterns. Automatic face recognition is all about extracting those meaningful features from an image, putting them into a useful representation and performing some kind of classification on them. [42]

In computerized face recognition, each face is represented by a large number of pixel values. Linear discriminant analysis is primarily used here to reduce the number of features to a more manageable number before classification. Each of the new dimensions is a linear combination of pixel values, which form a template. The linear combinations obtained using Fisher's linear discriminant are called Fisher faces, while those obtained using the related principal component analysis are called Eigen faces. [42]

2.2 Current State of OpenCV Acceleration.

There have been many efforts in accelerating the current OpenCV library. However, none of them are focused on the ARM architecture which is the de facto standard in mobile and embedded applications. It is of special interest for this research that the status for Hardware Acceleration of OpenCV to be studied due to the fact that this research will lead to improved chances of increasing the possibility of hardware acceleration of OpenCV on ARM.

2.2.1 OpenCV GPU

The OpenCV GPU module is a set of classes and functions to utilize GPU computational capabilities. lt is implemented using NVIDIA CUDA Runtime API and supports only NVIDIA GPUs. The OpenCV GPU module includes utility functions, low level vision primitives, and high-level algorithms. The utility functions and low-level primitives provide a powerful infrastructure for developing fast vision algorithms taking advantage of GPU whereas the high-level functionality includes some state-of-the-art algorithms (such as stereo correspondence, face and people detectors, and others) ready to be used by the application developers. [43]

2.2.2 OpenCV IPP

lntel lntegrated Performance Primitives (lntel IPP) is an extensive library of multicore-ready, highly optimized software functions for multimedia, data processing, and communications applications. lntel IPP offers thousands of optimized functions covering frequently used fundamental algorithms. There is a free non-commercial version of IPP for Linux as made available by lntel but the implementation is proprietary. [44] [1]

2.2.3 OpenCV Applications with Zynq-7000 All Programmable SoC

The design flow leverages HLS technology in the Vivado Design Suite, along with optimized synthesizable video libraries. The libraries can be used directly, or combined with application-specific code to build a customized accelerator for a particular application. This flow can enable many computer vision algorithms to be quickly implemented with both high performance and low power. The flow also enables a designer to target high data rate pixel processing tasks to the programmable logic, while lower data rate frame-based processing tasks remain on the ARM cores. [45]

Alternatively, the OpenCV function calls can be replaced by corresponding synthesizable functions from the Xilinx Vivado HLS video library. OpenCV function calls can then be used to access input and output images and to provide a golden reference implementation of a video processing algorithm. After synthesis, the processing block can be integrated into the Zynq Programmable Logic. Depending on the design implemented in the Programmable Logic, an integrated block may be able to process a video stream created by a processor, such as data read from a file, or a live real-time video stream from an external input. [45]

2.3 USB Video Class

This research also takes great care in making sure the available input imaging devices are supported by the Linux USB Video Class. The USB Device

Class Definition for Video Devices, or USB Video Class, defines video streaming functionality on the Universal Serial Bus. Much like nearly all mass storage devices can be managed by a single driver because they conform to the USB Mass Storage specification, UVC compliant peripherals only need a generic driver. [46, 47]

The UVC specification covers webcams, digital camcorders, analog video converters, analog and digital television tuners, and still-image cameras that support video streaming for both video input and output. However, as stated on their website, due to the limited available man power and the broad scope of the UVC specification, the Linux UVC project will concentrate the development efforts on video input devices, especially webcams. In addition, video output devices are supported in bulk mode only and are therefore less favored. [46, 47]

It was noted that the Logitech C525 Webcam utilized by the researcher as an input image device is fully supported by the Linux UVC. It is therefore necessary that the appropriate Kernel Modules for the Linux Kernel be included in the system.

CHAPTER III

METHODOLOGY

3.1 System Requirements

The research required the development of a System that for any compatible Input Image, an Output Image is produced for which the Face Detection filter would be applied. An indication of the Face that appears on the Image could be by way of the putting rectangle overlay over the Face that appears in the image. A very simple diagram of this is shown in **Figure 9**. In this research, in order to have fair treatment of multiple tests, the Face Detection sample program of the OpenCV Library was used as a standard algorithm. In addition, for statistical test, the standard test Image is a picture of Lena Söderberg for which many other Face Detection System Studies are using as a standard test image.

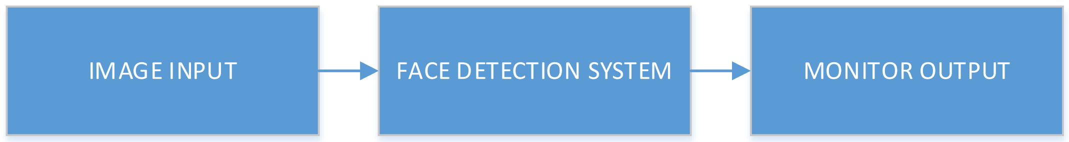


Figure 1. Top Level block diagram of the Entire System. The face detection system component is expanded in Appendix A.

3.2 Top Level Face Detection System Design Overview

The system design involved hardware and software abstraction and a clear separation between systems that interact with each other. At the bottom is the hardware layer in which the board and its peripherals such as the SoC, and IC Controllers are located. There is then an integration layer that is the Raw Binary File (RBF) which configures the FPGA fabric of the Hardware Layer. The OS Layer which lives in the Memory Card image is then configured to work in conjunction with the RBF which is also called by uBoot during System Boot. Refer to the diagram below (**Figure 10**) for the overall system design.

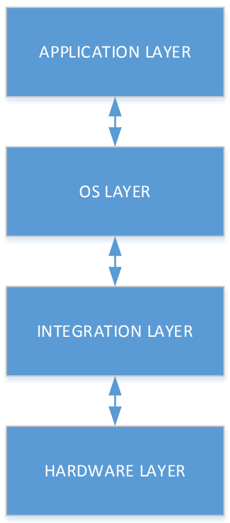


Figure 2. Top Level System Overview (See APPENDIX A for the Expanded Diagram)

3.3 Hardware Preparation

In order to begin the research the board which has different configuration has to be properly prepared by way of setting the proper BOOTSEL and CLKSEL settings header found on its TOP and BOTTOM side of the PCB. These headers are intended for configurability of the board boot process, boot source and clock distribution. Please Refer to **Appendix D** for the proper Board Configuration.

3.4 Development Environment Preparation

The researcher used the Altera Web-edition 13.1 Release of the Altera Design Suite. It is a free version of the Alter Design suite with Quartus and QSsys System. In addition the researcher developed a Linux Build Environment by way of a Virtual Desktop environment implemented on Virtual Box Open Source Virtualization Software. The build environment is based on the Open Source Ubuntu Linux. There are many more tools and utilities used in this research but some of them have trivial purposes and even some are an integral part of either the Windows OS or the Ubuntu based Linux OS. Shown in **Figure 11** are the development tools required for both Windows and Linux environments.

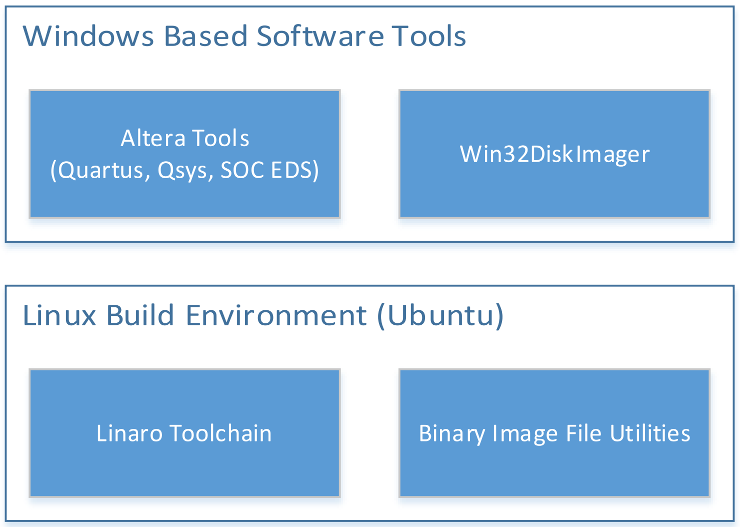

Windows Based Software Tools
Altera Tools (Quartus, Qsys, SOC EDS)
Win32DiskImager
Linux Build Environment (Ubuntu)
Linaro Toolchain
Binary Image File Utilities

Figure 3. Development Tools

3.5 Integration Layer

3.5.1 Golden Hardware Reference Design

Using the QSYS tools, the IP Blocks necessary to emulate the Golden Hardware Reference Design was compiled in to Verilog. After generation, the produced Verilog file with references to the required Altera and 3rd Party IP Blocks was Analysed and Synthesized using Quartus. In addition an included PIN Placement TCL script was utilized which was produced by QSYS. After Analysis and Synthesis, a Full Quartus Compile was done. Using Altera's BSP Editor which was included in SoC EDS, the preloader source files were produced. After that they are compiled using Altera's Built in *make* and *GCC*.

3.6 OS Layer

3.6.1 Modified Linux Kernel Source

Altera provides an example Linux Kernel through Rocket Boards Org, however the Kernel available is very limited and required extensive modification. This opted the researcher to clone the Linux Kernel Source they have made available at Github instead. The main advantages with Altera's Kernel source are the following:

1. Improved Frame buffer using the source copied from NIOS2 tree with the addition of some minor modifications in order to make it compatible with Cyclone V SoC.
2. The Kernel Build configuration was already modified to make sure some of the key systems are by default enable. This caused only a few minor adjustments to be made with the menuconfig utility after the socfpga_defconfig was ran.
3. The kernel device tree is now based on the one produced from the Altera Golden Hardware Reference Design (GHRD).

Using menuconfig, several modifications to the Linux Kernel configuration was made. Among the vital modifications is making sure the USB and UVC Kernel

Modules are properly configured. Using the Linaro GCC the Linux Kernel Source and the Kernel Modules was compiled.

3.6.2 U-Boot Configuration

Using the PuTTY console, The Board's U-Boot was also configured. The Linux kernel requires that the FPGA be configured with the video pipeline prior to the video frame buffer driver being implemented. The FPGA configuration is implemented by the u-boot fpgaload command. And the notes on Appendix E was added to U-BOOT. This makes sure that U-Boot understands and detects the location of the appropriate Kernel Drivers and Kernel Compiled code when booting.

3.6.3 SD Card Image

3.6.3.1 The Debian Based Linaro Ubuntu Image

The Debian based Image of Linaro Ubuntu was downloaded from the Linaro Org. This was extracted and stored on a specified location on the Linux Build environment.

3.6.3.2 Partitioning of the SD Card Image

Using the Ubuntu Linux Based Software Development OS, the MicroSD Card was identified and partitioned. An excerpt from RocketBoards is on Appendix F on how to partition an SD Card using Linux.

3.6.3.3 System Files

The preloader, was placed in the binary partition. The Linux kernel, Device Tree and FPGA RBF respectively the soc_system.dtb, soc_system.rbf, uImage are placed on the FAT 32 Partition. And finally, the entire Linaro Root Filesystem is copied to the Linux partition. In summary the Boot Process operates as described in the image in **Figure 12**.

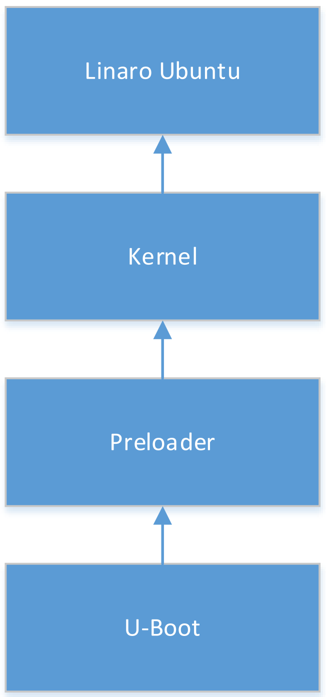


Figure 4. Boot Process

3.6.4 Utilities

The Linux System now installed the Board is capable of booting in to Linux. After booting, the important utilities are installed and compiled. Key utilities are the SSH for network connection, compiler utilities, and the X11, Synergy and XServer family of utilities for the user interface.

3.6.5 OpenCV Library

Finally, in this stage, the OpenCV library source and its associated dependencies are downloaded through Ethernet directly to the SD Card. After installing all the development tools necessary within the SoCKit Linux, OpenCV and its associated libraries are compiled. Key libraries are FFMPEG, V4L2 and QT. After compilation the Face Detect example program is modified to fit the environment. After modification the Face Detect program is ran and tested.

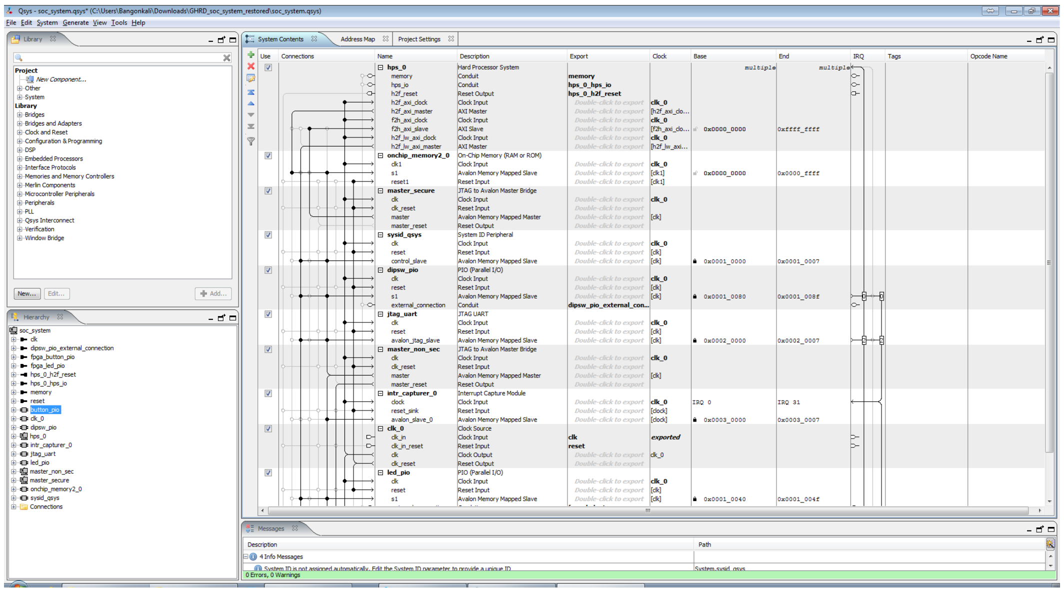
CHAPTER IV

RESULTS AND DISCUSSIONS

This chapter evaluates and describes the results and tests conducted by the researchers after the completion of the Face Detection System.

4.1 System Integration

A Hardware and Software Integration system required to run the Linux Kernel and its associated Linaro Ubuntu Operating System was successfully designed using Altera Quartus and Altera Qsys. In the **Figure 13** the Qsys environment is shown were the Hardware Integration is done.



Qsys - soc_system.qsys* (C:\Users\Bangonkali\Downloads\GHRD_soc_system_restored\soc_system.qsys)

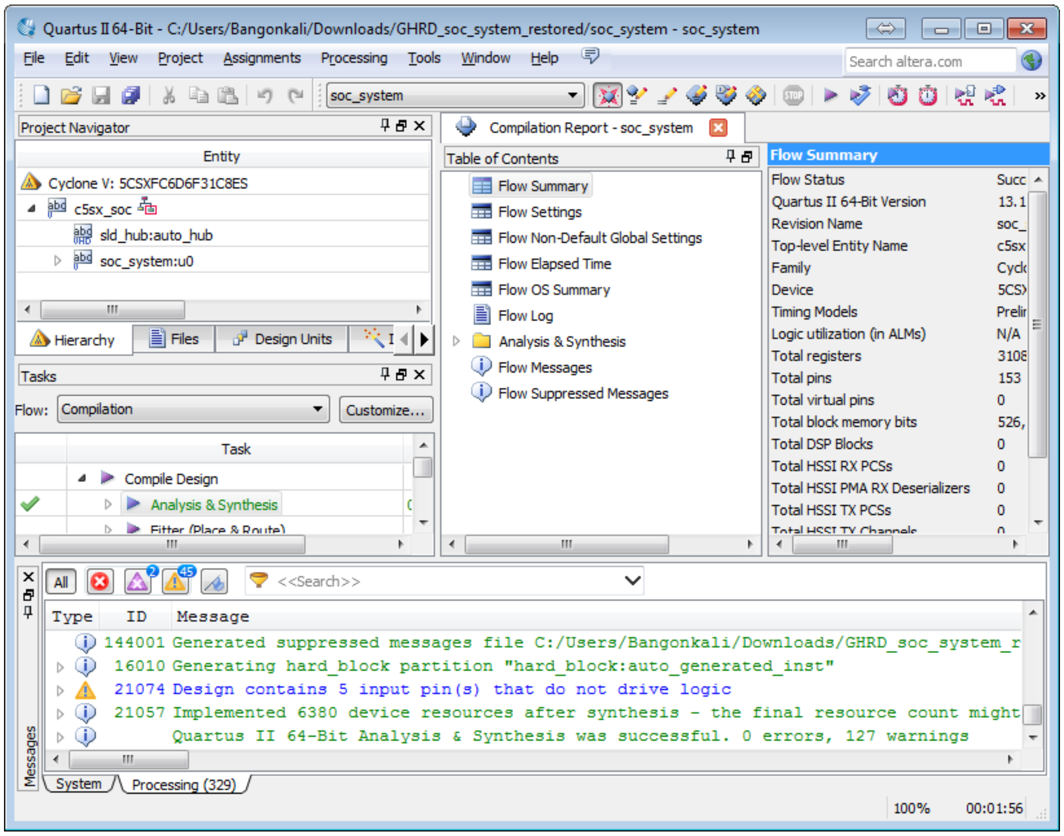
Use	Name	Description	Export	Clock	Base	End	IRQ
✓	hps_0	Hard Processor System			multiple	multiple	
	memory	Conduit	memory				
	hps_io	Conduit	hps_0_hps_io				
	h2f_reset	Reset Output	hps_0_h2f_reset				
	h2f_axi_clock	Clock Input	Double-click to export	clk_0			
	h2f_axi_master	AXI Master	Double-click to export	[h2f_axi_clo...			
	f2h_axi_clock	Clock Input	Double-click to export	clk_0			
	f2h_axi_slave	AXI Slave	Double-click to export	[f2h_axi_clo...	0x0000_0000	0xffff_ffff	
	h2f_lw_axi_clock	Clock Input	Double-click to export	clk_0			
	h2f_lw_axi_master	AXI Master	Double-click to export	[h2f_lw_axi...			
✓	onchip_memory2_0	On-Chip Memory (RAM or ROM)					
	clk1	Clock Input	Double-click to export	clk_0			
	s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]	0x0000_0000	0x0000_ffff	
	reset1	Reset Input	Double-click to export	[clk1]			
✓	master_secure	JTAG to Avalon Master Bridge					
	clk	Clock Input	Double-click to export	clk_0			
	clk_reset	Reset Input	Double-click to export				
	master	Avalon Memory Mapped Master	Double-click to export	[clk]			
	master_reset	Reset Output	Double-click to export				
✓	sysid_qsys	System ID Peripheral					
	clk	Clock Input	Double-click to export	clk_0			
	reset	Reset Input	Double-click to export	[clk]			
	control_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0001_0000	0x0001_0007	
✓	dipsw_pio	PIO (Parallel I/O)					
	clk	Clock Input	Double-click to export	clk_0			
	reset	Reset Input	Double-click to export	[clk]			
	s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0001_0080	0x0001_008f	0
	external_connection	Conduit	dipsw_pio_external_con...				
✓	jtag_uart	JTAG UART					
	clk	Clock Input	Double-click to export	clk_0			
	reset	Reset Input	Double-click to export	[clk]			
	avalon_jtag_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0002_0000	0x0002_0007	2
✓	master_non_sec	JTAG to Avalon Master Bridge					
	clk	Clock Input	Double-click to export	clk_0			
	clk_reset	Reset Input	Double-click to export				
	master	Avalon Memory Mapped Master	Double-click to export	[clk]			
	master_reset	Reset Output	Double-click to export				
✓	intr_capturer_0	Interrupt Capture Module					
	clock	Clock Input	Double-click to export	clk_0	IRQ 0	IRQ 31	
	reset_sink	Reset Input	Double-click to export	[clock]			
	avalon_slave_0	Avalon Memory Mapped Slave	Double-click to export	[clock]	0x0003_0000	0x0003_0007	
✓	clk_0	Clock Source					
	clk_in	Clock Input	clk	exported			
	clk_in_reset	Reset Input	reset				
	clk	Clock Output	Double-click to export	clk_0			
	clk_reset	Reset Output	Double-click to export				
✓	led_pio	PIO (Parallel I/O)					
	clk	Clock Input	Double-click to export	clk_0			
	reset	Reset Input	Double-click to export	[clk]			
	s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0001_0040	0x0001_004f	

Description	Path
4 Info Messages	
System ID is not assigned automatically. Edit the System ID parameter to provide a unique ID	System.sysid_qsys

0 Errors, 0 Warnings

Figure 13. Hardware Design using Altera QSys

The output for QSys was successfully compiled by Altera Quartus.



Flow Summary

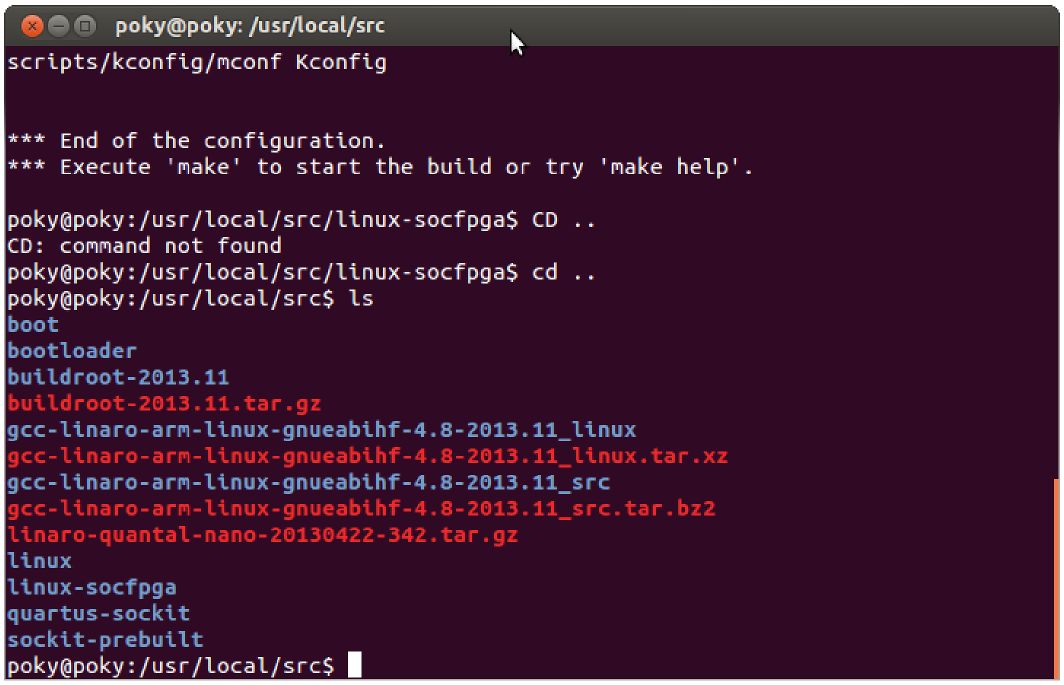
Flow Status	Succ
Quartus II 64-Bit Version	13.1
Revision Name	soc_
Top-level Entity Name	c5sx
Family	Cycl
Device	5CSX
Timing Models	Prelir
Logic utilization (in ALMs)	N/A
Total registers	3108
Total pins	153
Total virtual pins	0
Total block memory bits	526,
Total DSP Blocks	0
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI TX Channels	0

```
Type   ID    Message
  144001 Generated suppressed messages file C:/Users/Bangonkali/Downloads/GHRD_soc_system_r
   16010 Generating hard_block partition "hard_block:auto_generated_inst"
   21074 Design contains 5 input pin(s) that do not drive logic
   21057 Implemented 6380 device resources after synthesis - the final resource count might
         Quartus II 64-Bit Analysis & Synthesis was successful. 0 errors, 127 warnings
```

Figure 14. Successful compilation of the Boot loader and Hardware Integration System

4.2 Linux System

The Linux System was also successfully compiled with the intended modifications using the menuconfig tool of the Linux build environment. Shown in **Figure 15** is the Linux Build Environment toolset.



poky@poky: /usr/local/src

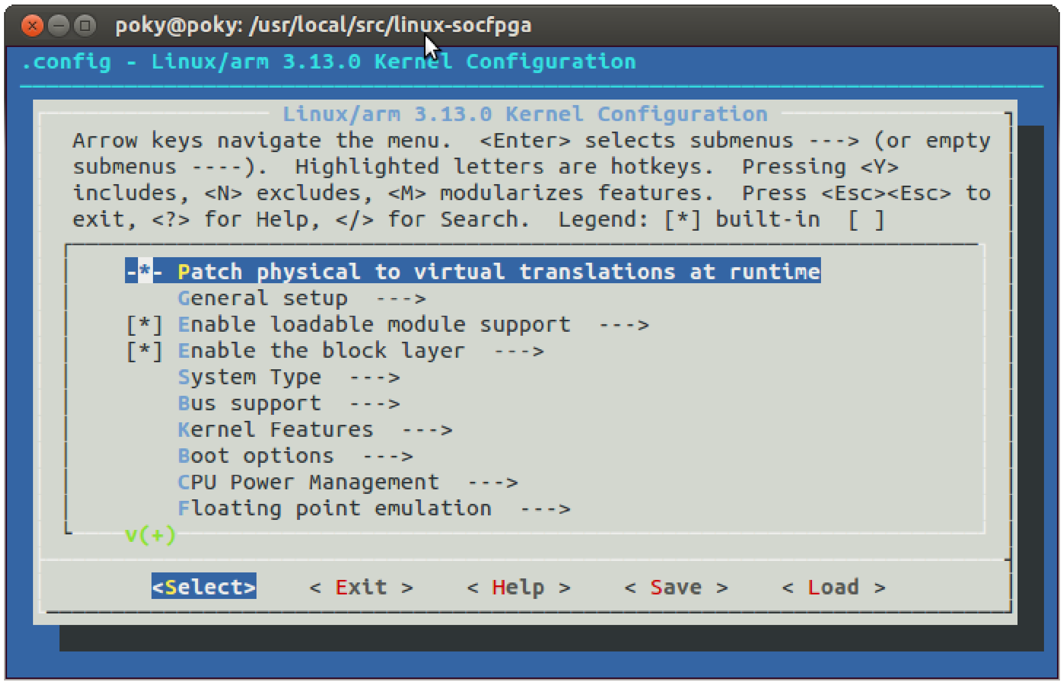
```
scripts/kconfig/mconf Kconfig


*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.

poky@poky:/usr/local/src/linux-socfpga$ CD ..
CD: command not found
poky@poky:/usr/local/src/linux-socfpga$ cd ..
poky@poky:/usr/local/src$ ls
boot
bootloader
buildroot-2013.11
buildroot-2013.11.tar.gz
gcc-linaro-arm-linux-gnueabihf-4.8-2013.11_linux
gcc-linaro-arm-linux-gnueabihf-4.8-2013.11_linux.tar.xz
gcc-linaro-arm-linux-gnueabihf-4.8-2013.11_src
gcc-linaro-arm-linux-gnueabihf-4.8-2013.11_src.tar.bz2
linaro-quantal-nano-20130422-342.tar.gz
linux
linux-socfpga
quartus-sockit
sockit-prebuilt
poky@poky:/usr/local/src$ 
```

Figure 15. Linux Build environment toolset.

Figure 16 shows the menuconfig tool where the UVC Kernel Module was successfully configured to allow the Linux Kernel to detect Webcam especially the Logitech C525 used in this study.

4.3 OpenCV Library & Face Detection

Within the Linux System on SoCKit, the OpenCV System was compiled along with the face detection program. A script was then programmed such that data from the webcam will be sent to the face detection program. In **Figure 17**, the compilation of the face detection program is shown as successful.



```
poky@poky: /usr/local/src/linux-socfpga
.config - Linux/arm 3.13.0 Kernel Configuration
                    Linux/arm 3.13.0 Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]

      -*- Patch physical to virtual translations at runtime
          General setup  --->
      [*] Enable loadable module support  --->
      [*] Enable the block layer  --->
          System Type  --->
          Bus support  --->
          Kernel Features  --->
          Boot options  --->
          CPU Power Management  --->
          Floating point emulation  --->
      v(+)

        <Select>    < Exit >    < Help >    < Save >    < Load >
```

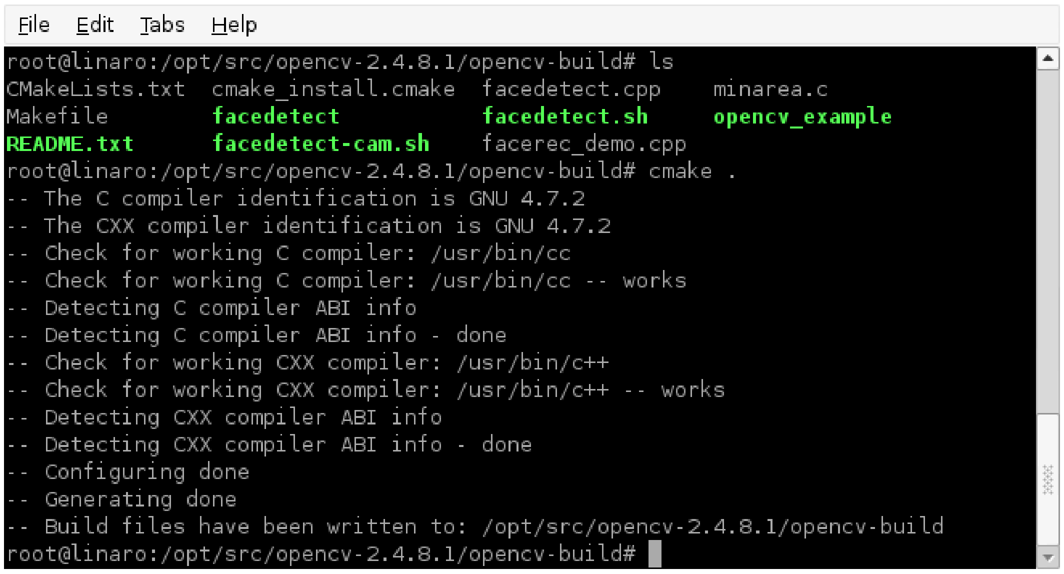
Figure 16. Menuconfig tool

4.4 Performance Comparison

Using the modified face detect example program and the standard Lena.jpg (See **Appendix K**) Image, the detection time is recorded on **Table 1** and compared to a CISC based system (See **Appendix G** for Specs). For test results of Cyclone V, see **Appendix J**. For the Intel Core i7 Test Results, See **Appendix L**.

Table 1 Comparrison of CISC based and Cyclon V SoC.

Intel Core i7 CISC Based	Cyclone V	% Difference
1687.06 ms	954.156 ms	43.44268%



```
root@linaro:/opt/src/opencv-2.4.8.1/opencv-build# ls
CMakeLists.txt  cmake_install.cmake  facedetect.cpp    minarea.c
Makefile        facedetect           facedetect.sh     opencv_example
README.txt      facedetect-cam.sh    facerec_demo.cpp
root@linaro:/opt/src/opencv-2.4.8.1/opencv-build# cmake .
-- The C compiler identification is GNU 4.7.2
-- The CXX compiler identification is GNU 4.7.2
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Configuring done
-- Generating done
-- Build files have been written to: /opt/src/opencv-2.4.8.1/opencv-build
root@linaro:/opt/src/opencv-2.4.8.1/opencv-build# 
```

Figure 17. Successfully compiled the face detection program

4.5 System Bugs

During the development phase despite the researchers careful deliberations in choosing the system components, sources code libraries, and algorithms used it is unavoidable that there may have been system bugs, in fact a few of them show and influence greatly the operation and performance of the face detection system being developed.

4.6.1 USB UVC Bandwidth Issues with USB OTG

Though the exact cause of the system can’t be determined there seems to be an issue with bandwidth issues with the USB OTG. This issue becomes manifests itself with the USB based WebCam Logitech C525 not working when the Keyboard and Mouse or other USB peripheral are inserted to the USB Port of an off-the-shelf USB HUB. Only when the USB Webcam is acting alone and on the USB Controller will UVC V4L2 detect the Web Cam.

The temporary solution for this problem was to incorporate the Opensource KVM Emulation Software called Synergy. Essentially the Mouse and Keyboard support is done from another computer. Mouse and Keyboard are both installed on a different computer. Synergy Client listening on the SoCKit board is listening to commands of the Synergy Server running on the computer where the Mouse and Keyboard are. In turn, the Synergy Server is listening to the Mouse and Keyboard Movements of the user and sending it to the client.

Both Synergy Client and Server communicate VIA Ethernet and that is why it is essential among many other reasons that the Development Board must be connected to the network where the Synergy Server is in order to operate it properly.

4.6.2 USB UVC has some issues FFMPEG

The researcher also observed that there are some errors in long term operation of the face detection system. This could be in line with the previous case stated on 4.2.1. The Kernel Panic logs are documented on **Appendix H**.

CHAPTER V

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

The researcher was able to develop a Hardware and Software Integration System required to run Linux on the Cyclone V SoC SoCKit Development board and the associated Linux Based Operating System on which the OpenCV Library runs the Face Detection Algorithm for the Face Detection System. In addition, the Researcher also developed a separate test system using an Intel CISC based Core i7 platform and compare the performance with the SoCKit Deployment of the Face Detection System. The researcher concludes that there is still a lot of development in terms of patches, fixes and improved system integration necessary to allow for the Face Detection System on the RISC based SoCKit to close the gap with a full pledge CISC CPU which according to the test result is around 43.44% difference.

5.2 Recommendation

The researcher highly recommends continued study in:

1. A Linux Environment including Kernel and GUI that is fully compatible with the SoCKit System and with the additional fixes of the issues encountered especially in UVC.
2. Adding the sound support for the Linux System. Future implementation may then support both Face and Audio Recognition.
3. Hardware Acceleration using FPGA component of the SoCKit for various algorithms that are involved in the OpenCV Library and various other libraries.

LIST OF REFERENCES

[1] "OpenCV." Wikipedia. Online. http://en.wikipedia.org/wiki/OpenCV.

[2] "Machine vision." Wikipedia. Online. http://en.wikipedia.org/wiki/Machine_vision.

[3] "System on a chip." Wikipedia. Online. http://en.wikipedia.org/wiki/System_on_a_chip.

[4] "Direct memory access." Wikipedia. Online. http://en.wikipedia.org/wiki/Direct_memory_access.

[5] "Field-programmable gate array." Wikipedia. Online. http://en.wikipedia.org/wiki/Field-Programmable_Gate_Array.

[6] "Central processing unit." Wikipedia. Online. http://en.wikipedia.org/wiki/Central_processing_unit.

[7] "ARM architecture." Wikipedia. Online. http://en.wikipedia.org/wiki/ARM_Architecture.

[8] "Reduced instruction set computing." Wikipedia. Online. http://en.wikipedia.org/wiki/RISC.

[9] "Complex instruction set computing." Wikipedia. Online. http://en.wikipedia.org/wiki/Complex_instruction_set_computing.

[10] "Video Graphics Array." Wikipedia. Online. http://en.wikipedia.org/wiki/Video_Graphics_Array.

[11] “Webcam.” Wikipedia. Online.
http://en.wikipedia.org/wiki/Webcam.

[12] “Ethernet.” Wikipedia. Online.
http://en.wikipedia.org/wiki/Ethernet.

[13] “Universal Serial Bus (USB).” Wikipedia. Online.
http://en.wikipedia.org/wiki/Universal_Serial_Bus.

[14] “USB On-The-Go.” Wikipedia. Online.
http://en.wikipedia.org/wiki/USB_On-The-Go.

[15] “Pixel.” Wikipedia. Online.
http://en.wikipedia.org/wiki/Pixels.

[16] “Microprocessor development board.” Wikipedia. Online.
http://en.wikipedia.org/wiki/Development_board.

[17] “ULPI - The Standard for High-Speed USB PHYs.” Mentor Graphics. Online.
http://www.mentor.com/products/ip/usb/usb20otg/phy_interfaces.

[18] “PHY (chip).” Wikipedia. Online.
http://en.wikipedia.org/wiki/PHY.

[19] “Unshielded twisted pair (UTP).” Wikipedia. Online.
http://en.wikipedia.org/wiki/Unshielded_twisted_pair.

[20] “Semiconductor intellectual property core.” Wikipedia. Online.
http://en.wikipedia.org/wiki/IP_Core.

[21] “Network on a chip.” Wikipedia. Online. http://en.wikipedia.org/wiki/Network_on_chip.

[22] “Arrow SoCKit Evaluation Board.” Rocket Boards Org. 2014. Online. 8 March, 2014 http://www.rocketboards.org/foswiki/Documentation/ArrowSoCKitEvaluationBoard.

[23] “Cyclone V SoC Hard Processor System.” Altera. Online. http://www.altera.com/devices/fpga/cyclone-v-fpgas/hard-processor-system/cyv-soc-hps.html.

[24] “Dual-Core ARM Cortex-A9 MPCore Processor.” Altera. Online. http://www.altera.com/devices/processor/arm/cortex-a9/m-arm-cortex-a9.html.

[25] Standard Microsystems Corporation, "Hi-Speed USB HOST, Device or OTG PHY with ULPI Low Pin Interface," 2007.

[26] Micrel, “KSZ9021RL/RN: Gigabit Ethernet Transceiver with RGMII Support”, October 2009

[27] Terasic Technologies Inc. “SoCKit: User Manual”, 2013

[28] “Logitech HD Webcam C525.” Logitech. Online. http://www.logitech.com/en-us/product/hd-webcam-c525

[29] “Secure Digital.” Wikipedia. Online. http://en.wikipedia.org/wiki/Micro_SD_Card.

[30] “Linux kernel.” Wikipedia. Online. http://en.wikipedia.org/wiki/Linux_Kernel.

[31] “Debian.” Wikipedia. Online. http://en.wikipedia.org/wiki/Debian.

[32] “LXDE.” Wikipedia. Online. http://en.wikipedia.org/wiki/LXDE.

[33] “Altera Quartus.” Wikipedia. Online. http://en.wikipedia.org/wiki/Altera_Quartus.

[34] “Qsys - Altera’s System Integration Tool.” Altera. Online. http://www.altera.com/products/software/quartus-ii/subscription-edition/qsys/qts-qsys.html.

[35] “SoC Embedded Design Suite.” Altera. Online. http://www.altera.com/devices/processor/arm/cortex-a9/software/proc-soc-embedded-design-suite.html.

[36] “Toolchain Working Group.” Linaro Open Source Organization. Online. https://wiki.linaro.org/WorkingGroups/ToolChain.

[37] “GNU Compiler Collection.” Wikipedia. Online. http://en.wikipedia.org/wiki/GNU_Compiler_Collection.

[38] “GitHub.” Wikipedia. Online. http://en.wikipedia.org/wiki/Github.

[39] “Win32DiskImager.” Ubuntu Wiki. Online. https://wiki.ubuntu.com/Win32DiskImager.

[40] “PuTTY.” Wikipedia. Online. http://en.wikipedia.org/wiki/PuTTY.

[41] “Face detection.” Wikipedia. Online. http://en.wikipedia.org/wiki/Face_detection.

[42] “Facial recognition system.” Wikipedia. Online. http://en.wikipedia.org/wiki/Face_recognition.

[43] “GPU Module Introduction.” OpenCV 2.4.8.0 Documentation. Online. http://docs.opencv.org/modules/gpu/doc/introduction.html.

[44] “Intel® IPP - Open Source Computer Vision Library (OpenCV) FAQ.” Intel Developer Zone. Online. http://software.intel.com/en-us/articles/intel-integrated-performance-primitives-intel-ipp-open-source-computer-vision-library-opencv-faq#performance.

[45] Neuendorffer, Stephen, Thomas Li, and Devin Wang (August 2013) “Accelerating OpenCV Applications with Zynq-7000 All Programmable SoC using Vivado HLS Video Libraries.” *Application Note: Vivado HLS*. Online. http://www.xilinx.com/support/documentation/application_notes/xapp1167.pdf.

[46] “USB video device class.” Wikipedia. Online. http://en.wikipedia.org/wiki/USB_video_device_class.

[47] “Linux UVC driver and tools.” Ideas On Board Organization. Online. http://www.ideasonboard.org/uvc/.

[48] Viola, P., Jones, M.J.: Rapid object detection using a boosted cascade of simple features. In: Proc. CVPR (2001)

[49] Schapire, Robert E.: Explaining Adaboost. Princeton University, Dept. of Computer Science, 35 Olden Street, Princeton, NJ 08540 USA

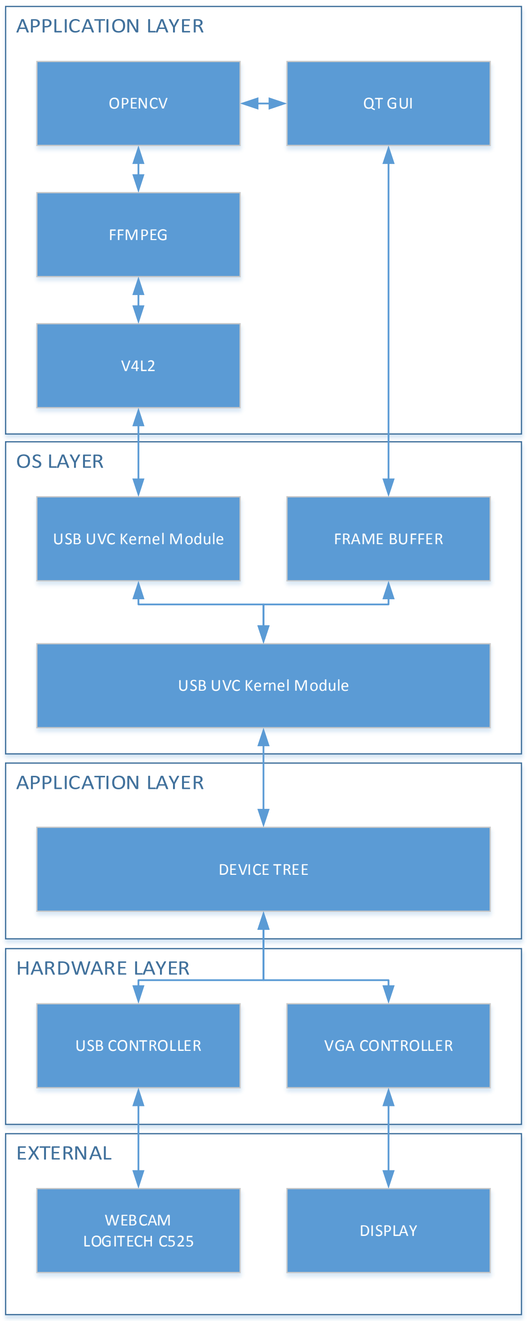
[50] J. Cho, S. Mrizaei, J. Oberg and R. Kastner, "FPGA-Based Face Detection System Using Haar Classifiers," ACM, Vols. 978-1-60558-410-2/09/02, 2009.

[51] H. Schneiderman and T. Kanade, "Object Detection Using the Statistics of Parts," International Journal of Computer Vision, vol. 56, no. 3, pp. 151-177, 2004.

[52] Theocharides, T., Vijayjrishman, N., Irwin, M.J.: A parallel architecture for hardware face detection. In: Emerging VLSI Technologies and Architectures (2006)

APPENDIX A

Expanded Block Diagram of the Face Detection System



APPENDIX B

SOCKIT DEVELOPMENT KITS & TOOLS DETAIL

Description	The SoCKit Development Kit presents a robust hardware design platform built around the Altera System-on-Chip (SoC) FPGA, which combines the latest dual-core Cortex-A9 embedded cores with industry-leading programmable logic for ultimate design flexibility. Users can now leverage the power of tremendous re-configurability paired with a high-performance, low-power processor system. Altera's SoC integrates an ARM-based hard processor system (HPS) consisting of processor, peripherals and memory interfaces tied seamlessly with the FPGA fabric using a high-bandwidth interconnect backbone. The SoCKit development board includes hardware such as high-speed DDR3 memory, video and audio capabilities, Ethernet networking, and much more. In addition, an on-board HSMC connector with high-speed transceivers allows for an even greater array of hardware setups. By leveraging all of these capabilities, the SoCKit is the perfect solution for showcasing, evaluating, and prototyping the true potential of the Altera SoC.
Features	FPGA Device • Cyclone V SoC 5CSXFC6D6F31 Device • Dual-core ARM Cortex-A9 (HPS) • 110K Programmable Logic Elements • 5,140 Kbits embedded memory • 6 Fractional PLLs • 2 Hard Memory Controllers • 3.125G Transceivers Configuration and Debug • Quad Serial Configuration device – EPCQ256 on FPGA • On-Board USB Blaster II (micro USB type B connector) Memory Device • 1GB (2x256MBx16) DDR3 SDRAM on FPGA • 1GB (2x256MBx16) DDR3 SDRAM on HPS

	• 128MB QSPI Flash on HPS • Micro SD Card Socket on HPS Communication • USB 2.0 OTG (ULPI interface with micro USB type AB connector) • USB to UART (micro USB type B connector) • 10/100/1000 Ethernet Connectors • One HSMC (8-channel Transceivers, Configurable I/O standards 1.5/1.8/2.5/3.3V) • One LTC connector (One Serial Peripheral Interface (SPI) Master ,one I2C and one GPIO interface) Display • 24-bit VGA DAC • 128x64 dots LCD Module with Backlight Audio • 24-bit CODEC, Line-in, line-out, and microphone-in jacks Switches, Buttons and LEDs • 8 User Keys (FPGA x4 ; HPS x 4) • 8 User Switches (FPGA x4 ; HPS x 4) • 8 User LEDs (FPGA x4 ; HPS x 4) • 2 HPS Reset Buttons (HPS_RSET_n and HPS_WARM_RST_n) Sensors • G-Sensor on HPS • Temperature Sensor on FPGA Power • 12V DC input
Kit Includes	• The SoCKit development board • USB Cable for FPGA programming and control • Ethernet Cable • 12V DC power adapter

APPENDIX C

ARM® Cortex™-A9 MPCore™ Specification

- 800 MHz dual-core processor supporting symmetric and asymmetric multiprocessing
- Each processor includes the following:
    - High-efficiency, dual-issue superscalar pipeline (2.5 MIPS* per MHz)
    - NEONTM media processing engine for media and signal processing acceleration
    - Single- and double-precision floating-point unit
    - 32 KB instruction and 32 KB data caches
    - Cache coherence for enhanced inter-processor communication
    - Memory Management Unit with TrustZone® security technology
    - Thumb®-2 technology for enhanced code density, performance, and power efficiency
    - Jazelle® architecture extensions for accelerating Java Virtual Machine
    - Program Trace Macrocell for full visibility of processor instruction flow
- Shared 512 KB, 8-way associative L2 cache, lockable by way, line, or master
- Acceleration coherency port that extends coherent memory access beyond the CPUs
- Generic interrupt controller
- 32 bit general purpose timer
- Watchdog timer
- Available in Altera® Arria® V SoCs and Cyclone® V SoCs

APPENDIX D

FPGA Configuration Mode (Taken from SoCKit User Manual)

3.1.2 FPGA Configuration Mode Switch

The Dipswitch SW6 (See Figure 3-3) can set the MSEL pins to decide the FPGA configuration modes. Table 3-2 shows the switch controls and descriptions. Table 3-3 gives the MSEL pins setting for each configuration scheme of Cyclone V devices. FPGA default works in ASx4 mode. However, once the FPGA is in AS x4 mode, and after successfully configuring the FPGA via the EPCQ256, the SoCKit will be unable to boot Linux from the SD card or other devices. Please switch SW6 to another mode (e.g. MSEL[4:0] = 10000) to enable normal operations of Linux.

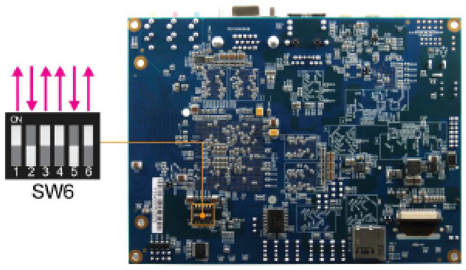


Figure 3-3 FPGA Configuration Mode Switch

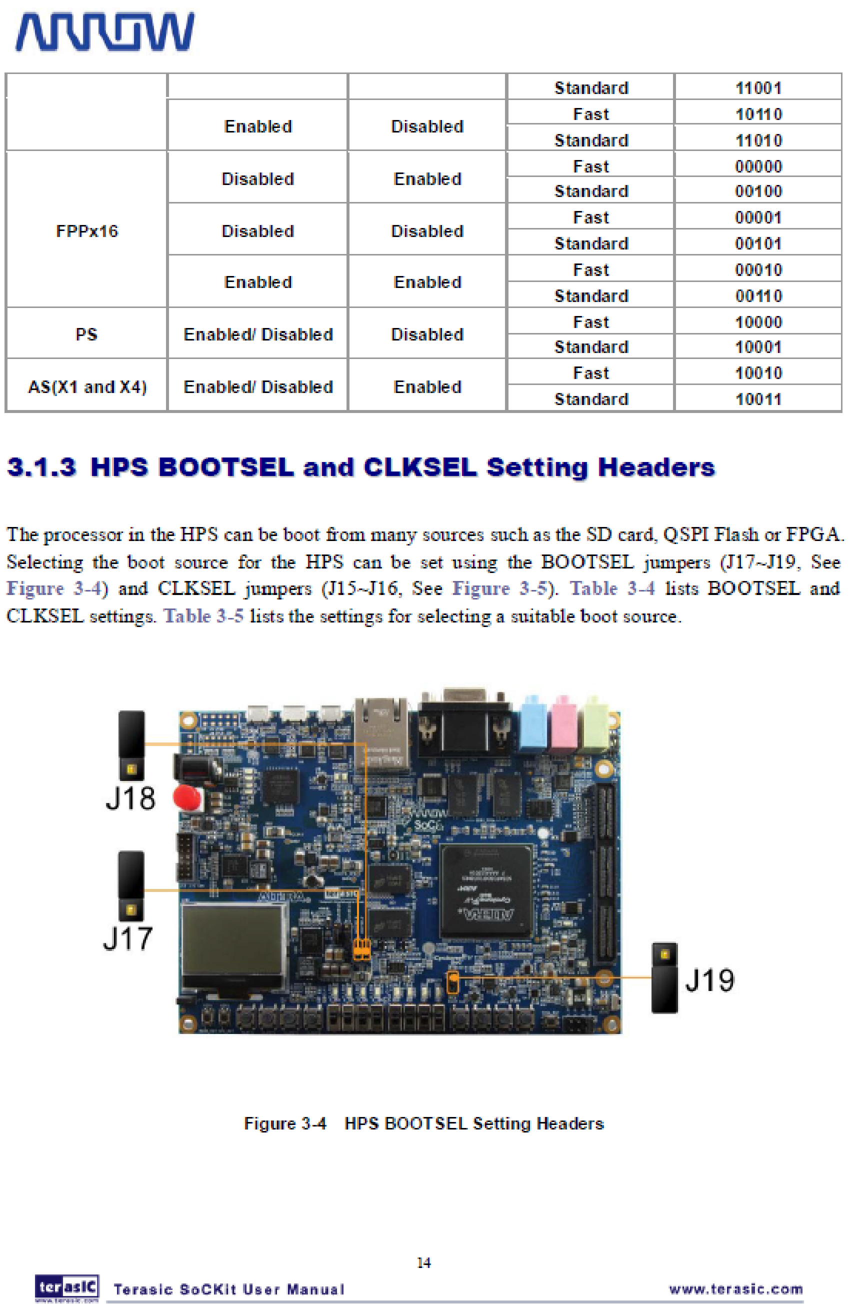
Table 3-2 SW6 FPGA Configuration Mode Switch

Board Reference	*Signal Name*	*Description*	*Default*
SW6.1	MSEL0	Sets the Cyclone V MSEL[4:0] pins. Use these pins to set the configuration scheme and POR delay.	On
SW6.2	MSEL1		Off
SW6.3	MSEL2		On
SW6.4	MSEL3		On
SW6.5	MSEL4		Off
SW6.6	N/A	N/A	N/A

Table 3-3 MSEL pin Settings for each Scheme of Cyclone V Device

Configuration Scheme	*Compression Feature*	*Design Security Feature*	*POR Delay*	*Valid MSEL[4:0]*
FPPx8	Disabled	Disabled	Fast	10100
			Standard	11000
	Disabled	Enabled	Fast	10101

FPGA Configuration Mode (Taken from SoCKit User Manual)



			Standard	11001
	Enabled	Disabled	Fast	10110
			Standard	11010
FPPx16	Disabled	Enabled	Fast	00000
			Standard	00100
	Disabled	Disabled	Fast	00001
			Standard	00101
	Enabled	Enabled	Fast	00010
			Standard	00110
PS	Enabled/ Disabled	Disabled	Fast	10000
			Standard	10001
AS(X1 and X4)	Enabled/ Disabled	Enabled	Fast	10010
			Standard	10011

3.1.3 HPS BOOTSEL and CLKSEL Setting Headers

The processor in the HPS can be boot from many sources such as the SD card, QSPI Flash or FPGA. Selecting the boot source for the HPS can be set using the BOOTSEL jumpers (J17~J19, See Figure 3-4) and CLKSEL jumpers (J15~J16, See Figure 3-5). Table 3-4 lists BOOTSEL and CLKSEL settings. Table 3-5 lists the settings for selecting a suitable boot source.

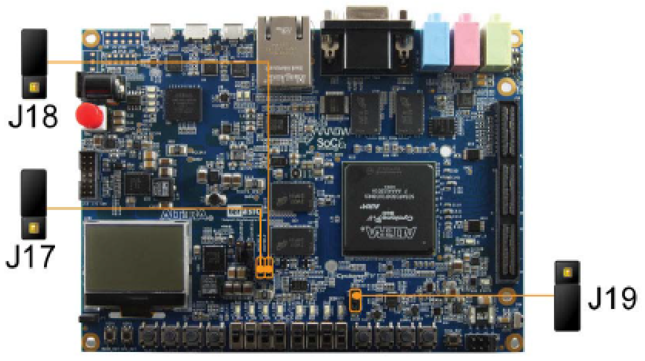


Figure 3-4 HPS BOOTSEL Setting Headers

14

Terasic SoCKit User Manual www.terasic.com

FPGA Configuration Mode (Taken from SoCKit User Manual)

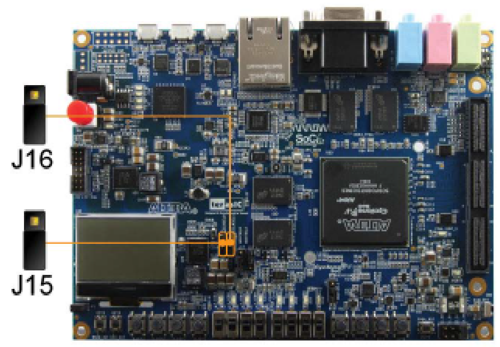


Figure 3-5 HPS CLKSEL Setting Headers

Table 3-4 HPS BOOTSEL and CLKSEL Setting Headers

Board Reference	Signal Name	Setting	Default
J17	BOOTSEL0	Short Pin 1 and 2: Logic 1 Short Pin 2 and 3: Logic 0	Short Pin 1 and 2
J19	BOOTSEL1	Short Pin 1 and 2: Logic 1 Short Pin 2 and 3: Logic 0	Short Pin 2 and 3
J18	BOOTSEL2	Short Pin 1 and 2: Logic 1 Short Pin 2 and 3: Logic 0	Short Pin 1 and 2
J15	CLKSEL0	Short Pin 1 and 2: Logic 1 Short Pin 2 and 3: Logic 0	Short Pin 2 and 3
J16	CLKSEL1	Short Pin 1 and 2: Logic 1 Short Pin 2 and 3: Logic 0	Short Pin 2 and 3

Table 3-5 BOOTSEL[2:0] Setting Values and Flash Device Selection

BOOTSEL[2:0] Setting Value	Flash Device
000	Reserved
001	FPGA (HPS-to-FPGA bridge)
010	1.8 V NAND Flash memory (*1)
011	3.0 V NAND Flash memory(*1)
100	1.8 V SD/MMC Flash memory(*1)
101	3.0 V SD/MMC Flash memory
110	1.8 V SPI or quad SPI Flash memory(*1)
111	3.0 V SPI or quad SPI Flash memory

(*1) : Not supported on SoCKit board

APPENDIX E

Configuring U-Boot to the Proper Locations of Kernel Program

```
bootcmd=run mmcload; run fpgaload; run mmcboot

fpga=0

fpgadata=0x2000000

fpgadatasize=2126D5

fpgaload=fatload mmc 0:1 0x2000000 soc_system.rbf; fpga load 0
${fpgadata} ${fi}
```

Configuring the Serial Console to Port 57600 for Debugging

```
setenv mmcboot 'setenv bootargs console=tty0 console=ttyS0,57600
root=${mmcroot} rw rootwait;bootm ${loadaddr} - ${fdtaddr}'
```

APPENDIX F

From RocketBoards Org On How to Partition a uSD Card.

Partition the SD Card

1. Determine the device associated with the SD card on the host by running the following command before and after inserting the card in the reader:

```
$ cat /proc/partitions
Let's assume it is /dev/sdx
```

2. Repartition the card, with the sudo fdisk/dev/sdx command

```
$ Use p to print current partitioning
$ Use d to delete all partitions one by one
Create partition using these options, pressing enter
after each one: n p 2 4096 +4496384 t 83
Create partition using these options, pressing enter
after each one: n p 1 9000000 +20480K t 1 b
Create partition using these options, pressing enter
after each one: n p 3 2048 +1024K t 3 a2
```

3. Press w to write partitions to card, then exit

```
Verify the card was partitioned correctly, it should
look like this (use fdisk and p to see it):
Device Boot Start End Blocks Id System
/dev/sdx1 9000000 9020480 20480 b W95 FAT32
/dev/sdx2 4096 8996863 4496384 83 Linux
/dev/sdx3 2048 4095 1024 a2 Unknown
```

4. Make partition one a DOS partition

```
$ sudo mkdosfs /dev/sdx1
```

APPENDIX G

Specs of the Intel CISC based system

```
Host Name:                 ANDROMEDA
OS Name:                   Microsoft Windows 7 Ultimate
OS Version:                6.1.7601 Service Pack 1 Build 7601
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Standalone Workstation
OS Build Type:             Multiprocessor Free
Registered Owner:          Bangonkali
Registered Organization:
Product ID:                00426-OEM-8992662-00006
Original Install Date:     11/30/2013, 1:30:10 AM
System Boot Time:          3/13/2014, 9:30:26 PM
System Manufacturer:       Acer
System Model:              Aspire 4755
System Type:               x64-based PC
Processor(s):              1 Processor(s) Installed.
                           [01]: Intel64 Family 6 Model 42
Stepping 7 GenuineInt
el ~792 Mhz
BIOS Version:              Phoenix Technologies Ltd. V2.13,
10/21/2011
Windows Directory:         C:\Windows
System Directory:          C:\Windows\system32
Boot Device:               \Device\HarddiskVolume1
System Locale:             en-us;English (United States)
Input Locale:              en-us;English (United States)
Time Zone:                 (UTC+08:00) Taipei
Total Physical Memory:     8,003 MB
Available Physical Memory: 4,280 MB
Virtual Memory: Max Size:  16,005 MB
Virtual Memory: Available: 12,066 MB
Virtual Memory: In Use:    3,939 MB
Page File Location(s):     C:\pagefile.sys
Domain:                    WORKGROUP
Logon Server:              \\ANDROMEDA
```

APPENDIX H

Kernel Panic Logs on Continue Long Term Operation of Face Detection System

```
Unable to handle kernel NULL pointer dereference at virtual address
0000003c
pgd = 80004000
[0000003c] *pgd=00000000
Internal error: Oops: 17 [#1] SMP ARM
Modules linked in: ipv6 uvcvideo videobuf2_core videodev
videobuf2_vmalloc video                                   buf2_memops
gpio_dw
CPU: 0    Not tainted  (3.8.0-00111-g85cc90f #1)
PC is at usb_hcd_unmap_urb_setup_for_dma+0x8/0xb0
LR is at usb_hcd_unmap_urb_for_dma+0x14/0x134
pc : [<8029b110>]    lr : [<8029b1cc>]    psr: a0000193
sp : bed8bc88  ip : 00000000  fp : 00000001
r10: be11b0d0  r9 : 80559a34  r8 : 00000000
r7 : 00000018  r6 : 000000a0  r5 : be11b000  r4 : 00000000
r3 : bc194b08  r2 : 800400c7  r1 : 00000000  r0 : be11b000
Flags: NzCv  IRQs off  FIQs on  Mode SVC_32  ISA ARM  Segment user
Control: 10c5387d  Table: 3c01c04a  DAC: 00000015
Process facedetect (pid: 1646, stack limit = 0xbed8a240)
Stack: (0xbed8bc88 to 0xbed8c000)
bc80:                   00000000 be11b000 000000a0 8029b1cc be11b0fc
be11b104
bca0: be010a10 bebf2280 0000000c 000000a0 00000018 802b2738 bed8bd14
00001000
bcc0: 00000081 80546800 00000001 00001000 805467e8 8024db6c 0000004c
00000033
bce0: 20b48bfc bebf2280 c0d805e0 802afd08 8001c960 00000001 000003fc
00000040
bd00: 00000000 bebf2280 00000000 00000000 0000b071 60000193 be010a10
000000a0
bd20: 00000000 00000000 80559a34 80559a20 00000000 802af9f8 802af9ec
8029ab1c
bd40: bebf1100 80075bf0 00000001 bebf2ec0 00000002 be0109c0 bed8bdb4
be0109c0
bd60: be010a10 00000000 fee00100 be11b0d0 00000000 fffffffe a0000013
80075e24
bd80: be0109c0 be010a10 00000000 80078bac 80078b30 000000a0 000000a0
800755d4
bda0: 80525030 8000f014 fee0010c 8052e418 bed8bdd0 80008530 802b070c
803b6a88
bdc0: 60000013 ffffffff bed8be04 8000dd40 be11b0d0 a0000013 00000000
8a258a25
```

```
bde0: bedb4e00 be5f7000 bc194b00 be11b000 be11b0d0 00000000 fffffffe
a0000013
be00: 0000003d bed8be18 802b070c 803b6a88 60000013 ffffffff 802b0644
fffffffe
be20: be5f7000 be11b000 fffffffe be419000 be31b788 00000001 bed8a000
8029b3f8
be40: be1c0400 fffffffe be5f7000 60000013 00000000 be419000 00000001
8029c708
be60: be5f7000 be5f700c 00000001 8029d590 bed8beac be41900c be5f6000
00000002
be80: 00000000 be419000 be31b788 be419010 be5f7000 7f041a68 00000001
be419000
bea0: 00000000 00000000 be419000 00000008 00000000 7f043e54 00000001
be43bd80
bec0: be31b780 bd11ea18 be419000 7f03ff78 7f03fee0 be23fc00 be664e88
bd11ea18
bee0: be05b310 7f014390 7f014358 be31b780 be664e88 800db0d8 00000000
00000000
bf00: be2d283c beba66c0 805618c4 be2d2580 8000e2a8 bed8a000 00000000
8003e488
bf20: be1e68c0 fd87fd87 be2d2580 be1e68c0 be2d284c be2d2580 be1e68fc
800265c8
bf40: be1e68c0 00000000 bed8a000 00000001 00382000 800c4960 fd87fd87
800da1ac
bf60: be1e6900 be57f680 00000000 bed8a000 000000f8 8000e2a8 bed8a000
00000000
bf80: 00000000 80026d88 00000000 000703c2 7698e760 7698e760 000000f8
80026e2c
bfa0: 00000000 8000e100 000703c2 7698e760 00000000 000703ae 74c214c0
00000000
bfc0: 000703c2 7698e760 7698e760 000000f8 00000000 00000000 76fa1000
00000000
bfe0: 000000f8 7eab82bc 7691fce3 768c71e6 600f0030 00000000 00000000
00000000
[<8029b110>] (usb_hcd_unmap_urb_setup_for_dma+0x8/0xb0) from [<be11b104>]
(0xbe1                                   1b104)
Code: 8055994b 80497e64 e92d4070 e1a04001 (e591303c)
---[ end trace 7240ef6d486fb03d ]---
```

APPENDIX I

Researcher’s Notes on SoCKit Usage

```
Sockit usage notes:

As root:

sudo chmod u+s /usr/bin/sudo

su linaro
sudo apt-get update && \
sudo apt-get upgrade && \
sudo ntpdate time.nist.gov && \

sudo apt-get install wget build-essential checkinstall libncurses5-dev libncurses5
libevent-1.4 libevent-dev libncurses5-dev pkg-config libpng12-0 libpng12-dev libpng++-
dev libpng3 libpnglite-dev zlib1g-dbg zlib1g zlib1g-dev pngtools libtiff4-dev libtiff4
libtiffxx0c2 libtiff-tools ffmpeg  v4l-conf v4l-utils v4l2ucp  libjpeg-dev  libpng-dev
libtiff-dev libjasper-dev libavcodec-dev libavformat-dev libswscale-dev usbutils
gparted vim less psmisc lynx gcc g++ xserver-xorg xterm firefox mplayer lxde xinit x11-
apps xinput-calibrator mesa-utils prboom
```

```
Check webcam

v4l2-ctl --list-formats
v4l2-ctl --list-formats-ext
v4l2-ctl --set-fmt-video=width=1280,height=720,pixelformat=1
v4l2-ctl --set-fmt-video=width=640,height=480,pixelformat=1
v4l2-ctl --set-parm=30

lsusb
```

```
1. Find & install CMAKE
download cmake
http://www.cmake.org/cmake/resources/software.html

tar xf cmake*
cd cmake*
./bootstrap && make && sudo make install
```

```
Installing Synergy
sudo apt-get install xorg-dev libqt4-dev python
 libcurl4-openssl-dev [cant be detected]
 python-dev make gcc

http://synergy-foss.org/wiki/Compiling

./hm.sh conf -g1 --make-gui
./hm.sh build --make-gui
```

```
HTOP
```

```
svn co https://htop.svn.sourceforge.net/svnroot/htop/trunk htop


--disable-unicode
```

```
Installing OpenSSH
http://www.unixwiz.net/techtips/openssh.html#tarballs


./configure --with-ipv4-default --with-md5-passwords --with-pam && make && make install

ssdh privilege escalation
groupadd sshd
useradd -M -g sshd -c 'sshd privsep' -d /var/empty -s /sbin/nologin sshd
passwd -l sshd

useradd -g root linaro


building cmake
http://www.cmake.org/cmake/help/install.html

http://docs.opencv.org/doc/tutorials/introduction/linux_install/linux_install.html

sudo apt-get install ffmpeg && \
sudo apt-get install v4l-conf v4l-utils v4l2ucp && \
sudo apt-get install libjpeg-dev  libpng-dev libtiff-dev libjasper-dev && \
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev
```

```
time
sudo apt-get install ntp

Probably the ntp service is running, that's why ntpdate can't open the socket (port 123
UDP) and connect to ntp server.

Try from command line:

sudo service ntp stop
sudo ntpdate -s time.nist.gov
sudo service ntp start
If you want to put this in /etc/rc.local use the following:

( /etc/init.d/ntp stop
until ping -nq -c3 8.8.8.8; do
   echo "Waiting for network..."
done
ntpdate -s time.nist.gov
/etc/init.d/ntp start )&

http://askubuntu.com/questions/254826/how-to-force-a-clock-update-using-ntp
https://help.ubuntu.com/10.04/serverguide/NTP.html
```

```
chmod -R 777 /tmp
chmod -R 777 /usr/local/src
chown -R root:root /usr/bin/sudo
```

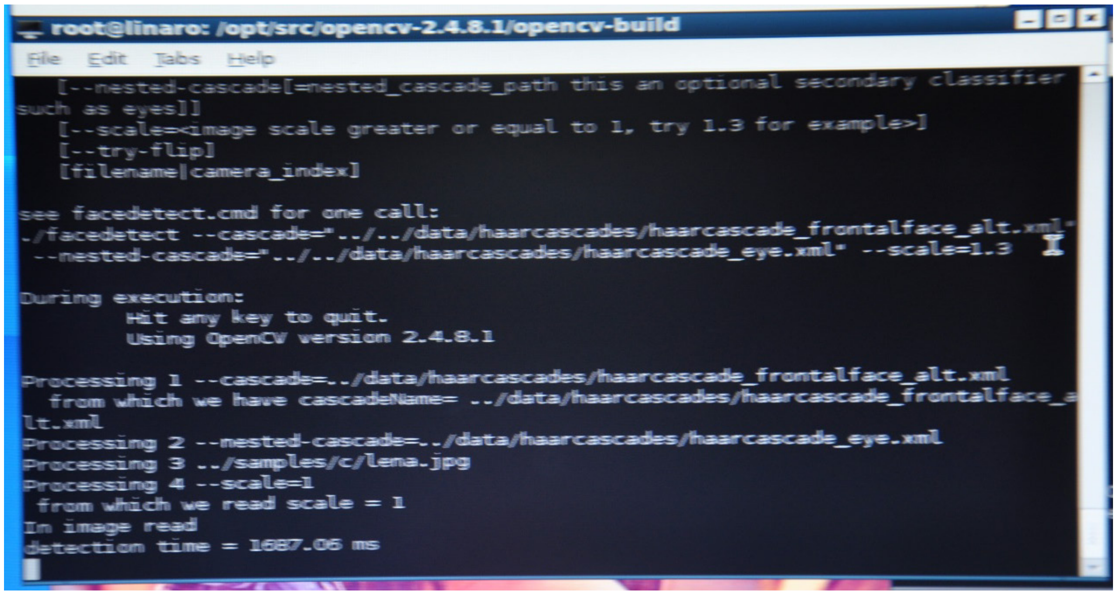
```
chmod 4755 /usr/bin/sudo

apt-get update
apt-get dist-upgrade
apt-get install wget

#!/bin/bash
cd /usr/local/src

//zlib
wget http://zlib.net/zlib-1.2.8.tar.gz
wget http://www.openssl.org/source/openssl-1.0.1f.tar.gz
wget ftp://ftp3.usa.openbsd.org/pub/OpenBSD/OpenSSH/portable/openssh-6.4p1.tar.gz

tar -xf openssh-6.4p1.tar.gz
tar -xf openssl-1.0.1f.tar.gz
tar -xf zlib-1.2.8.tar.gz

cd /usr/local/src

cd zlib-1.2.8
./configure && make && sudo make install

cd ../openssl-1.0.1f
./config && make && make test && sudo make install

cd ../openssh-6.4p1
./configure --with-ipv4-default --with-md5-passwords && make
sudo groupadd sshd
sudo useradd -M -g sshd -c 'sshd privsep' -d /var/empty -s /sbin/nologin sshd
sudo passwd -l sshd
sudo make install
```

```
// set time
http://askubuntu.com/questions/254826/how-to-force-a-clock-update-using-ntp
apt-get install ntp
service ntp start
sudo service ntp stop
sudo ntpdate -s time.nist.gov
sudo service ntp start

//ssh
http://www.unixwiz.net/techtips/openssh.html

sshd re-exec requires execution with an absolute path


// install x11


// install arm compiler
sudo apt-get install libc6-dev-i386
download source
compile source

http://askubuntu.com/questions/251978/cannot-find-crti-o-no-such-file-or-directory
apt-get install libc-dev

export LIBRARY_PATH=/usr/lib32:/usr/lib/x86_64-linux-gnu
export LIBRARY_PATH=/usr/lib/x86_64-linux-gnu/

http://askubuntu.com/questions/98416/error-kernel-headers-not-found-but-they-are-in-
place
sudo apt-get install linux-headers-$(uname-r)

http://stackoverflow.com/questions/11471722/libstdc-so-6-cannot-open-shared-object-
file-no-such-file-or-directory
sudo apt-get install lib32z1
sudo apt-get install lib32z1 lib32z1-dev
sudo apt-get install lib32stdc++6
sudo apt-get install ia32-libs
sudo apt-get install uboot-mkimage
```

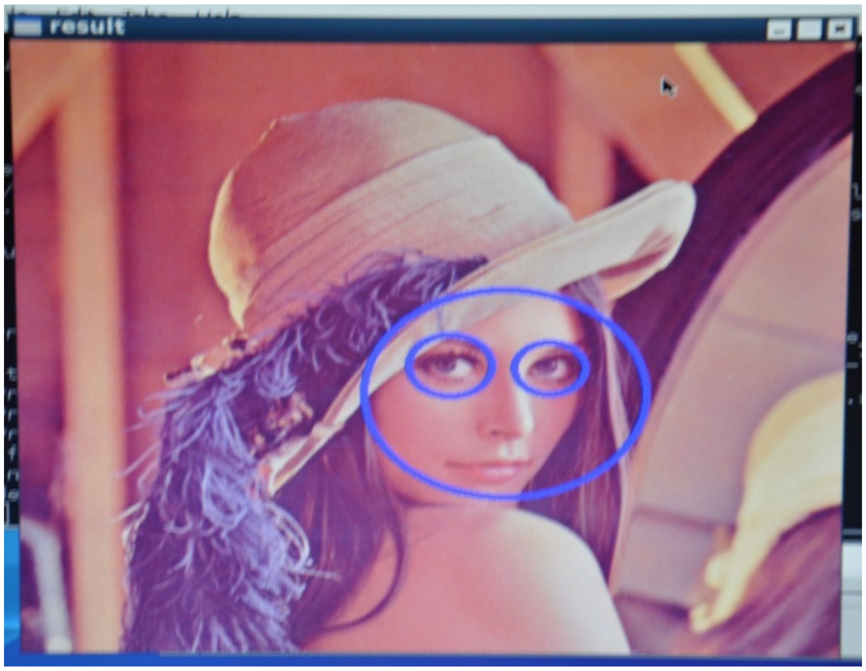
```
CHANGING IMAGE on Boot

http://sumanprasanna.wordpress.com/2012/12/11/how-to-change-your-boot-logo-in-linux-
kernel/
```

APPENDIX J

Face Detection System SoCKit Test Results



```
   [--nested-cascade[=nested_cascade_path this an optional secondary classifier
such as eyes]]
   [--scale=<image scale greater or equal to 1, try 1.3 for example>]
   [--try-flip]
   [filename|camera_index]

see facedetect.cmd for one call:
./facedetect --cascade="../../data/haarcascades/haarcascade_frontalface_alt.xml"
 --nested-cascade="../../data/haarcascades/haarcascade_eye.xml" --scale=1.3

During execution:
        Hit any key to quit.
        Using OpenCV version 2.4.8.1

Processing 1 --cascade=../data/haarcascades/haarcascade_frontalface_alt.xml
  from which we have cascadeName= ../data/haarcascades/haarcascade_frontalface_a
lt.xml
Processing 2 --nested-cascade=../data/haarcascades/haarcascade_eye.xml
Processing 3 ../samples/c/lena.jpg
Processing 4 --scale=1
 from which we read scale = 1
In image read
detection time = 1687.06 ms
```

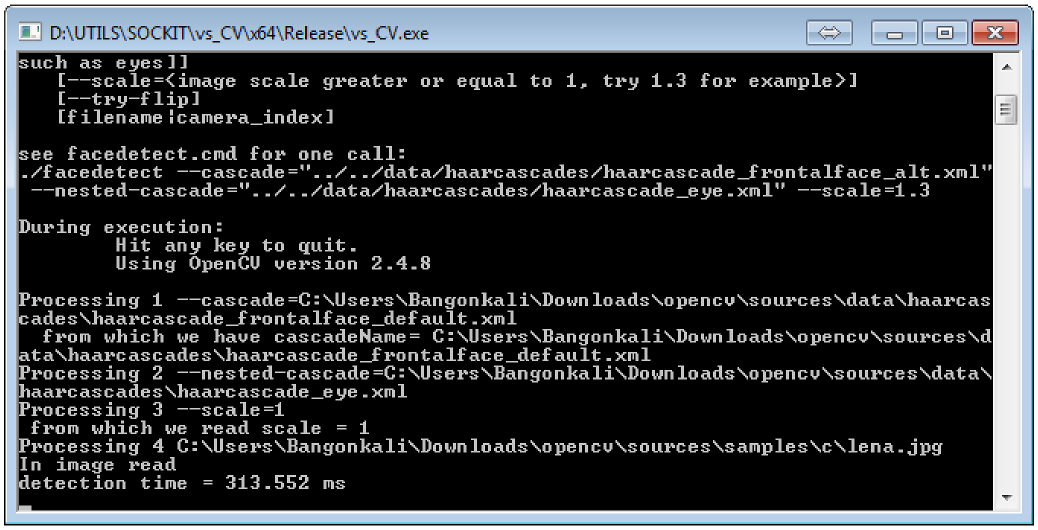


APPENDIX K

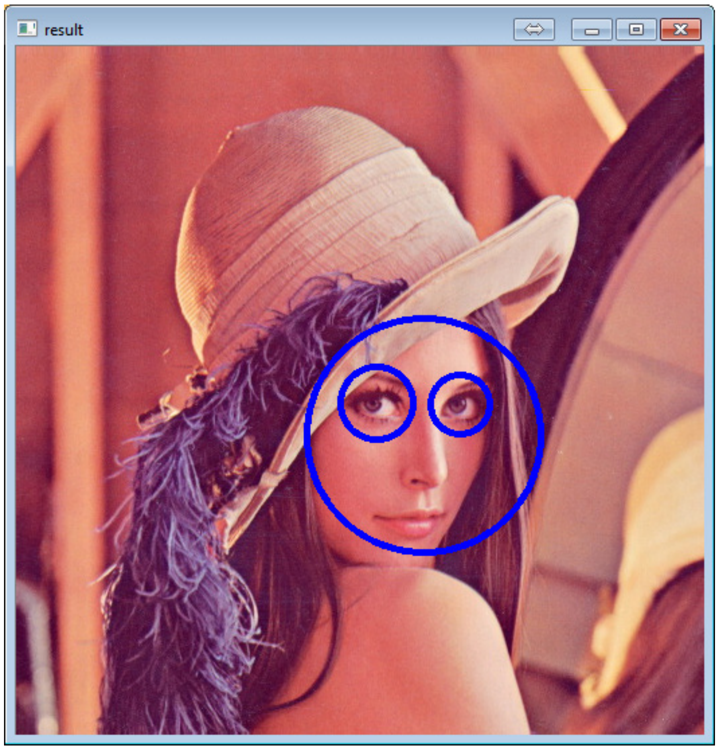
Standard 512x512 pixels test image.

APPENDIX L

Intel CISC Based Core i7 Test Results



```
such as eyes]]
   [--scale=<image scale greater or equal to 1, try 1.3 for example>]
   [--try-flip]
   [filename|camera_index]

see facedetect.cmd for one call:
./facedetect --cascade="../../data/haarcascades/haarcascade_frontalface_alt.xml"
 --nested-cascade="../../data/haarcascades/haarcascade_eye.xml" --scale=1.3

During execution:
        Hit any key to quit.
        Using OpenCV version 2.4.8

Processing 1 --cascade=C:\Users\Bangonkali\Downloads\opencv\sources\data\haarcas
cades\haarcascade_frontalface_default.xml
  from which we have cascadeName= C:\Users\Bangonkali\Downloads\opencv\sources\d
ata\haarcascades\haarcascade_frontalface_default.xml
Processing 2 --nested-cascade=C:\Users\Bangonkali\Downloads\opencv\sources\data\
haarcascades\haarcascade_eye.xml
Processing 3 --scale=1
 from which we read scale = 1
Processing 4 C:\Users\Bangonkali\Downloads\opencv\sources\samples\c\lena.jpg
In image read
detection time = 313.552 ms
```



APPENDIX M

Source code used for performance comparison facedetect.cpp

```
#include "stdafx.h"
#include "opencv2/objdetect/objdetect.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"


#include <cctype>
#include <iostream>
#include <iterator>
#include <stdio.h>

using namespace std;
using namespace cv;

static void help()
{
    cout << "\nThis program demonstrates the cascade recognizer. Now you can
use Haar or LBP features.\n"
            "This classifier can recognize many kinds of rigid objects, once
the appropriate classifier is trained.\n"
            "It's most known use is for faces.\n"
            "Usage:\n"
            "./facedetect [--cascade=<cascade_path> this is the primary
trained classifier such as frontal face]\n"
               "   [--nested-cascade[=nested_cascade_path this an optional
secondary classifier such as eyes]]\n"
               "   [--scale=<image scale greater or equal to 1, try 1.3 for
example>]\n"
               "   [--try-flip]\n"
               "   [filename|camera_index]\n\n"
            "see facedetect.cmd for one call:\n"
            "./facedetect --
cascade=\"../../data/haarcascades/haarcascade_frontalface_alt.xml\" --nested-
cascade=\"../../data/haarcascades/haarcascade_eye.xml\" --scale=1.3\n\n"
            "During execution:\n\tHit any key to quit.\n"
            "\tUsing OpenCV version " << CV_VERSION << "\n" << endl;
}

void detectAndDraw( Mat& img, CascadeClassifier& cascade,
                    CascadeClassifier& nestedCascade,
                    double scale, bool tryflip );

string cascadeName =
"../../data/haarcascades/haarcascade_frontalface_alt.xml";
string nestedCascadeName =
"../../data/haarcascades/haarcascade_eye_tree_eyeglasses.xml";

int main( int argc, const char** argv )
```

```
{
    CvCapture* capture = 0;
    Mat frame, frameCopy, image;
    const string scaleOpt = "--scale=";
    size_t scaleOptLen = scaleOpt.length();
    const string cascadeOpt = "--cascade=";
    size_t cascadeOptLen = cascadeOpt.length();
    const string nestedCascadeOpt = "--nested-cascade";
    size_t nestedCascadeOptLen = nestedCascadeOpt.length();
    const string tryFlipOpt = "--try-flip";
    size_t tryFlipOptLen = tryFlipOpt.length();
    string inputName;
    bool tryflip = false;

    help();

    CascadeClassifier cascade, nestedCascade;
    double scale = 1;

    for( int i = 1; i < argc; i++ )
    {
        cout << "Processing " << i << " " <<  argv[i] << endl;
        if( cascadeOpt.compare( 0, cascadeOptLen, argv[i], cascadeOptLen ) ==
0 )
        {
            cascadeName.assign( argv[i] + cascadeOptLen );
            cout << "  from which we have cascadeName= " << cascadeName <<
endl;
        }
        else if( nestedCascadeOpt.compare( 0, nestedCascadeOptLen, argv[i],
nestedCascadeOptLen ) == 0 )
        {
            if( argv[i][nestedCascadeOpt.length()] == '=' )
                nestedCascadeName.assign( argv[i] + nestedCascadeOpt.length()
+ 1 );
            if( !nestedCascade.load( nestedCascadeName ) )
                cerr << "WARNING: Could not load classifier cascade for
nested objects" << endl;
        }
        else if( scaleOpt.compare( 0, scaleOptLen, argv[i], scaleOptLen ) ==
0 )
        {
            if( !sscanf( argv[i] + scaleOpt.length(), "%lf", &scale ) ||
scale < 1 )
                scale = 1;
            cout << " from which we read scale = " << scale << endl;
        }
        else if( tryFlipOpt.compare( 0, tryFlipOptLen, argv[i], tryFlipOptLen
) == 0 )
        {
            tryflip = true;
            cout << " will try to flip image horizontally to detect
assymetric objects\n";
        }
        else if( argv[i][0] == '-' )
        {
            cerr << "WARNING: Unknown option %s" << argv[i] << endl;
        }
        else
            inputName.assign( argv[i] );
    }

    if( !cascade.load( cascadeName ) )
    {
        cerr << "ERROR: Could not load classifier cascade" << endl;
        help();
        return -1;
    }

    if( inputName.empty() || (isdigit(inputName.c_str()[0]) &&
inputName.c_str()[1] == '\0') )
    {
        capture = cvCaptureFromCAM( inputName.empty() ? 0 :
inputName.c_str()[0] - '0' );
        int c = inputName.empty() ? 0 : inputName.c_str()[0] - '0' ;
        if(!capture) cout << "Capture from CAM " <<  c << " didn't work" <<
endl;
    }
    else if( inputName.size() )
    {
        image = imread( inputName, 1 );
        if( image.empty() )
        {
            capture = cvCaptureFromAVI( inputName.c_str() );
            if(!capture) cout << "Capture from AVI didn't work" << endl;
        }
    }
    else
    {
        image = imread( "lena.jpg", 1 );
        if(image.empty()) cout << "Couldn't read lena.jpg" << endl;
    }

    cvNamedWindow( "result", 1 );

    if( capture )
    {
        cout << "In capture ..." << endl;
        for(;;)
        {
            IplImage* iplImg = cvQueryFrame( capture );
            frame = iplImg;
            if( frame.empty() )
                break;
            if( iplImg->origin == IPL_ORIGIN_TL )
                frame.copyTo( frameCopy );
            else
                flip( frame, frameCopy, 0 );
            detectAndDraw( frameCopy, cascade, nestedCascade, scale, tryflip
);

            if( waitKey( 10 ) >= 0 )
                goto _cleanup_;
        }

        waitKey(0);

_cleanup_:
        cvReleaseCapture( &capture );
    }
    else
    {
        cout << "In image read" << endl;
        if( !image.empty() )
        {
            detectAndDraw( image, cascade, nestedCascade, scale, tryflip );
            waitKey(0);
        }
        else if( !inputName.empty() )
        {
            /* assume it is a text file containing the
            list of the image filenames to be processed - one per line */
            FILE* f = fopen( inputName.c_str(), "rt" );
            if( f )
            {
                char buf[1000+1];
                while( fgets( buf, 1000, f ) )
                {
                    int len = (int)strlen(buf), c;
                    while( len > 0 && isspace(buf[len-1]) )
                        len--;
                    buf[len] = '\0';
                    cout << "file " << buf << endl;
                    image = imread( buf, 1 );
                    if( !image.empty() )
                    {
                        detectAndDraw( image, cascade, nestedCascade, scale,
tryflip );
                        c = waitKey(0);
                        if( c == 27 || c == 'q' || c == 'Q' )
                            break;
                    }
                    else
                    {
                        cerr << "Aw snap, couldn't read image " << buf <<
endl;
                    }
                }
                fclose(f);
            }
        }
    }

    cvDestroyWindow("result");

    return 0;
}

void detectAndDraw( Mat& img, CascadeClassifier& cascade,
                    CascadeClassifier& nestedCascade,
                    double scale, bool tryflip )
{
    int i = 0;
    double t = 0;
    vector<Rect> faces, faces2;
    const static Scalar colors[] =  { CV_RGB(0,0,255),
        CV_RGB(0,128,255),
        CV_RGB(0,255,255),
        CV_RGB(0,255,0),
        CV_RGB(255,128,0),
        CV_RGB(255,255,0),
        CV_RGB(255,0,0),
        CV_RGB(255,0,255)} ;
    Mat gray, smallImg( cvRound (img.rows/scale), cvRound(img.cols/scale),
CV_8UC1 );

    cvtColor( img, gray, CV_BGR2GRAY );
    resize( gray, smallImg, smallImg.size(), 0, 0, INTER_LINEAR );
    equalizeHist( smallImg, smallImg );

    t = (double)cvGetTickCount();
    cascade.detectMultiScale( smallImg, faces,
        1.1, 2, 0
        //|CV_HAAR_FIND_BIGGEST_OBJECT
        //|CV_HAAR_DO_ROUGH_SEARCH
        |CV_HAAR_SCALE_IMAGE
        ,
        Size(30, 30) );
    if( tryflip )
    {
        flip(smallImg, smallImg, 1);
        cascade.detectMultiScale( smallImg, faces2,
                                 1.1, 2, 0
                                 //|CV_HAAR_FIND_BIGGEST_OBJECT
                                 //|CV_HAAR_DO_ROUGH_SEARCH
                                 |CV_HAAR_SCALE_IMAGE
                                 ,
                                 Size(30, 30) );
        for( vector<Rect>::const_iterator r = faces2.begin(); r !=
faces2.end(); r++ )
        {
            faces.push_back(Rect(smallImg.cols - r->x - r->width, r->y, r-
>width, r->height));
        }
    }
    t = (double)cvGetTickCount() - t;
    printf( "detection time = %g ms\n",
t/((double)cvGetTickFrequency()*1000.) );
    for( vector<Rect>::const_iterator r = faces.begin(); r != faces.end();
r++, i++ )
    {
        Mat smallImgROI;
        vector<Rect> nestedObjects;
        Point center;
        Scalar color = colors[i%8];
        int radius;

        double aspect_ratio = (double)r->width/r->height;
        if( 0.75 < aspect_ratio && aspect_ratio < 1.3 )
        {
            center.x = cvRound((r->x + r->width*0.5)*scale);
            center.y = cvRound((r->y + r->height*0.5)*scale);
            radius = cvRound((r->width + r->height)*0.25*scale);
            circle( img, center, radius, color, 3, 8, 0 );
        }
        else
            rectangle( img, cvPoint(cvRound(r->x*scale), cvRound(r-
>y*scale)),
                       cvPoint(cvRound((r->x + r->width-1)*scale),
cvRound((r->y + r->height-1)*scale)),
                       color, 3, 8, 0);
        if( nestedCascade.empty() )
            continue;
        smallImgROI = smallImg(*r);
        nestedCascade.detectMultiScale( smallImgROI, nestedObjects,
            1.1, 2, 0
            //|CV_HAAR_FIND_BIGGEST_OBJECT
            //|CV_HAAR_DO_ROUGH_SEARCH
            //|CV_HAAR_DO_CANNY_PRUNING
            |CV_HAAR_SCALE_IMAGE
            ,
            Size(30, 30) );
        for( vector<Rect>::const_iterator nr = nestedObjects.begin(); nr !=
nestedObjects.end(); nr++ )
        {
            center.x = cvRound((r->x + nr->x + nr->width*0.5)*scale);
            center.y = cvRound((r->y + nr->y + nr->height*0.5)*scale);
            radius = cvRound((nr->width + nr->height)*0.25*scale);
            circle( img, center, radius, color, 3, 8, 0 );
        }
    }
    cv::imshow( "result", img );
}
```

APPENDIX N

Source Codes, Datasheet and Demo Videos (Inside the Compact Disc)

CURRICULUM VITAE

GIL MICHAEL ECHAVEZ REGALADO

Address: 0008, Pk 15, Zn 6, Fuentes, Maria Cristina,
Iligan City, 9200

Email Address: bangonkali@gmail.com

Mobile Number: +639157764387

Date of Birth: September 16, 1992

Place of Birth: Bacayo, Iligan City

Civil Status: Single

Religious Affiliation: Roman Catholic

Father's Name: Gil Tan Regalado
Mother's Name: Victoria Echavez Regalado

Education

- MSU – IIT (2014)
- PSHS – CMC (2009)
- LSA (2005)

Work Experience

- Intern Test Engineer at Emerson Process Management – Fisher Rosemount Systems Incorporated Philippines. (April-June 2013)

Affiliations/Positions

- Open Source Software Contributor at Github (https://github.com/bangonkali/)
- Author at CodeProject (http://www.codeproject.com/Members/bangonkali)
- Junior Institute of Electronics and Communications Engineers of the Philippines Member (2009-2014)