Face detection and image storage on public transport front door with an FPGA based prototype system

By

Carlos García Lucero

Thesis submitted in partial fulfillment of the degree of

Master of Science

Department of Electronics
National Institute of Astrophysics, Optics and Electronics
(INAOE)
Tonantzintla, Puebla, México

Advisor:

Dr. Jorge Francisco Martínez Carballido

INAOE

Abstract

Assault and robbery on the public transport is a society problem. Almost every day, people lose their possessions and, in some cases, thieves or citizens get hurt or even die, in the process. In Mexico, attempts to reduce or prevent crime include vigilance by armed agents [1] and video cameras on the public transport [2].

The present thesis project objective is to contribute to crime reduction and/or prevention on public transport by having photographic records of people as they get on the bus, using a camera based system. These can be used later when an assault takes place to identify thieve(s).

An FPGA is used as the processing device which contains all the necessary logic to obtain the images from the camera, process them to locate the face, extract face region and store them into a Secure Digital (SD) media using an uncompressed BMP file, which could be read by any device which supports it without any special software.

To locate face area we used single channel image to extract regional attributes and their inter-regional relationship, resulting in a face detection algorithm with tolerance to skin tones and light conditions.

Resumen

Asaltos y crímenes en el transporte público es un problema en la sociedad. Casi todos los días, personas pierden sus propiedades y, en algunos casos, asaltantes o víctimas resultan heridos o incluso mueren en el acto. En México, los intentos para disminuir o contrarrestar este problema van desde el monitoreo de agentes armados [1] o la colocación de cámaras de video en el transporte [2].

El objetivo del proyecto de tesis que se presenta en este reporte, es ayudar a disminuir y/o prevenir el asalto al transporte público mediante la obtención de un registro fotográfico de las personas que abordan el autobús. De tal forma, que en caso de asalto, éste pueda ser utilizado para identificar al (los) asaltantes.

Se utiliza un FPGA como procesador que contendrá toda la lógica necesaria para comunicarse con la cámara para obtener las fotografías, procesarlas para obtener las coordenadas del rostro y almacenarlas en una memoria del tipo Secure Digital (SD) utilizando un formato de imagen BMP, el cual puede ser leído por cualquier dispositivo que soporte tarjetas SD y formato BMP sin necesidad de utilizar un software especial.

Para localizar las áreas del rostro, se utilizó un solo canal en las imágenes, extrayendo sus características regionales y sus interrelaciones. La estrategia anterior dio como resultado un algoritmo que tolera diferentes tonos de piel y diferentes iluminaciones.

Acknowledgments

To my family, thanks for your unconditional support these years. Thanks mother, for your loving and encouraging words and thanks to you father for your patient and for believing on me.

Thanks to my adviser, Dr. Carballido, who is also a good friend and from whom I have learn so much.

Thanks to the people who’s contribution was vital for the development of this thesis project. Thanks for letting us use their pictures and for lending us the bus (thanks father).

Thanks to you Nely, for hearing my INAOE’s stories during these two and a half years and for supporting my professional development.

Finally, Thanks to the CONACyT for the economical support.

Table of Contents

Figures' List

Tables’ List

Chapter 1 Introduction

1.1 Motivation

The use of technology as a tool for preventing and solving crime is a natural response to the decreasing security. Also, faster and smaller integrated circuits (ICs) which can be used to process ever increasing image resolution and higher processing capability of information are now available at lower cost. Increasing demand of high resolution cameras of cellular phones, personal devices and laptops, lead to economies of scale and decrease of prices.

Technology to prevent and decrease assault and robbery on public transport, and public places, is being used around the world. For instance, United Kingdom government has invested billions of pounds on video cameras to be used to monitor the security by police [3]. The initiative included [3]:

- Image database to track and identify offenders.
- Publish on internet images of robbery cases.
- Building a national CCTV database, incorporating pictures of convicted offenders as well as unidentified suspects.

Although a good strategy to reduce crime was planed, only 3% of assaults were solved by the use of CCTV technology [3]. Problem consists on the people monitoring the cameras because they had little training, and watching video for hours is not an effective task. Also when something occurs, search for the exact moment of occurrence is a hard work.

After training and new strategies, CCTV technology was helpful in solving 15-20% of street robberies [3].

In Mexico, the use of technology to prevent crime is very limited and little information can be found. In June, 2009, the State of Puebla Congress' security commission approved the use of camera installation on public transport [2]. This preventing technology is on its experimental phase, and Congress expressed its concern about the high cost of this technology for transport owners. The article didn't talk about the people who will be monitoring the video and the training they'll have to deal with this task.

1.2 Problem Description

At this time, technology is in use to solve or prevent crime and video monitoring seems to be the most used one.

Having real time video from transport to observe the events as they take place seems like a good strategy, because police could respond immediately if an assault is happening. But when talking about hundreds of transport units that could have the same technology, meaning that hundreds of videos are being received for their inspection in real time, the monitoring task begins to be difficult, then perhaps their option is to record the video on some kind of storing media for later review/inspection which of course means police could not respond as the assault is taking place.

As other countries had experienced, people monitoring videos must be trained for this task, so assigning a police to monitor videos generally gives mixed results, but purposely trained police agents guarantees to have an stable performance.

This thesis approach is to use digital cameras as a media to obtain still images (frames) from people faces to store these into a storage media without human intervention during the process. When an assault takes place, the face image of the thief has already been stored on the media, which also prevents the thief from erasing its picture by destroying the camera, for its later identification. Also a massive publishing of this picture could prevent the thief from assaulting again, and give the police a good starting point to locate and process this delinquent. By no using any human intervention during the process, the system reduces its operating cost due to the use of less equipment and any kind of monitoring training is no longer needed.

1.3 Objectives

System's general objectives:

- Develop an image processing algorithm fast enough to be able to process people images in order to locate faces and save them on the SD media.

1.1When people get on the bus, they take between 5 and 10 seconds to walk from the first step to the bus driver.

Chapter 2 Develop an integrated system with: camera, storage media (SD) and the locating faces algorithm to an FPGA developing board, taking into account its resources and limitations.

1.4 Solution Approach

Figure 1-1 shows the system approach block diagram. At first, the FPGA will initiate all the peripherals for their correct operation.

Among others, Camera module will be configured to work using the RGB555, the PSRAM will be configured to work on its burst mode and information will be received from the SD memory to obtain its size and FAT directories and data addresses. Details on these configurations will be explained latter.

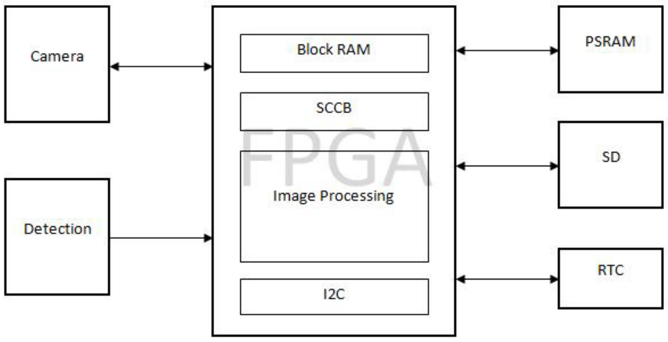


Figure 1-1 Proposed system's block diagram

After initiated, a frame will be obtained by the FPGA from the camera by a flag from the people Detection block. As the image is received the FPGA begins to store the image on the PSRAM, also, the Image Processing unit begins to process the image in order to locate the coordinates corresponding to the individual face. Once the image has been processed and completely stored on the PSRAM, the FPGA begins the BMP file saving process, recovering information from the FAT file system and directories by reading the SD. RTC module provides date and time information for file data, then will store the image data on the SD using the coordinates provided by the Image Processing unit where the face is located.

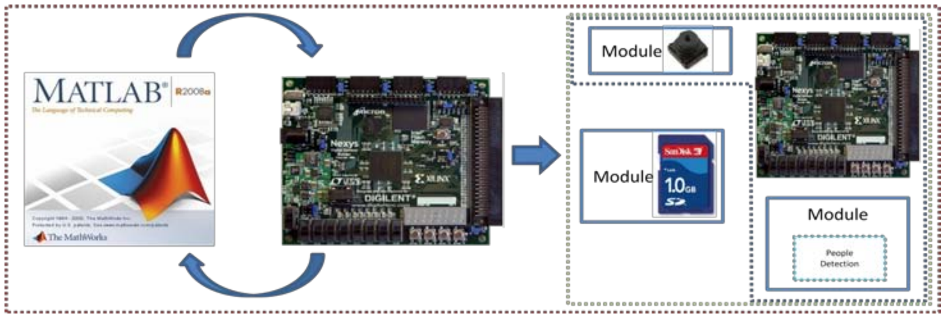


Figure 1-2 MATLAB←→VHDL working phase

Project was developed on three different stages:

- The first stage consisted on developing all the necessary logic for the FPGA to communicate with the external peripherals. This included all communication standards, intermediate buffers and printed circuit boards.
- On the second stage, the processing algorithm design using Matlab to develop the algorithm taking advantage of its mathematical functions; working on real people face images, taken with the camera system and downloaded images from internet sites. As the algorithm was developed and tested, equivalent VHDL code blocks were developed (see Figure 1-2).
- Finally on stage three, the algorithm was finally implemented on the FPGA and tests on public transport were made to ensure the algorithm was properly working, fixing any errors, until the system was working as expected.

1.5 Justification

Although transmitting video is perhaps the most commonly used technology for solving and preventing crime on the public transport, its transmitting and receiving infrastructure makes it very expensive to install and operate. Furthermore without a correct plan or strategy it is not necessarily useful [3]. By taking pictures of people who use the public transport for their later inspection, this thesis could help to decrease and solve crime, like transmitting and monitoring video; but, taking out all possible human monitoring error and making it less expensive. Although no possible real time action to coordinate police action is considered.

Because there is no monitoring during the process and later inspection would only take place after an assault has happened and only by police, the human error no longer exists. And because there is no transmitting technology, no use of air space or internet, and no use of receiving equipment or monitoring people, the costs are significantly reduced.

As people faces will be taken on BMP picture format, transferring the image to transport owners has no difficulty and thieves’ faces could be printed and placed on several buses. This could prevent them (thieves) from assaulting again, improving the quality of the public transport services and, hence, making it better.

1.6 Summary

On this chapter, the security problem on public transport and how this thesis project would contribute to improve it were discussed, including information about how some countries, including Mexico, are trying to solve this social problem.

Problem description, objectives and solution approach were described.

Next chapter will review all devices used by the system and how the FPGA system communicates with them. The description sequence is as they were developed for the system.

Chapter 2 Peripheral Design

2.1 Introduction

This chapter concentrates on the peripherals used by the developed system:

- CAMERA.
- PSRAM.
- SD memory.
- Real Time Clock.

Devices behavior and how the FPGA communicates with them will be explained on the following sections.

Development began with two FPGA boards: Digilent's Nexys and a Celoxica's RC10. The Digilent's Nexys has a PSRAM built-in and the Celoxica's board has the camera module connector built-in. As the camera printed circuit board (PCB) would be easier to develop and manufacture, the final objective was to use the Nexys board alone. However, at the beginning, the Celoxica's RC10 was used to wire the Nexys board to the camera.

After the devices were fully functional on these two wired boards, the necessary PCBs were developed and manufactured, in order to only use the Digilent's Nexys board alone.

Public transport results shown on chapter 4 were all taken with the Digilent's Nexys board alone.

2.2 Camera

The camera used to capture people images is a 1.3 Megapixel, 1024x1280 pixels, camera from Omnivision, model OV9650. Its functional block diagram is shown on Figure 2-1.

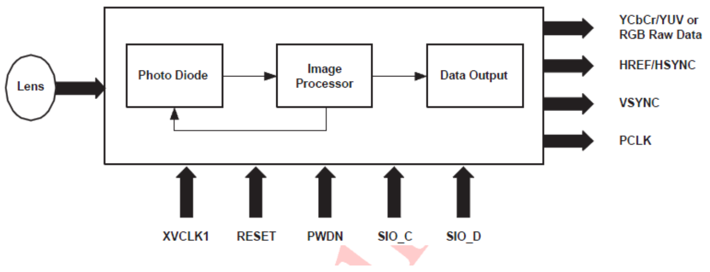


Figure 2-1 Camera functional block diagram [4]

It gives data using several formats including the RGB555, which is the one used on this work, YUV, YCbCr, etc. and it is capable of transmitting up to 15 frames per second when using a 48 MHz clock at full resolution [4]. To select which format to use or how many frames to be sent, it must be configured using special function registers, which also configure gain, brightness and many other built-in functions, for further reference see [4].

The camera chip uses the Serial Camera Control Bus (SCCB) communication standard to configure its registers. It sends and receives data using two lines: the bidirectional data line and the unidirectional clock line. Figure 2-2 shows SCCB transmission protocol.

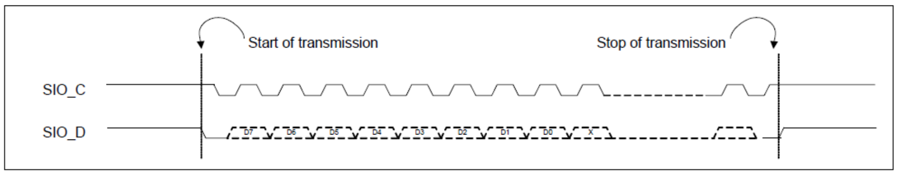


Figure 2-2 SCCB data transmission scheme [4]

All registers are accessed this way and once the registers are written, the camera takes up to 10 frames for changes having effect. After configured, the read process can take place.

A 10 bit wide bus is used to send picture data from the camera to the host (FPGA) but, for the RGB555 mode only the 8 most significant bits are used.

Two bytes are needed for every pixel containing the three color planes: one byte sends the red and green's two most significant bits, the second byte sends the rest of green and blue, as shown on Figure 2-3.

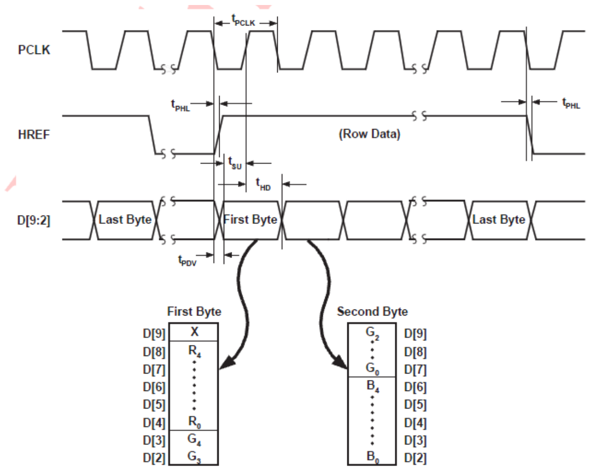


Figure 2-3 RGB555 pixel Color format transmission [4]

Figure 2-3 also shows the camera PCLK clock and the HREF signal. The PCLK clock is sent by the camera to the host, but it only exists as long as the host sends its own clock to the camera. The HREF is high for every row, meaning 1024 times HREF being high, and, since two bytes are needed for every pixel, then 2560 bytes are sent while HREF is high, a single row at maximum resolution.

The VSYNC signal says when the camera starts and ends a frame data, as shown on Figure 2-4.

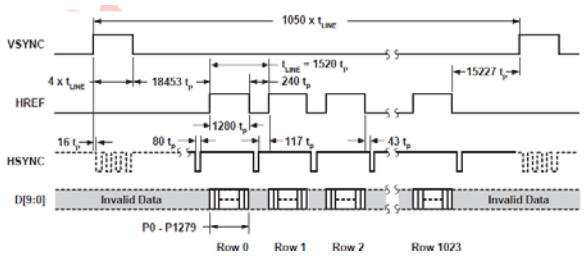


Figure 2-4 Synchronizing signals VSYNC and HREF [4]

PCLK, HREF and VSYNC are unidirectional signals from the camera to the host.

The FPGA uses the HREF and the VSYNC data signals in order to receive a complete frame, all synchronized with the PCLK clock.

Figure 2-5 is a block diagram representation of the FPGA communicating with the camera and the peripherals involved within the process.

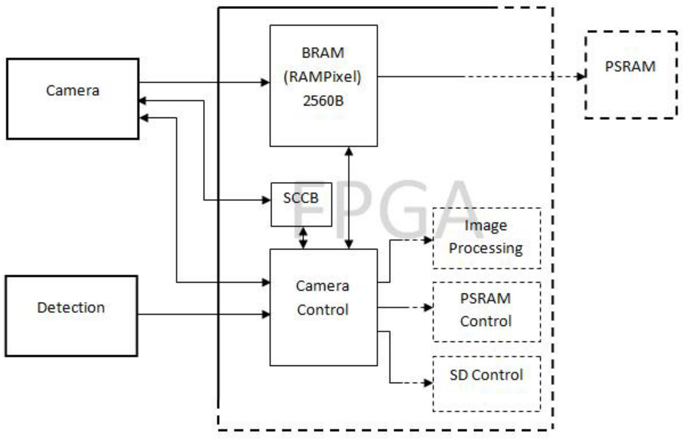


Figure 2-5 Camera-FPGA block diagram scheme

At the beginning, the camera is configured accordingly the project needs and criteria, configuring among others:

- Output format
- Gain
- Exposure time and
- White balance

After configuration, a new frame can be detected with the following sequence, VSYNC is asserted, the FPGA waits for the HREF signal to start and when the HREF signal is set to ‘1’, the FPGA begins to fill a 2560 bytes buffer, called RAMPixel, with row data. RAMPixel is configured to use Block RAM, which is RAM memory built on the FPGA. This buffer stores the complete row and passes the data to the PSRAM immediately after complete reception. The need for this buffer is due to the way the PSRAM works: it needs to refresh its data for a fixed period of time, every time, allowing it to write or read data only for fixed quantities of bytes. Details on this will be explained when reviewing the PSRAM.

Once the camera has sent a frame, meaning data stored on the PSRAM and already processed, the recording process on the SD memory begins.

2.3 PSRAM

The Micron’s PSRAM is an 8 Mega x 16 bytes high speed Cellular RAM which contains a 128 Mbit DRAM core [5]. It has a hidden refresh data mechanism which needs only to be configured for it to operate, then, there’s no need for the host (FPGA) to call it. Figure 2-6 shows its block diagram.

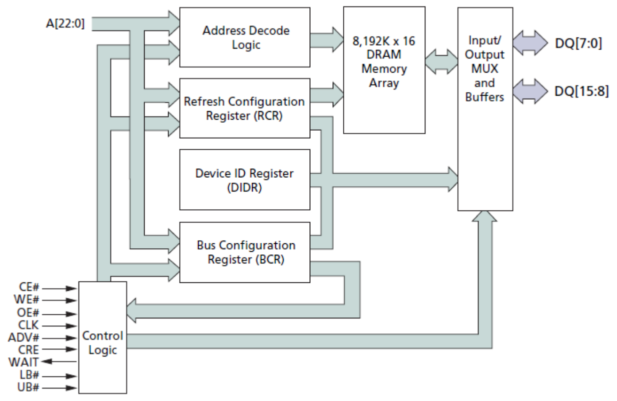

A[22:0]
Address Decode Logic
8,192K x 16 DRAM Memory Array
Input/ Output MUX and Buffers
DQ[7:0]
DQ[15:8]
Refresh Configuration Register (RCR)
Device ID Register (DIDR)
Bus Configuration Register (BCR)
CE#
WE#
OE#
CLK
ADV#
CRE
WAIT
LB#
UB#
Control Logic

Figure 2-6 PSRAM Functional block diagram [5]

PSRAM is used to store all frame data. It’s needed because the SD memory can only write data for fixed quantities and the recording time is not always the same, so the storing data must be placed into another storing device temporally.

It has three different writing and reading modes: page, burst and asynchronous. On the page mode, it writes and reads any quantities of data without a synchronizing clock as long as this quantity does not exceed the fixed time required for the PSRAM to refreshing its data. Burst mode writes and receives fixed quantities of data with a single given address and using a synchronizing clock, this mode is the one used on this work. The asynchronous mode receives and sends one byte data within a single address and without a synchronizing clock, this is the slowest mode. Figure 2-7 shows the burst write process.

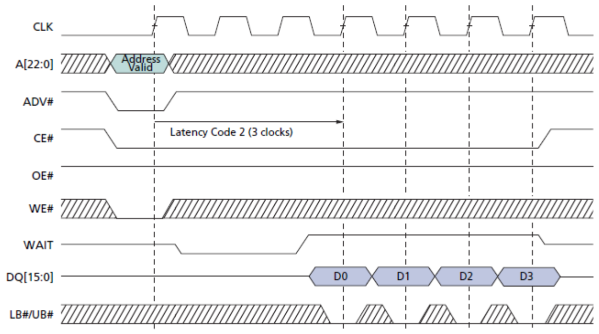


Figure 2-7 PSRAM burst mode writing process [5]

User accessible registers configure the different PSRAM operations burst, page, async, refreshing time, the burst length, etc. The host must access these registers in order to configure the desired operation mode of the PSRAM.

Figure 2-8 shows a block diagram of the FPGA communicating with the PSRAM. On its initial configuration the refresh time is set, the burst mode and the burst length as well, among others.

As said before, the PSRAM needs to refresh its data; on a burst mode operation the PSRAM can write and read data packages of a maximum of 128 bytes, and, as the camera sends the frame data continuously, this data must be stored on the RAMPixel buffer before stored on the PSRAM.

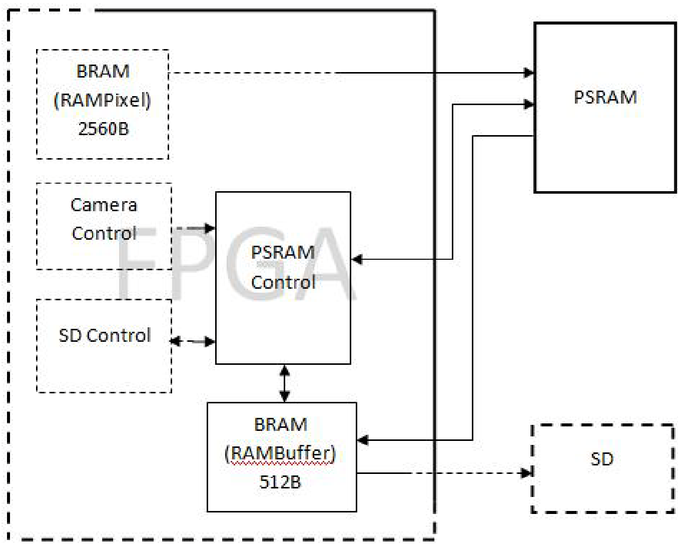


Figure 2-8 PSRAM-FPGA block diagram scheme

Camera control gives to the PSRAM control when to start writing a frame data then SD control gives when to start reading the frame data to be stored on the SD memory. For the data being read from the PSRAM and stored on the SD memory, an in-between buffer is used, called RAMBuffer. The need of this buffer is to compensate size and time differences between devices. The 512 bytes size of the buffer is the SD memory's sectors size.

2.4 SD Memory

The SD memory card has several aspects that must be detailed:

- Physical layer
- FAT system
- FPGA – SD

Communication protocol and commands for its configuration are explained on the physical layer. How an operating system recognizes a file is explained on the FAT system section.

2.4.1 Physical layer

2.4.1.1 Communication Protocol

SD card can be configured to work using two different modes:

- SPI mode. As its name indicates, uses a SPI standard communication protocol making it easier to use but slower. Communication protocol is called SPI Bus mode.
- SD card mode. It uses its own communication protocol with the advantage that data can be transferred using four data lines, making it faster but difficult to use. On this mode, the communication protocol is called SD Bus mode.

Due to the faster transfer data rate that can be achieved by the SD card mode, this mode was implemented on the FPFGA for the thesis project. Details on this protocol will be explained next and configuration commands will be explained on another section.

Communication over the SD bus is based on command and data bit streams, which are initiated by a start bit and terminated, by a stop bit [6]:

- Command. A command is a token that starts an operation. A command is sent from the host either to a single card (addressed command) or to all connected cards (broadcast command). A command is transferred serially on the CMD line.
- Response. A response is a token that is sent from an addressed card, or synchronously from all connected cards, to the host as an answer to a previously received command. A response is transferred serially on the CMD line.
- Data. Data can be transferred from the card to the host or vice versa. Data is transferred via the data lines.

Card addressing is implemented using a session address that is assigned to the card during the initialization phase. The basic transaction on the SD bus is the command/response transaction (see Figure 2-9). This type of bus transaction transfers their information directly within the command or response structure. In addition, some operations have a data token [6].

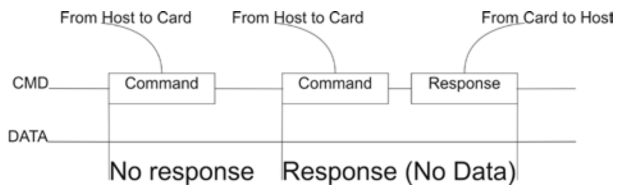


Figure 2-9 SD's basic transaction process scheme

Data transfers to/from the SD Card are done in blocks. Data blocks are always followed by CRC bits. Single and multiple block operations are defined. Note that the Multiple Block operation mode is better for faster write operation. A multiple block transmission is terminated when a stop command follows on the CMD line. Data transfer can be configured by the host to use single or multiple data lines, as long as the card supports this feature [6].

The block write operation uses a simple busy signaling of the write operation duration on the DAT0 data line (see Figure 2-10) regardless of the number of data lines used for transferring the data [6].

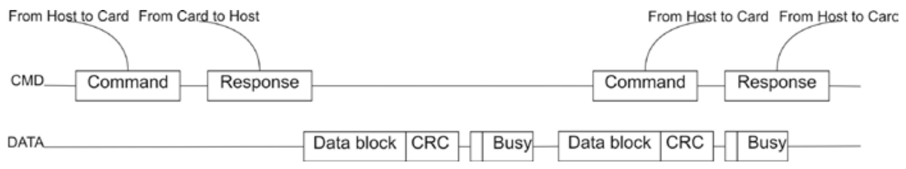


Figure 2-10 SD memory writing process scheme

Write and read commands structure is shown on Figure 2-11:

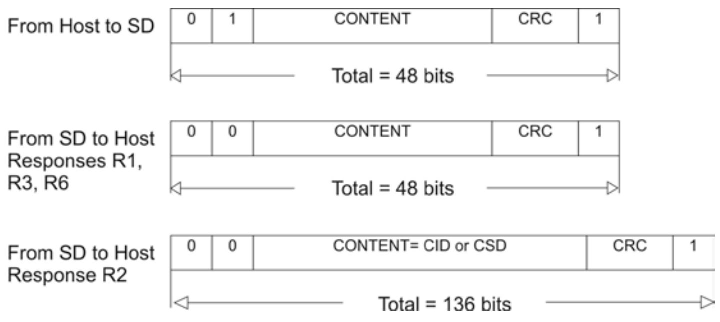


Figure 2-11 Data write and read packages scheme on SD memory

On this scheme, the content from Host to SD sends command's addresses and from SD to Host receives the command response data. Content is transferred MSB to LSB.

The SD wide bus option is used for transferring image data. This is the fastest mode available for the SD. Figure 2-12 shows the SD wide bus transferring scheme.

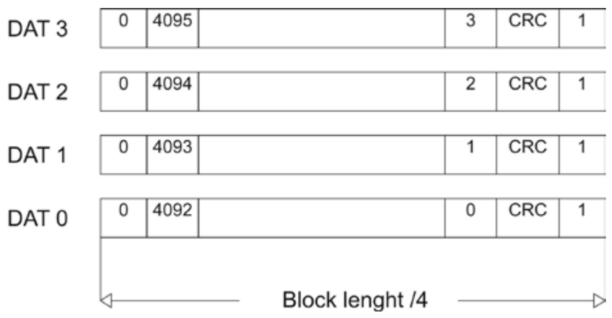


Figure 2-12 SD wide bus data scheme

2.4.1.2 Commands

The SD supports several commands for writing and reading data, or configuring its behavior. Only commands used by this thesis project will be explained. Table 2-1 shows the commands used.

Command	**Function**
CMD 0	Software reset. Goes to idle state. No response from the SD.
CMD 55	APP_CMD. Precedes every Application Specific command.
CMD 2	Ask for the Card Identification (CID) number.
CMD 3	Ask for the Relative Card Address (RCA).
CMD 9	Ask for the Card Specific Data (CSD)
CMD 7	Toggle between Stand-by and Transfer states.
CMD 17	Reads a block of data.
CMD 13	Ask for the Status Register.
CMD 24	Writes a block of data.
CMD25	Writes blocks of data continuously.
CMD 12	Terminates a multiple write blocks.
ACMD 6	Defines data bus width: 1 or 4 bit bus.
ACMD 41	Ask for the Operating Condition Register (OCR).

Table 2-1 SD commands used by the thesis project

CID number is not used; however, the initialization process of the SD requires this command. It has information written during the manufacturing process and cannot be changed.

RCA register gives a 16-bit card address used by the host to communicate with the SD.

CSD is a 136-bit register which gives information about the SD card physical behavior: maximum block length, device size, data transfer speed and data transfer rate, etc.

Status Register gives the CURRENT_STAT status bit of the SD card. It’s used to check when data writing has finished on the SD.

The OCR gives information about the voltage range in which card data can be accessed. Also, gives a card power up status bit which informs the Host that the power up procedure is done and access can continue.

Detailed information on this registers can be found on [6].

2.5 FAT System

When people buys a SD card, it has preloaded a FAT file system for personal computers, most cell phones, digital cameras, and other common devices recognize it and use it to store any kind of data files. Without a FAT file system, the card can be used to write and read data, but this can only be read or written by using special software which would require extra design and add extra costs. This thesis project uses the FAT system so any device which supports it could read the images stored and no special software is needed.

SD FAT file system organizes the card in: User data, Root Directories, File Allocation Table, Partition Boot Sector (PBS) and a Master Boot Record (MBR).

MBR is located on the first sector (512 bytes length) and its content is shown on Table 2-2.

Byte Position(BP)	**Field Name**	**Contents**
0	Master Boot Record	Not Restricted
446	Partition Table	Refer to Table 2-3
462	Partition Table	All 0x00
478	Partition Table	All 0x00
494	Partition Table	All 0x00
510	Signature Word	0x55, 0xaa

Table 2-2 Master Boot Record used by FAT within the SD memory

The Relative Sector informs the total amount of sectors before starting the PBS and it's the first one read. Extensive explanation on contents can be found on [7].

BP	Field Name	Contents
0	Boot Indicator	0x00 or 0x80
1	Starting Head	Numeric Value
2	Starting Sector/Starting Cylinder	Numeric Value
4	System ID	0x01 or 0x04 or 0x06
5	Ending Head	Numeric Value
6	Ending Sector/Ending Cylinder	Numeric Value
8	Relative Sector	Numeric Value
12	Total Sector	Numeric Value

Table 2-3 Master's Boot Record Partition Table

The PBS data is shown on Table 2-4.

BP	Field Name	Contents
0	Jump Command	Bytes
3	Creating System Identifier	a-character
11	Sector Size	Numeric Value
13	Sectors per Cluster	Numeric Value
14	Reserved Sector Count	Numeric Value
16	Number of FATs	Numeric Value
17	Number of Root-directory entries	Numeric Value
19	Total Sector	Numeric Value

21	Medium Identifier	0xf8
22	Sectors per FAT	Numeric Value
24	Sectors per Track	Numeric Value
26	Number of Sides	Numeric Value
28	Number of Hidden Sectors	Numeric Value
32	Total Sectors	Numeric Value
36	Physical Disk Number	0x80
37	Reserved	0x00
38	Extended Boot Record Signature	0x29
39	Volume ID Number	Numeric Value
43	Volume Label	d-characters
54	File System Type	d-characters
62	(Reserved of system use)	Not Restricted
510	Signature Word	0x55, 0xaa

Table 2-4 Partition Boot Sector used by FAT within SD memory

For SD FAT system specification is always 512. Sectors per FAT says how many sectors are reserved for the FAT and, for the SD FAT system, there are always to FATs and 512 Root-directory entries. On the next sector, after the PBS, the first FAT starts.

The File Allocation Table supports both the 12-bit FAT and the 16-bit FAT. The FAT structure is compliant to ISO/IEC 9293. The FAT type shall be determined by the number of clusters that depends on the parameter from the physical layer. If the cluster number is less than 4085, FAT12 shall be used. Otherwise, FAT16 shall be used. The first byte of the FAT shall specify the format identifier and be recorded 0xf8. In case of FAT12, the byte 2 and 3 shall be recorded as 0xff each. In case of FAT16, the byte 2, 3 and 4 shall be recorded as 0xff each. The sectors of the FAT may include unused area, because the number of clusters shall determine the FAT size in bytes. This unused area shall be recorded as ZEROs [7]. Both FATs must be recorded using the same data.

A Directory is a descriptor that shall contain a set of Directory entries each of which identifies a file, a Volume Label, another Directory or is unused. A Directory can contain the 65536 Directory entries. The format of the file name for the Directory entries should use 8.3 name file format. Although the Long File Name (LFN) can exist in the Directory entries, the SD Memory Card file system may ignore these entries, and refers only to the file name using 8.3 name file format which is stored with the LFN. The character code in the Directory entry can be used the code which is permitted by the ISO/IEC 293 [7].

The User Area shall be organized into clusters. Each cluster has its own Cluster Number. The first cluster in the User Area is Cluster Number 2. Other than that, there are no special restrictions for the SD Memory Card file system [7].

2.6 FPGA-SD

FPGA internal blocks which were developed to communicate with the SD card are shown on Figure 2-13. When powering up, the SD is on its card identification mode and it must be initiated using the sequence shown by the flowchart shown on Figure 2-14.

Command 0 is needed in order to reset the SD card and put it on an idle state. No response from the SD is send to the FPGA. A command 55 is next send to be able to send the application specific command 41. This command initiates the powering up process and it won’t continue until a powering up (ready) bit is asserted. Once asserted, the FPGA sends command 2; as said before, no data is reviewed from this command but it’s needed in order to initiate the SD. Finally, command 3 is used. This command gives the RCA which will be stored and used by the FPGA to communicate during the read and write processes. With command 3, the SD changes from Identification mode to data transfer mode, but data cannot be transferred until command 7 is sent. Command 7 selects which card can transfer data by using the RCA on its argument, meaning that the card bound to the transferred RCA is selected to write and read data.

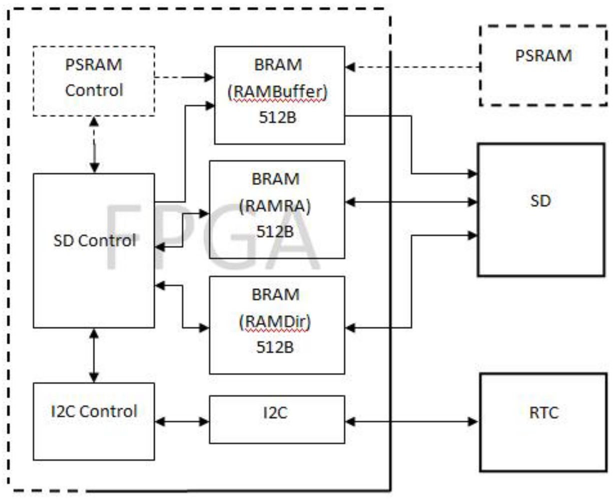


Figure 2-13 SD-FPGA block diagram scheme

During initiation process, no image data is read or written from the SD to the FPGA or vice versa so no buffers are needed.

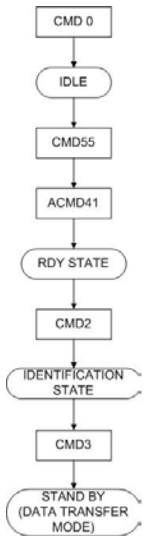


Figure 2-14 SD Card initiation flowchart

After initiated, the card reads information to retrieve the location of the FATs, directories and user area. Figure 2-15 shows flowchart of this process.

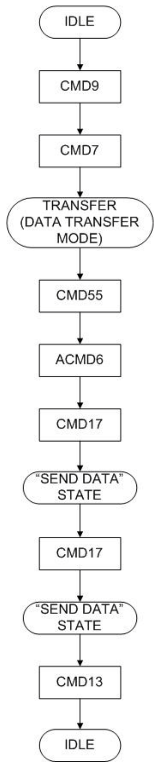


Figure 2-15 FAT, directories and user area identification flowchart

Although command 9 is sent, no data from the CSD register is used by the FPGA. Next, command 7 is sent which selects the SD by the use of the RCA. After this, the card is ready to send and receive FAT data.

Bus width is set to 1 bit first, using command 55 and application command 6, then command 17 is used to retrieve data from the MBR, which is stored on the RAMRA block buffer. After storing, the FPGA can locate the PBS and the total size of the SD card. The second command 17 is used to retrieve data from the PBS which is stored on the RAMRA block buffer as well. From the PBS the location of the FATs, the directories and the user data area can be accessed. Command 13 gives the SD card status: if it's idle or busy.

Once initiated, after powering up the system and reading the MBR and PBS, the initiation process and the reading of these sectors won't be repeated unless the SD card is taken out from the SD adaptor and inserted back. This means that system supports hot insertion.

The PSRAM control generates signals to the SD control when a new BMP file is about to be stored. Figure 2-16 shows the flowchart of a writing BMP file process.

When a new BMP file is about to be stored, the directories are first read using 1 bit transfer mode. Directories are stored on the buffer RAMDir and it is needed because one directory is written side by side within a sector. Next, the data bus width is change to 4 bits transfer mode. The FATs are stored and the directory as well. Finally the BMP file data is stored beginning with the BMP file header and finally the BMP file data retrieved from the RAMBuffer.

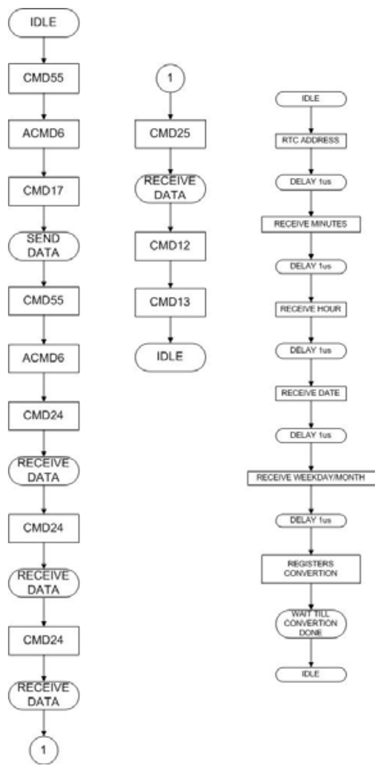


Figure 2-16 BMP file writing flowchart including RTC communication flowchart

As the writing process begins, the RTC is also started, in order to receive the date. This is a parameter used by the BMP file header. The device which reads the SD memory retrieves the creation date of the file, and could be useful by police or transport security or owners to know the exact moment when the thief gets on the bus.

All write, or read, operations with an I2C device must start with its I2C device address (see Figure 2-17 for a data write example). For the RTC, this address can be found on the Philips datasheet [8]. Next, the data registers can be read. Figure 2-16 shows the flowchart of the RTC read process as well. Data read from the RTC is on a format not compatible with the FAT format, and cannot be used directly with the BMP file header, so it has to be converted by the FPGA in order to be useful. Until this converting process is finished, the

RTC cannot read anymore data and the FPGA would not continue to write data of the BMP file.

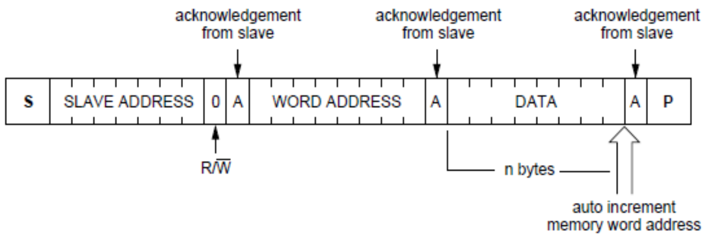

acknowledgement from slave
acknowledgement from slave
acknowledgement from slave
S
SLAVE ADDRESS
0
A
WORD ADDRESS
A
DATA
A
P
R/$\overline{W}$
n bytes
auto increment memory word address

Figure 2-17 I2C writing operation scheme [8]

2.7 Summary

All devices used by the final system prototype were reviewed: Camera, PSRAM, SD memory and RTC; along with their block diagram representation used to communicate with the FPGA system. Important definitions of the FAT system were described as well as the BMP file storage on the SD memory.

Next chapter will focus on face detection algorithm as it was developed on MATLAB and ends describing the developing process for the corresponding VHDL algorithm.

Chapter 3 MATLAB-VHDL Algorithm

3.1 Introduction

Face detection algorithm was developed using data obtained from several experiments on real pictures from internet sites and pictures taken with the camera-SD system developed and explained on chapter 2.

Taking advantage from the mathematical tools offered by MATLAB, the algorithm was at its first stage developed entirely on m-language. Next, as elements of the algorithm were stable and defined, the corresponding VHDL code was developed.

The focus on algorithm development was to find a simple face detection method. Simple meaning that it uses resources that allow implementation on a low cost FPGA, and being fast enough to deal with the time requirement imposed by the public transport users getting on the bus.

3.2 Background

Several works have been developed on the face detection field and many approaches have been considered.

At the beginning of this thesis project, several articles were found which were developed to work on personal computers [9], [10], [11] and only one which was developed on a reconfigurable platform [12].

Implementation of algorithms on personal computers are not concerned about mathematical resources as programming libraries are fully capable of working with complex mathematical functions, from the VHDL point of view. For instance; system on reference [9] claims to be “simple and accurate”, it performs lighting compensation, color space transformation, skin color detection, low pass filter, candidate regions, height to width ratio detection, eye detection, mouth detection and finally face detection. Authors can claim it to be simple because it was developed on a personal computer and mathematical tools are fully available. For an FPGA, even a simple mathematical operation as a division needs corresponding IP; either by buying or developing it, so a solution space is not quite simple.

Systems in [10] and [11] also use a space transformation from RGB to GLHS [10] (generalized LHS) and YCbCr [11] which require several mathematical operations using fractional data. For [10], the face detection process takes up to 15 seconds. Basic VHDL libraries do not support floating point operations and, as mentioned on chapter 1, people take between 5 to 10 seconds to get on the bus; so 15 seconds is not an option.

Reference [12] presents a solution on an FPGA Spartan 3 using a LEON2 processor which is a 32-bit SPARC V8 compliant processor. Authors used an eye detection approach for this article, which leads the system to be unable to find people faces if one eye is not found [12]. As [10], authors used a GLHS space which introduces the need of floating point operations which are implemented on the LEON2 processor are great FPGA's resource consumers when using a non processor based solution on VHDL using free or paid IPs.

For this thesis' system needs, no article was found to be useful as a foundation. The need for a fast processing algorithm simple enough to fit into a low cost FPGA lead to the development approach explained on next sections.

3.3 MATLAB Algorithm

3.3.1 Algorithm Basis

Research began applying histogram to a several RGB images to observe how colors are distributed on faces' surface or, more specific its face's skin.

Histograms are useful because they give information about the color levels found on the image and the quantity of these color levels distributed on the entire image, although they don't give information about their location. Here we apply regional histograms to images with people skin, so that by using the color levels of the skin face location can be obtained. During research, it was found that red color levels have higher values on skin regions than green and blue colors, so the project was limited to use only red color levels.

Figure 3-1 shows two images containing a person's face and their red, green and blue histograms respectively. Figure 3-1a shows an internet obtained image in 24-bit JPEG format and Figure 3-1b shows a RGB555, taken with our system on dark interior conditions.

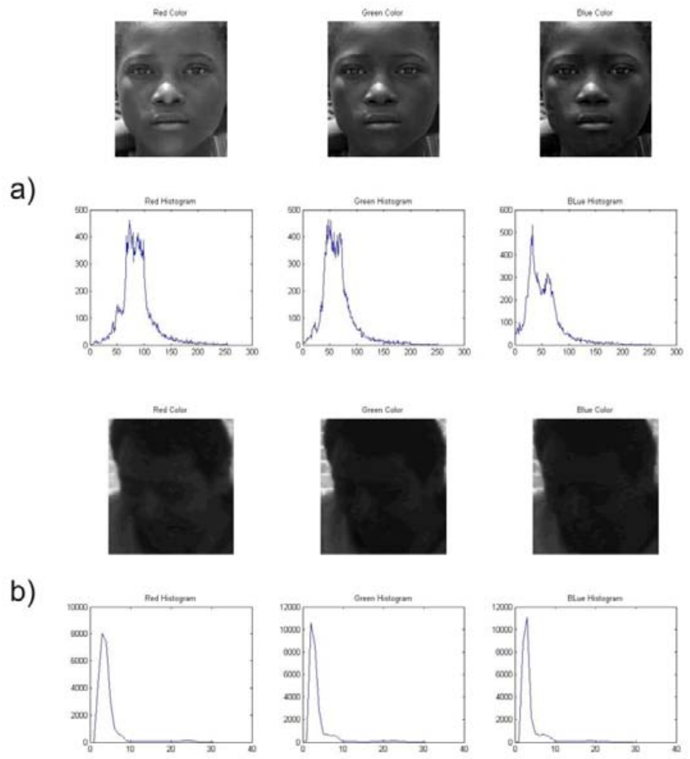


Figure 3-1 Histograms on a) 24-bit JPEG and b) RGB555

Although it’s easier to observe the higher red values on the JPEG just by looking at Figure 3-1a, this is less clear with the RGB555 image. Table 3-1 shows the first ten values of each color remarking the higher values to denote the same, higher red values, are happening with the RGB555 format. The Figure 3-1b face image was taken with the camera-SD system on the public transport.

Level	Red Value	Green Value	Blue Value
1	0	0	0
2	0	142	2
3	4318	**10549**	**8796**
4	**8060**	**8557**	**11106**
5	**7374**	2791	2199
6	3005	744	757
7	1004	647	610
8	641	619	653
9	445	622	585
10	119	288	323

Table 3-1 RGB555 histogram first ten values

When obtaining the histogram of the images, the color level range of the skin was found to be around 4 levels for the RGB555 format and about 40 to 50 levels for the 24-bit JPEG. Also, with these experimental observations it could be found that the quantity of any particular color level is not necessary but only its existence, which dramatically simplifies the algorithm. Here on we will use modified histogram to mean a representation for distribution of color existence in a given image region.

To apply this modified histogram to a picture, a rectangular region of a given size was used to sweep the entire image. Using a rectangle of a size proportional to the face wouldn't be a good strategy due to the different face sizes of people; for instance, a teenager and an adult have not the same face size. Instead, rectangles of a smaller size could be used, and sweeping the image's area as shown on Figure 3-2.

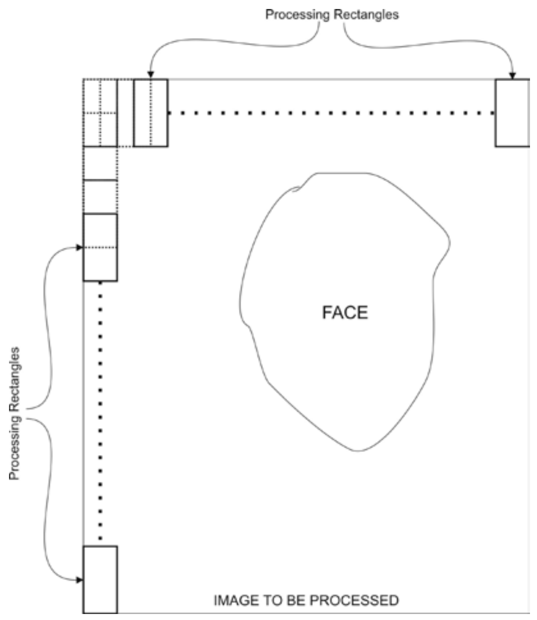


Figure 3-2 Horizontal and vertical sweeping representation

When using rectangles of small sizes, these have a higher probability to process a face portion which contains only skin than using larger regions which could result in processing image portions which also may contain hair or background; although skin region has around 4 colors, this doesn't ensure that the entire face will have only these four, and that could cause system failure by not locating the face coordinates. When small rectangles are used, they would process portions of the face containing hair, background, or face region with more than 4 colors, and these frames would not be considered, but, the overall processing rectangles which containing only the around 4 colors restriction would be higher, resulting on the face being located. Figure 3-3 shows two images processed using rectangles of size proportional to the face. The resulting location shows on the left image that locates the face; but on the right image, this size criteria fails to locate a face region on the image.

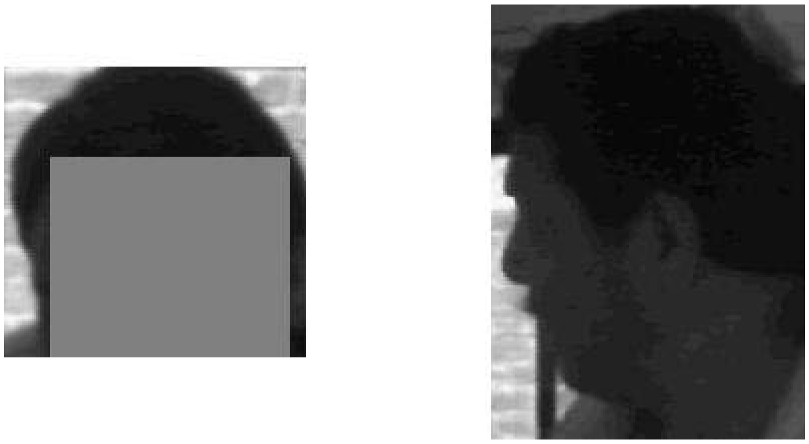


Figure 3-3 Images processed using rectangle size's proportional to face size

Figure 3-4 shows the same images processed using smaller rectangles, these results illustrates the usefulness of this strategy. The rectangle's size criteria consider improves detection quality and reduces FPGA's resources.

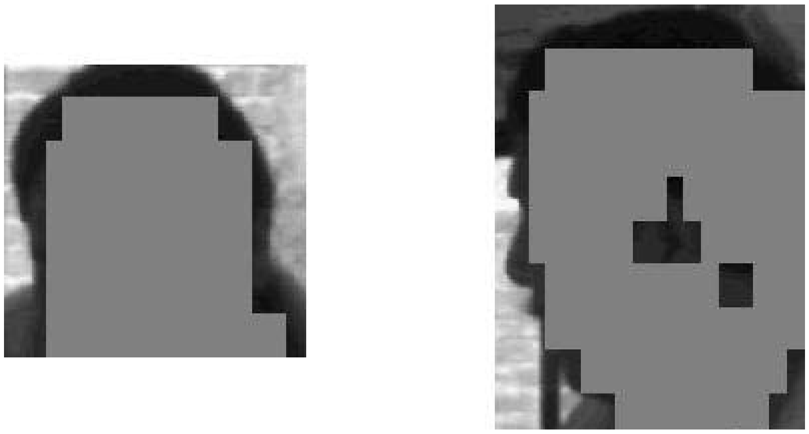


Figure 3-4 Images processed using rectangle size's smaller than face size

When observing people getting on the public transport, it's easy to notice that only an image area is useful. Only the portion of the image used by people getting on the bus is important and it can be defined by observing images samples taken using the camera-SD system or simply by the physical characteristics of bus entrance and camera location.

Figure 3-5 shows images used to determine the portion of the image which is useful. These images show people of different heights on different steps of the bus. They were holding a stick, marked every 10 cm, used to have a reference on the height of the image which was to be located. The width also was located using these same images, taking into account the bus door and the trajectory walked by people as they get on the bus and walk closer to the driver. Figure 3-6 shows, on these same images, the region defined to be useful for processing and recovering face region coordinates. It can be seen that two regions were found to be useful: one corresponds to the first step position (FSP) of the bus and the other corresponds to the second step position (SSP).

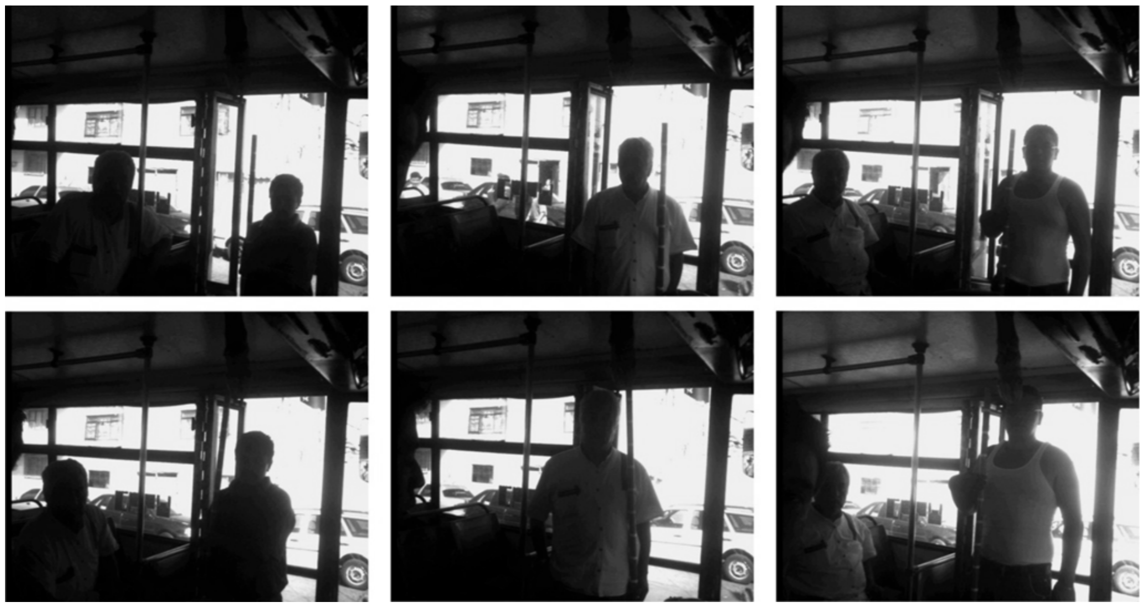


Figure 3-5 Images used to determine useful detection positions

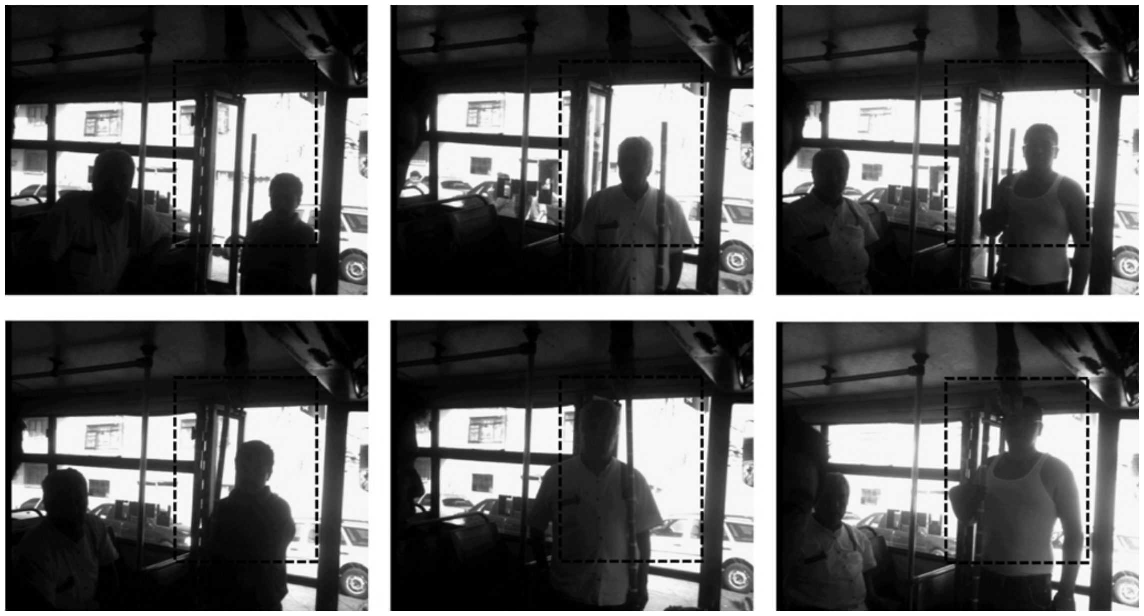


Figure 3-6 Useful detections positions (dashed rectangles)

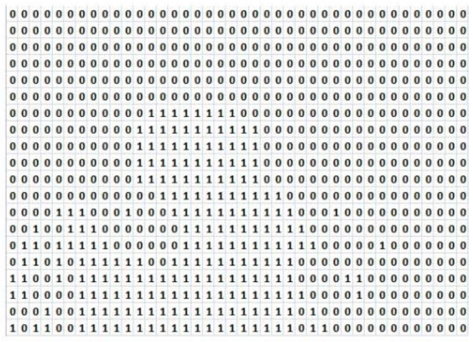
3.3.2 XOR and Clothing filters

Once the FSP and SSP of the image are established, the color levels rate was obtained and the geometrical size of the rectangles determined, then, the process unit filters the image from noise.

Processing rectangles find people skin, but, also can find random noise due to the background, clothing or from other sources. To filter this noise, an XOR operation was used on a morphological fashion: a structure element was used and those groups of rectangles which are similar to this structure element are conserved, those which are not, are discarded. For achieving this, the processed image for FSP, or SSP, is converted to a binary matrix being ‘0’ those rectangles which didn’t accomplished the skin criteria and being ‘1’ those which did. Figure 3-7a shows an FSP image is processed and Figure 3-7b shows the binary matrix which represents the same image. This image introduces a scale factor which depends on the sweeping process; for instance, if the sweep is made every 10 columns, the scale factor will be 1:10

and when retrieving the face locating coordinates, the result will be multiplied by 10.

a)



```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 0 0 0 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0
0 1 1 0 1 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0
1 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
```

b)

Figure 3-7 a) Processed image and b) Binary matrix representation

Once the binary matrix has been obtained, it is filtered by an XOR operation between the matrix and the structural element. Just like the rectangular region sweeps the useful portion of the image, the structural element will sweep the entire matrix in order to discard the bits which correspond to noise and leave only the image corresponding to people´s skin. Figure 3-8a has the same Figure 3-7b with the asserted bits highlighted and Figure 3-8b shows the result of the filter operation with the asserted bits highlighted as well (note that Figure 3-8b is smaller than Figure 3-8a which is ‘n’ rows by ‘m’ columns due to the height and width of the structural element).

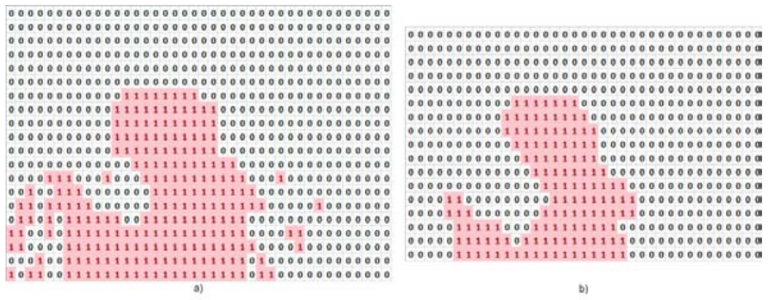
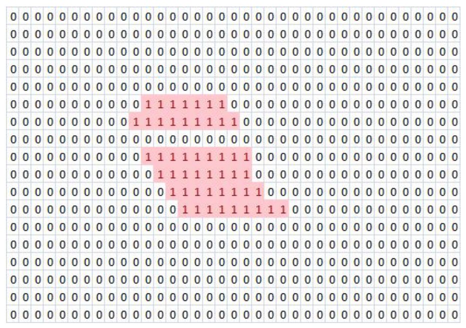


Figure 3-8 a) Binary matrix representation b) XOR filtered

As Figure 3-8 shows, not all noise regions are eliminated, specially the noise due to people clothing. This noise follows the subject's corporal structure, making it wider than the face and most of the time it will behave this way because no person has the face wider than his/her shoulders. To eliminate this, we use the fact the width of a face can be measured and a threshold higher than this width can be used to eliminate the clothing noise.



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3-9 Clothing filtered matrix used to obtain the final coordinates

Figure 3-9 shows the filtered image from Figure 3-8b using the clothes threshold. Now it can be seen that only the face is obtained.

3.3.3 Face Region Coordinates

To obtain the coordinates of the located image, the average of the binary matrix is used. The average is used due to its property in which its result value tends to be near the higher grouped values. This will cause its result to be near the face even when there is noise which wasn't possible to be filtered by the operations described on the last section.

The resulted binary matrix from the XOR and clothing filters is then changed to two different matrices each one with decimal values instead binary one. Figure 3-10 shows the two decimal value matrices as if they were filled with '1', Figure 3-10a shows the matrix for locating the 'x' coordinate and Figure 3-10b shows the corresponding one for the 'y' coordinate.

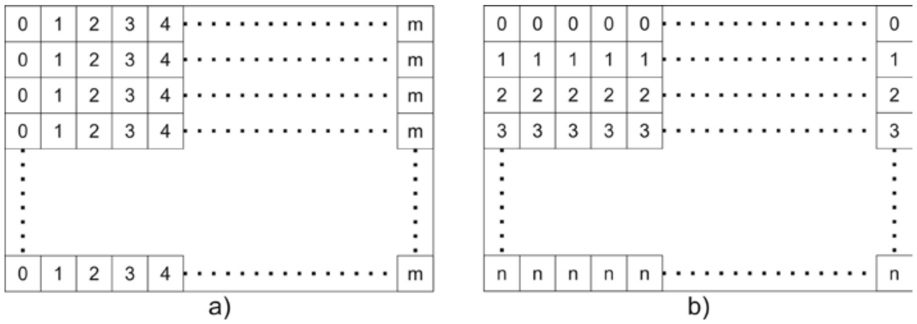


Figure 3-10 Decimal representation matrices for obtaining average on a) 'x' and b) 'y' coordinates

The coordinate obtained by the average, it's multiplied by the scale factor of the FSP, or SSP depending on the step being processed, and the processing rectangles and, finally, added the offset resulted from the absolute coordinates belonging to the useful region on the 1.3 megapixel image. Figure 3-11 shows the processed image resulting from locating the coordinates using the average of the matrix shown on Figure 3-9. Figure 3-11 is the final result of the image processing block.

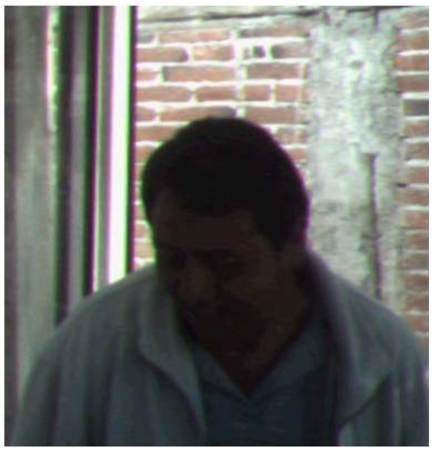


Figure 3-11 Face obtained from applying the entire algorithm.

Next the VHDL equivalent algorithm will be explained.

3.4 VHDL implementation

Figure 3-12 shows the FPGA inner blocks which were developed to obtain the face region coordinates from the MATLAB algorithm. The color detection block acquires the data from the processing rectangular regions for the FSP or SSP. Both, FSP and SSP, found with MATLAB are constant for all people processing so there’s no need of an additional block for this task.

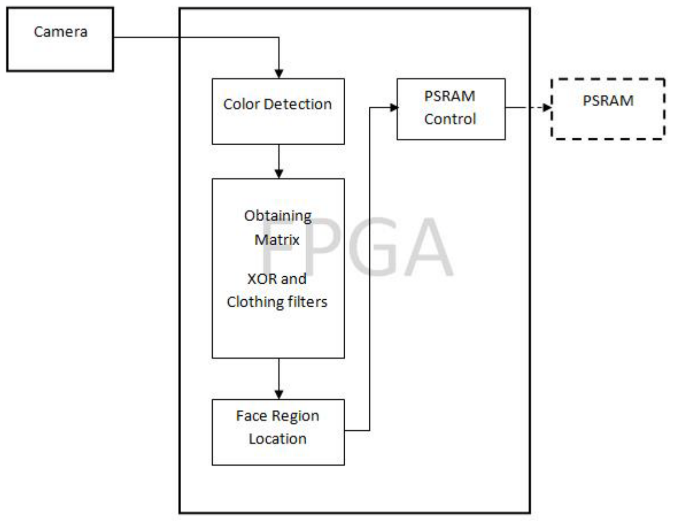


Figure 3-12 Image Locating Algorithm – FPGA block diagram scheme

On MATLAB there’s one rectangle which sweeps the entire image. For a 500x400 image (FSP), a 50x20 rectangular region, sweeping every 10 columns and 25 rows, must be repeated eight hundred times, 40 horizontally by 20 vertically.

Taking advantage from the inherent parallelism achieved by FPGAs there can be as many rectangles to fill the width of the useful portion, and loop would only be repeated for the height of this region; this means that the subsequent blocks processes the image per row. All these rectangles are being executed as the image is being received which makes the locating coordinates process faster, being capable of locating the face coordinates before the entire image (1024x1280 pixels) is received.

After the first ‘n’ or ‘m’ rows, depending on which of FSP or SSP region is used, it has been completely processed, saving them Block RAM on the Matrix Obtaining block; then the XOR operation process begins. This process executes the same sweep as the MATLAB algorithm, when it’s done it finds the clothing quantity in order to apply the clothing threshold.

When the entire row has been filtered, it passes its data to the Face Region Location Block which calculates the average and retrieves the face coordinates to pass them to the PSRAM Control and stay idle until the next row is received and processed. Figure 3-13 shows the algorithm flowchart. Once the height of the FSP or SSP is reached, then all signals return to their initial state value and waits for a new picture.

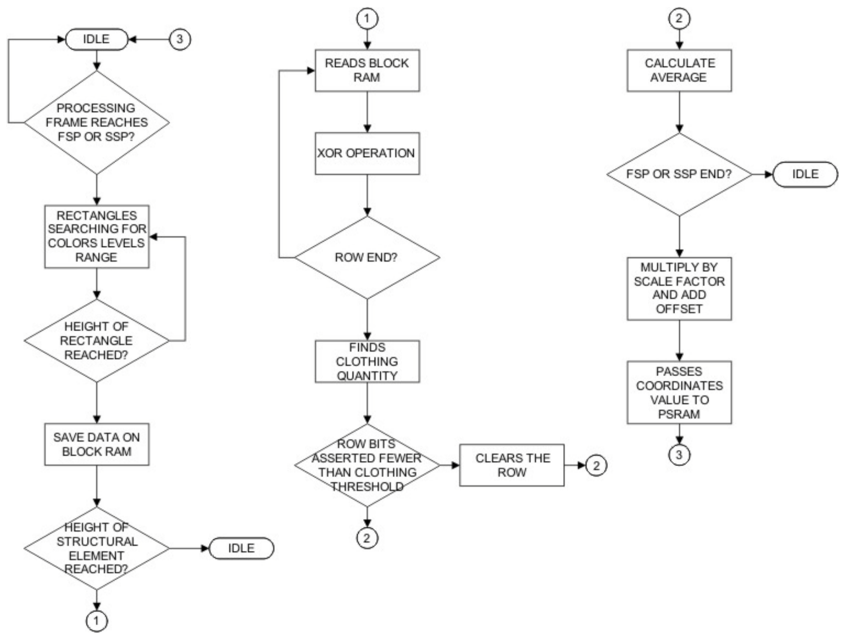

IDLE
3
PROCESSING FRAME REACHES FSP OR SSP?
RECTANGLES SEARCHING FOR COLORS LEVELS RANGE
HEIGHT OF RECTANGLE REACHED?
SAVE DATA ON BLOCK RAM
HEIGHT OF STRUCTURAL ELEMENT REACHED?
IDLE
1
1
READS BLOCK RAM
XOR OPERATION
ROW END?
FINDS CLOTHING QUANTITY
ROW BITS ASSERTED FEWER THAN CLOTHING THRESHOLD
CLEARS THE ROW
2
2
2
CALCULATE AVERAGE
FSP OR SSP END?
IDLE
MULTIPLY BY SCALE FACTOR AND ADD OFFSET
PASSES COORDINATES VALUE TO PSRAM
3

Figure 3-13 Image Processing flowchart for the VHDL equivalent algorithm

3.5 Summary

The image processing algorithm development was explained. To illustrate the entire process images showing different stages were included for better comprehension. Key changes which were used to take advantage on the FPGA resources were stated and corresponding flowchart presented.

Next chapter contains tests made using MATLAB and the FPGA showing the performance of the system developed on the public transport. Equivalent binary matrices obtained by both MATLAB and FPGA will be presented along with tests for configuring the camera.

Chapter 4 Tests and Results

4.1 Introduction

At the beginning tests were made while developing the modules which communicate with the FPGA: Camera, PSRAM, SD and RTC; receiving 1280x1024 images, checking that the images were correctly stored on the SD card so that they were read by a computer correctly.

Once this first stage of the developing was done; people's pictures were taken on the bus in order to develop the algorithm on MATLAB, using real data. Finally, once the algorithm was fully developed then corresponding VHDL algorithm was completed for each system's module, tests were made for each comparing results on both platforms.

Also, experimental tests were made using accelerometers to detect people as a possibility for later inclusion on the system. Results are shown on the following sections.

4.2 CAMERA-SD-FPGA Tests

Difficulties were found when configuring the camera due to the lack of register's configuration detail on readily available Omnivision's literature. In order to configure correctly the camera many pictures were taken observing the effects on them (pictures) by changing registers parameters. Figure 4-1 shows several images taken with incorrect configurations (high contrast, low gain, large exposure time, etc).

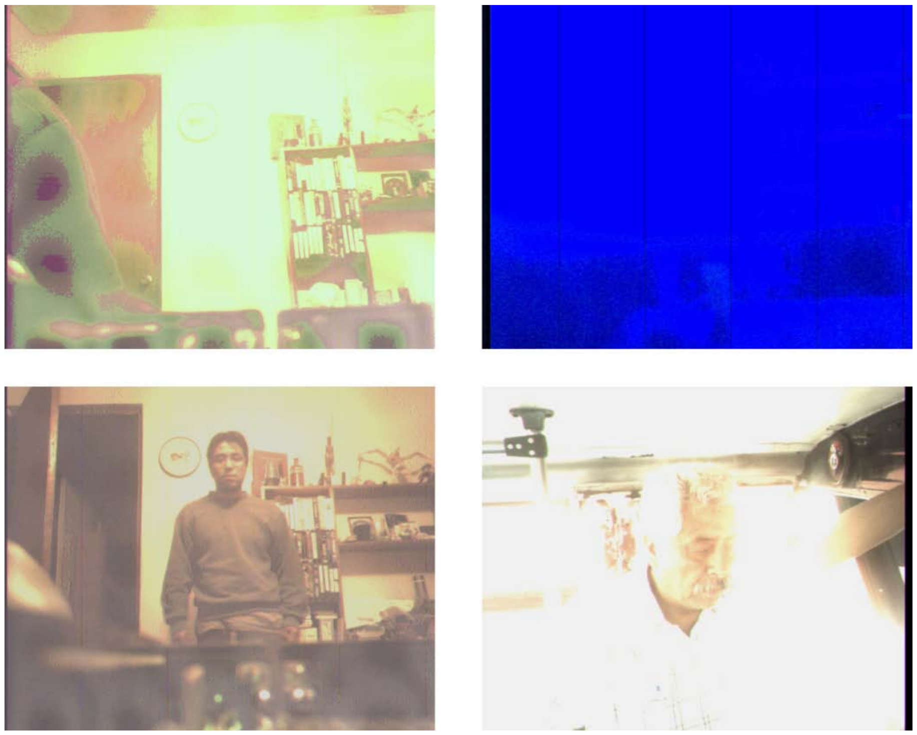


Figure 4-1 Camera tests without a correct configuration

This approach was very slow in figuring out a correct configuration because of the several registers within the camera. After a more extensive search we found a Linux based driver [13] for the same camera and where the important registers were configured. Not all the registers were configured by the configuration provided by the Linux driver; there was the need for changing others for the system’s purposes (resolution size, RGB555 format, etc). Figure 4-2 shows images taken using the registers correctly configured for the system’s needs.

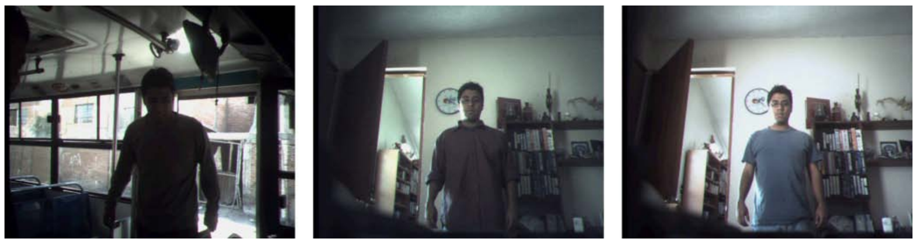


Figure 4-2 Camera configured correctly

Night pictures (dark ones) were also taken with the camera finding that these pictures are good only when artificial light is provided, even when the camera has its Night Mode operation register. When enabling the Night Mode operation, the exposure time highly increases leading the camera to take only blurred pictures, which are not useful. Figure 4-3a and Figure 4-3b show images without and with artificial light respectively. Figure 4-3c shows a person's picture taken with the Nigh Mode enabled; where it can be observed it's blurred, because of time exposure increase and people-camera movement.

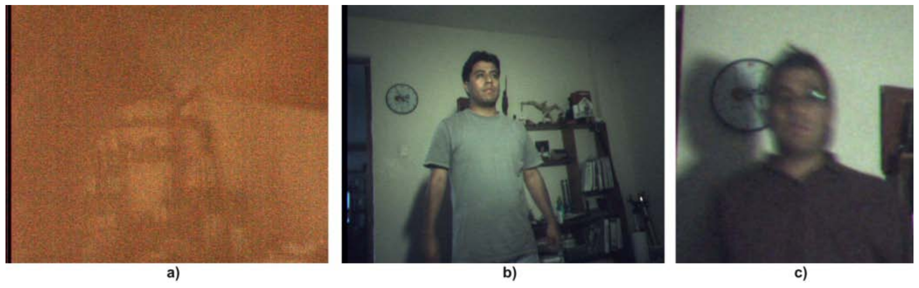


Figure 4-3 a) Dark environment picture, b) Illuminated dark environment, c) Camera's Night Mode enable

When observing these results, it could be determined that in order for the system to work in the poor light conditions, the bus needs artificial light on the steps to illuminate people's faces, which is beyond the system's objectives and, thus, only daylight tests, and pictures, will be shown.

Figure 4-4 shows pictures taken on the public transport. In some pictures shown on Figure 4-4, the contrast seems to affect the image making the inside of the bus very dark, including people, when light from the outside is very high. On the next section, it will be proven that this doesn't affect the processing algorithm and since a post-processing can be performed when acquiring the pictures on a computer, this register's configuration is good enough.

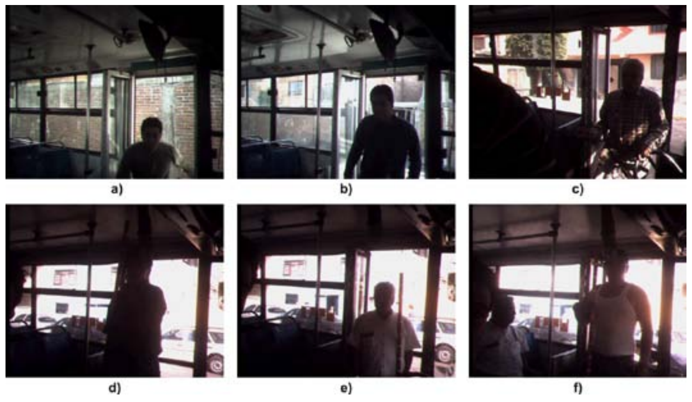


Figure 4-4 Public transport pictures taken with the SD-FPGA system

4.3 Algorithm Tests using MATLAB

After establishing the configuration of the camera, the tests on the bus started in order to design the MATLAB algorithm. Face locating algorithm will be used on images from Figure 4-4 using the same numbering to refer each picture.

Figure 4-5a, b, c, e are FSP areas and Figure 4-5d and f are SSP areas processed applying the modified histograms. Figure 4-6 shows the binary matrices representing the same images highlighting the asserted image regions.

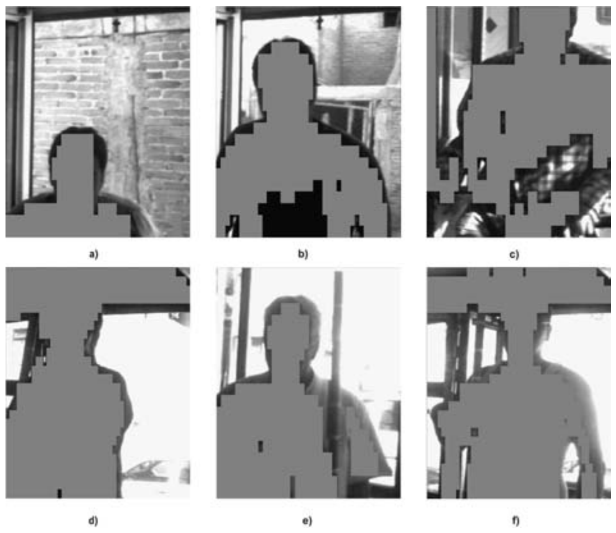


Figure 4-5 Processing rectangles applied to Figure 4-4 pictures

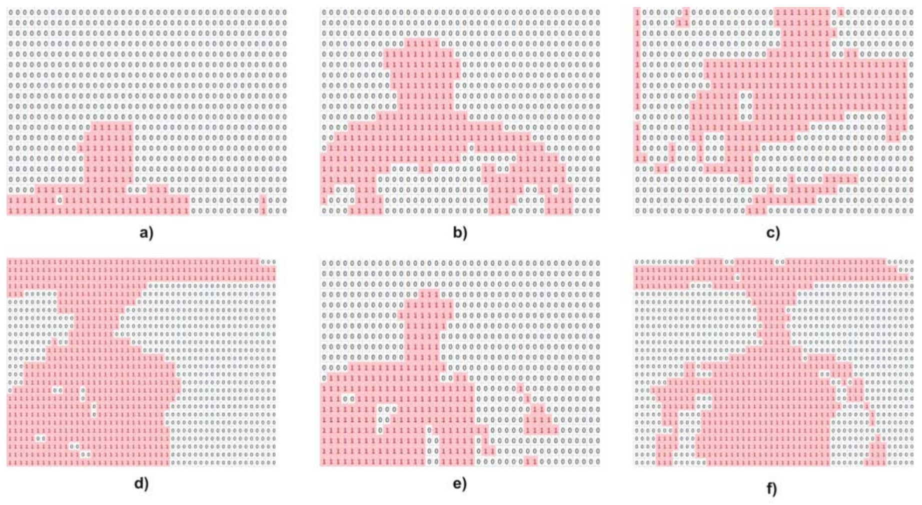


Figure 4-6 Binary representation of Figure 4-5

Once the matrices have been obtained, the XOR and clothing filters are applied. Figure 4-7 shows the binary matrices obtained after XOR filtering and Figure 4-8 shows the matrices after applying the clothing filtering.

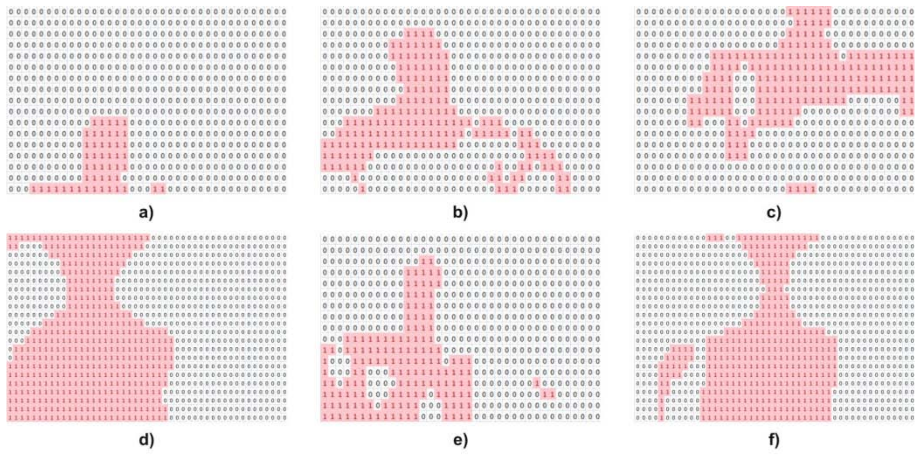


Figure 4-7 XOR filtering applied to Figure 4-6 matrices

Finally, Figure 4-9 shows the located face of all these images after retrieving coordinates by using the average. Figure 4-9b and c show noise which wasn't eliminated using the XOR and Clothing filters. Figure 4-9b is not affected by this noise due to the high quantity of data grouped on the face and less noise bits making the average work as expected. Figure 4-9c it's affected by this noise because a less grouped bits quantity on the face and the noise is more than a half of this face bits. However the face is still present on the image making possible to identify the person.

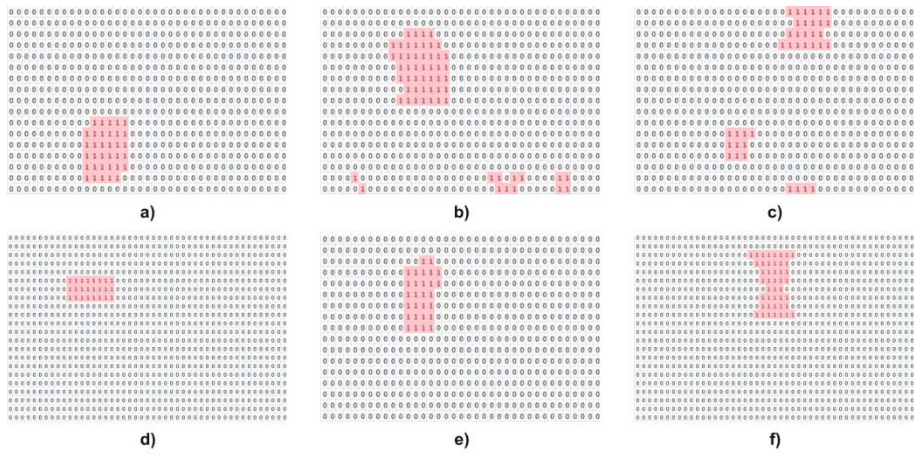


Figure 4-8 Clothing filtering from matrices on Figure 4-7

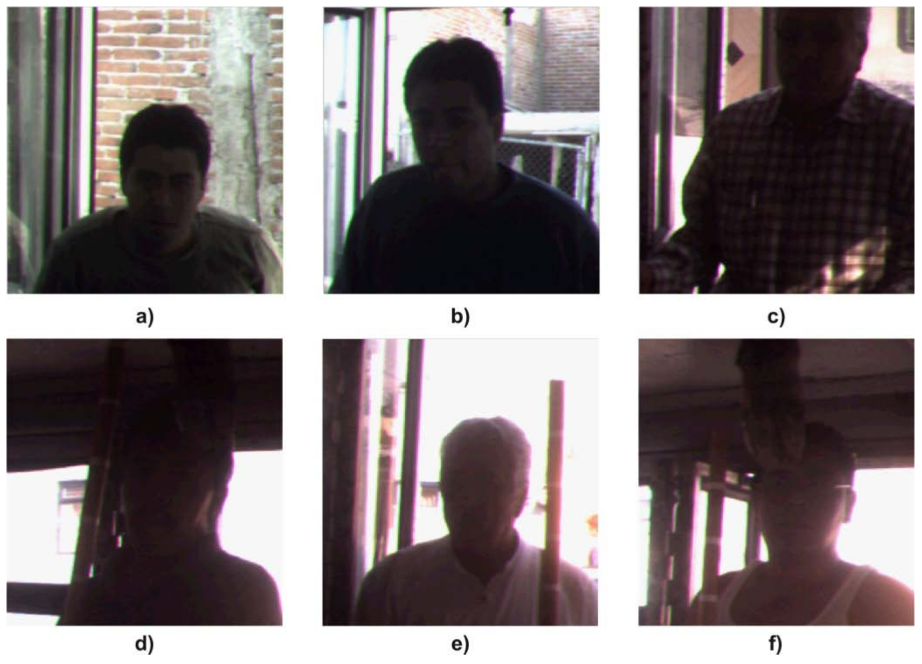


Figure 4-9 Faces located from Figure 4-4 pictures by MATLAB

4.4 Algorithm Tests using VHDL

To verify that the VHDL algorithm was working as the MATLAB algorithm, three different tests were made.

- Confirm that the binary matrix, generated by the modified histograms algorithm implementation using VHDL is compatible.
- Verify correct implementation of XOR and Clothing filters by filling the binary matrix with known patterns and observe if the located image's area is the same on the FPGA and MATLAB.
- Finally, test the entire VHDL algorithm on the public transport.

To confirm the binary matrix was receiving correct information, a LabVIEW interface was developed to receive the binary matrix data from the FPGA (see Figure 4-10) to a PC. The interface generates a spreadsheet with the binary data which can be imported by MATLAB; thus easing confirmation of correct implementation of the XOR and Clothing on the FPGA.

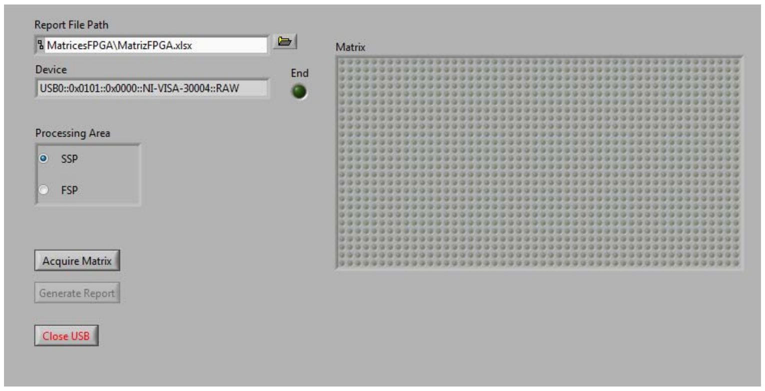

Report File Path: MatricesFPGA\MatrizFPGA.xlsx
Device: USB0::0x0101::0x0000::NI-VISA-30004::RAW
Processing Area: [✓ SSP] FSP

Figure 4-10 LabVIEW interface used to retrieve binary matrices from the FPGA

Figure 4-11a shows matrices obtained from the FPGA by the LabVIEW interface for a SSP and FSP, and Figure 4-11b shows corresponding matrices acquired by MATLAB.

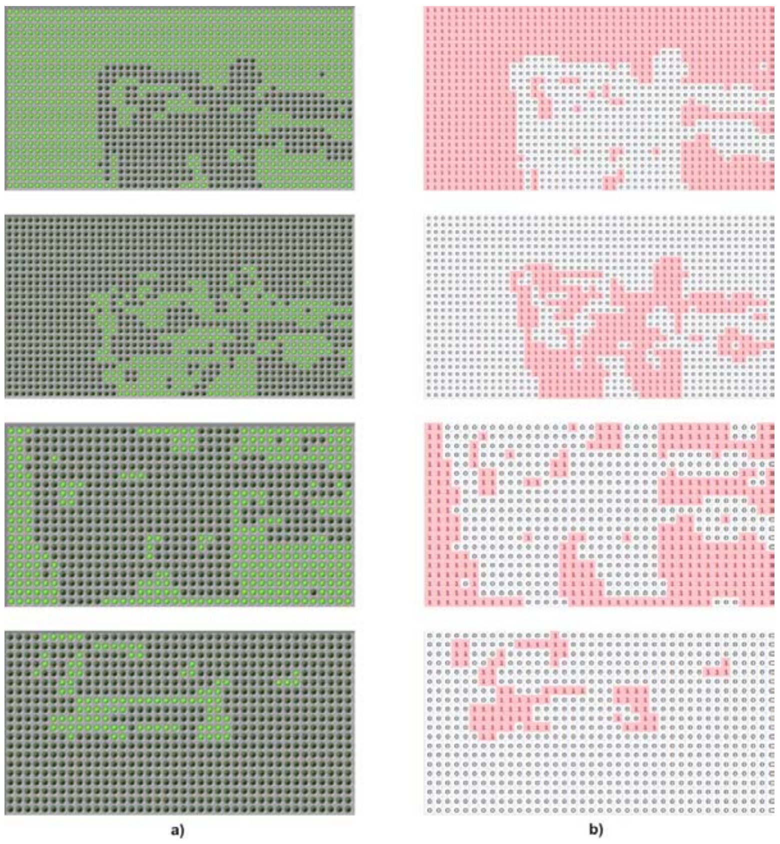


Figure 4-11 Comparison between binary matrices obtained by a) FPGA and b) MATLAB

These images were taken on the same environment on a short time interval with the purpose to have similar light conditions (light conditions change with time, hence, assuring that the same conditions were met, is almost impossible without using artificial lighting and environment control). On Figure 4-11 one can observe that matrices are equivalent, for the purpose of locating face cluster, but not the same. This can be due to the light conditions, by the analog to digital conversion and the inherent noise within the camera; never the less, patterns can be found on both FPGA and MATLAB which demonstrates the VHDL algorithm is working as expected.

It's important to note that these tests were not performed on public transport. The purpose of these analyses was to confirm the correct algorithm operation, so that data matrices were not important but the equivalent behavior.

Next, test consisted on taking pictures of known patterns (see Figure 4-12) by filling the binary matrix of the FPGA with these known patterns, the correct behavior of the XOR filtering, Clothing filtering and average calculation were verified.

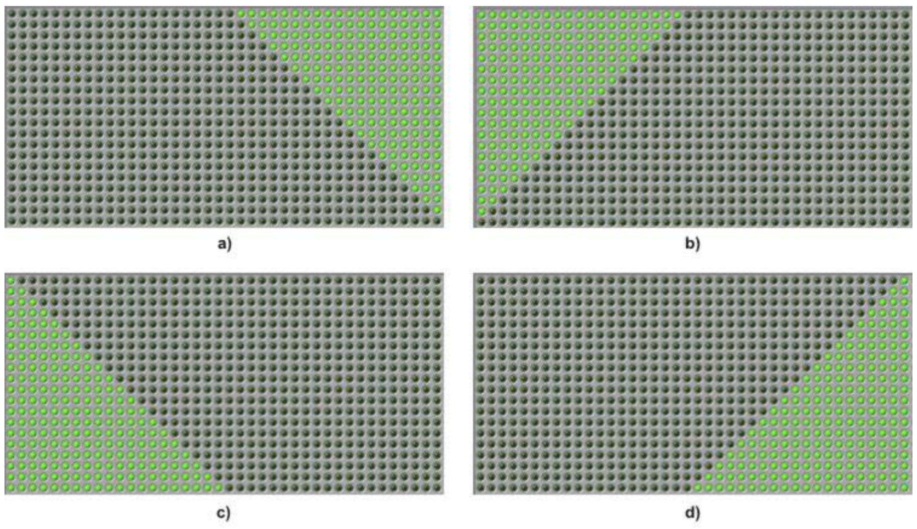


Figure 4-12 Known patterns applied to the binary matrices within the FPGA

In order to accomplish these tests, pictures were taken with the FPGA system and the known pattern was received by LabVIEW and exported to a spreadsheet. Then, without touching the system, full 1.3 megapixel pictures were taken. Finally the spreadsheet was used by MATLAB to locate the image area corresponding to the known pattern and these pictures were compared with those found by the FPGA system; equal pictures show the algorithm is working correctly, Figure 4-13 shows the FPGA images on the left and the MATLAB images on the right. The numbering on Figure 4-13 corresponds to the numbering for the known patterns on Figure 4-12.

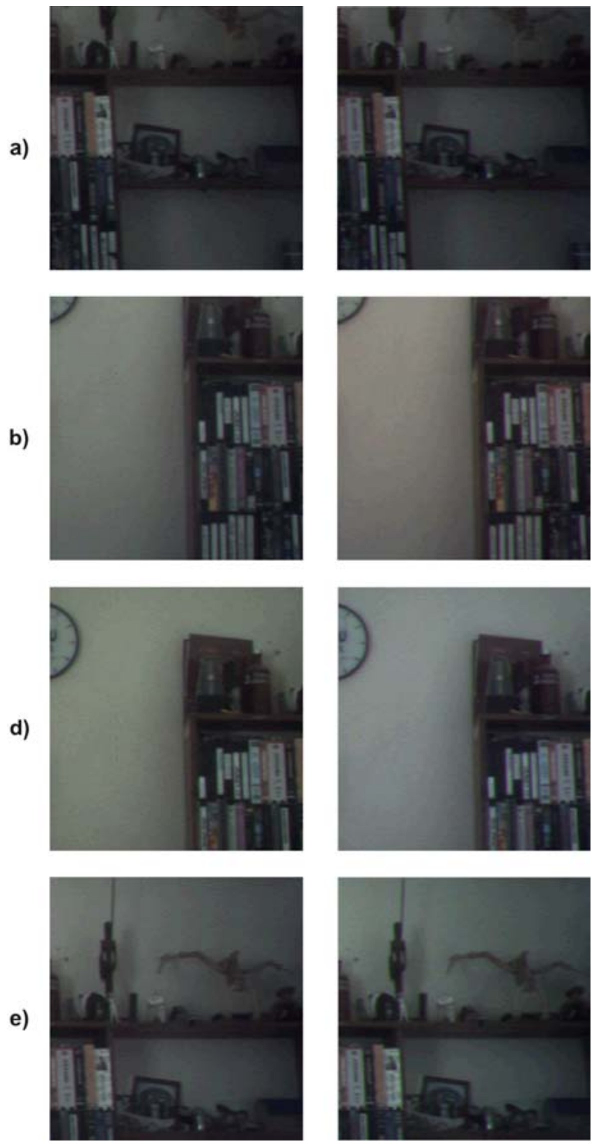


Figure 4-13 Areas located by the knows patterns from the FPGA (left) and MATLAB (right)

Once proven that the algorithm on the FPGA system was working as the algorithm on MATLAB, the final tests were made on public transport. Figure 4-14a shows the full 1.3 megapixel obtained by the camera on the public transport; Figure 4-14b shows the FSP used to locate people faces, retrieved by MATLAB from Figure 4-14a; finally, Figure 4-14c shows the face located from the FPGA.

Finally, Figure 4-15a shows the matrix and image recovered by the FPGA and Figure 4-15b shows the matrix and image recovered by MATLAB using a full 1.3 megapixel image.

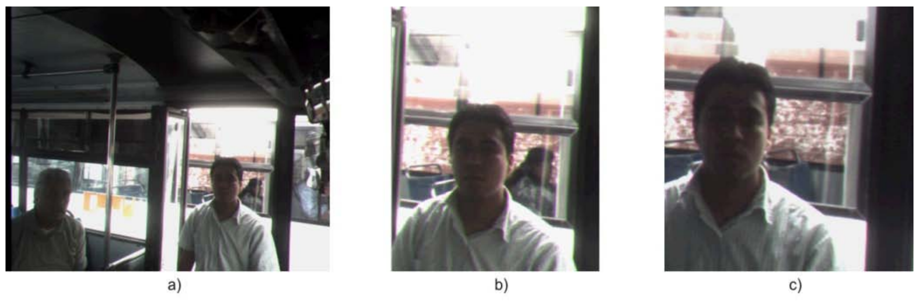


Figure 4-14 a) Full 1.3 Megapixel image, b) FSP retrieved with MATLAB, c) Face area located by the FPGA

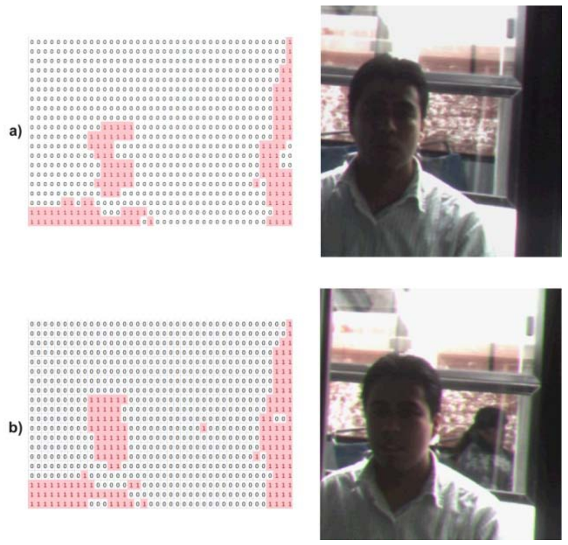


Figure 4-15 Comparison between binary matrix and face area from a) FPGA and b) MATLAB

4.5 Accelerometer Tests

Experimental tests were made to determine if accelerometers could be used as a detector for people presence on the entrance of the bus using a Freescale's ZSTAR2 wireless kit (see Figure 4-16). This kit includes a 3 axis

MMA7360L accelerometer, and an USB interface which communicates with the computer and software to acquire data.

Figure 4-16 ZSTAR2 wireless kit from Freescale [14]

Kit's software is very limited and doesn't allow the user to see the data and save it to a file. This was solved by developing a LabVIEW application which exported the obtained data to be analyzed with MATLAB.

Tests were made on different positions as Figure 4-17 shows. People get on and off the bus while acquiring data saving it on files for later analysis.

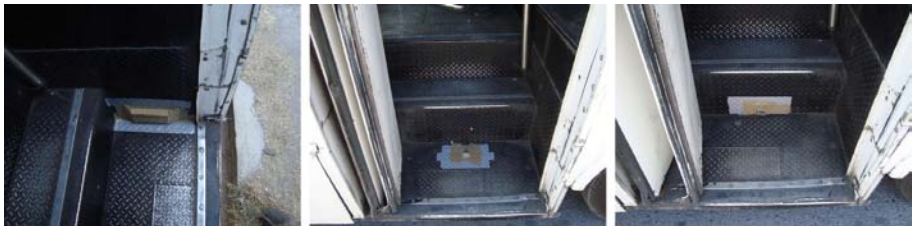


Figure 4-17 ZSTAR2 placed on different position on the public transport

After obtained the data and importing it to MATLAB, patterns were observed when people get on and off the bus and depending on position, signals involved were stronger or weaker for different accelerometer axis that sensed a particular signal, this depending on the kit's location on the bus. Thus to avoid confusion, instead of referring to these axis as 'x', 'y' and 'z' they will be referred as 'bus direction' (BD), 'perpendicular to floor' (PF) and 'perpendicular to bus side' (PS).

When observing the first tests, it can be seen that the BD axis, along with noise vibration, follows the bus acceleration and stopping, the PF axis follows the up and down vibrations and PS would sense inclination. Analyzing this data, it could be inferred that BD axis would give information about the public transport breaking when people getting on and off the bus followed by acceleration; when the bus stays idle, the PF axis would give information about people getting on and off the bus by vibrations sensed on the steps and, finally, the PS axis would help the PF axis by sensing inclination (if there is any) on the bus by people's weight added to the bus.

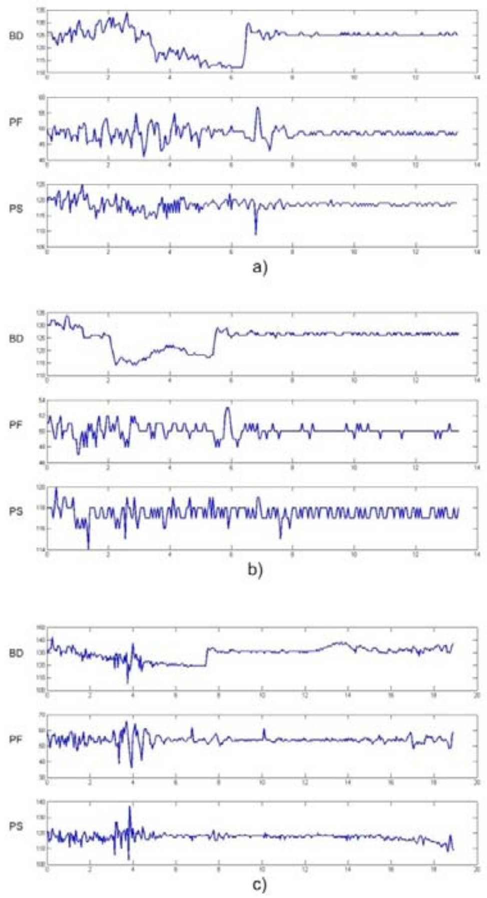


Figure 4-18 a) A person getting on the bus, b) A person getting off the bus, c) A person getting off and on the bus

When analyzing data from tests with people getting on and off, some of the assumptions made before were confirmed. BD axis does have the acceleration and breaking of the bus at any time and PF axis has the up and down vibration caused by, but not only, people getting on the bus. On PS, in some cases an inclination was observed but not always, which could be due to people's weight, or street condition. Figure 4-18a shows a person signals getting on the bus, Figure 4-18b shows a person signals getting off the bus and Figure 4-18c shows a person getting off and on the bus using the same axis' positions. Vertical axis is the amplitude on an 8-bit resolution and horizontal axis is time in seconds.

This tests show that accelerometers are promising sensors to detect people but, a more extensive analysis must be done in order to absolutely confirm this hypothesis. Also, and finally, when acquiring data with accelerometers, situations about the driver's behavior can be obtained, which could be of great use by public transport owners. Figure 4-19a shows the sensed data when passing through a pot hole and Figure 4-19b shows when passing on a speed bump.

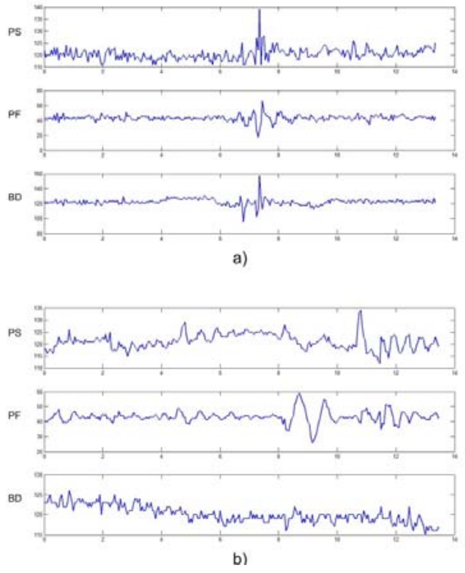


Figure 4-19 a) Pot hole sensed data, b) Speed bump sensed data

4.6 Summary

System tests with MATLAB and FPGA were shown. Image processing tests with MATLAB included people face being located using binary matrices.

The FPGA tests were shown to prove the algorithm was working as the MATLAB one. The binary matrices extracted from the processed image position are shown with those obtained by MATLAB to highlight the equivalence between them. To show that filters and average calculation were working as expected, image regions were obtained using known patterns on both MATLAB and FPGA. Tests on the public transport show that the algorithm implementation on FPGA is working within the prototype system.

Finally, accelerometer tests on the public transport are shown. These tests allow us to expect that accelerometer signals can be used as a detecting people device. Also, it’s interesting to note that operator’s behavior can be inferred by processing some of these sensors’ signals.

On the next, and final, chapter, conclusions and future work will be presented.

Chapter 5 Conclusion and Future Work

5.1 Conclusions

A face extraction system based on a camera chip, SD card, and FPGA for the public transport security was developed, using a 1.3 Megapixel camera and a 1GB memory SD.

Face algorithm development was based on regional characteristics from skin color distribution, making it tolerant to light conditions and skin tones.

Tests show that current implementation is capable to take 2 useful pictures from each person as they get on the bus.

Processing and saving only the face’s area reduces the SD storing time approximately from 1.3 seconds to .45 seconds and uses almost 9 times less space.

System prototype fully supports SD hot-insertion, making it capable to remove and insert the SD memory on the system without turning it off and on each time. Practically all current systems; devices and computers include software viewers for BMP files. Thus; resulting images can be reviewed using almost any SD reading device.

As dedicated hardware is being used, no hidden processes are involved, which could result in system failure.

There is no human intervention during the entire Detection→Processing→Storing process, which makes it less expensive than the traditional video transmission products that require human intervention and monthly communication rental.

The accelerometer tests show them as a promising way to detect people as they get on and off the bus. These tests also show that these sensors are useful to retrieve important data about the operator's driving behavior.

Due to the camera's low light sensitivity, operation on dark environments (night operation) must be performed adding artificial light to the public transport.

5.2 Future Work

The 1.3-Megapixel camera can be changed for a higher resolution one. As said before, due to the high demand of cameras by daily used devices, their manufacturing costs get lower and, hence, a 2 Megapixel camera could have a similar price as a 1.3 Megapixel one.

The JPEG format conversion can be implemented as a block on the FPGA or as an external device, dramatically reducing the space on memory used by each picture.

The SD memory can be update to a micro SD memory which is more commonly used by daily devices. Also, storing time could be reduced: micro SD has a data rate of 50 MHz (SD has a maximum of 25 MHz).

An LCD screen device can be added in order to review the stored pictures in-system. This could be useful for quick thief identification.

Addition of an accelerometer module can be used as the detection unit sensor. Also, data analysis can be stored in order to give information to the transport owner about the number of passengers who got on the bus at the end of the day and information about the driver's performance and behavior.

A video transmission thru an USB interface could be developed, in order to observe, in real time, the effects of changing camera's registers data.

Appendix A Images from Tests and Results

Several images will be presented that were taken during the development process. Images including the first prototype, the LabVIEW interface, the final results, etc. are shown.

A.1 First Prototype

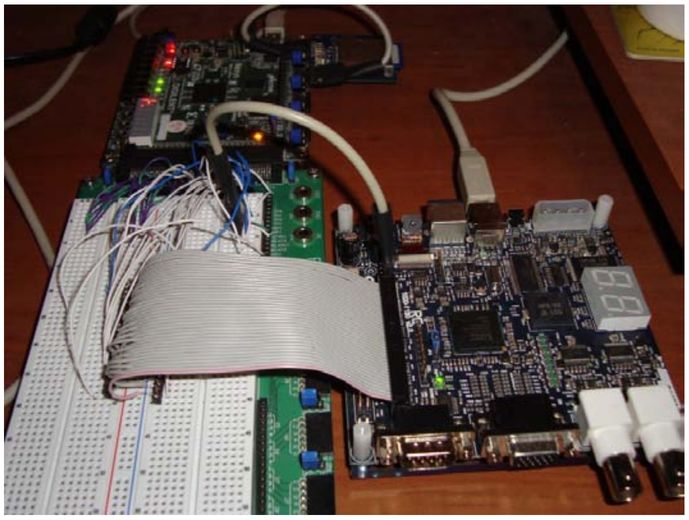


Figure A-1 Celoxica-Nexys Prototype 1

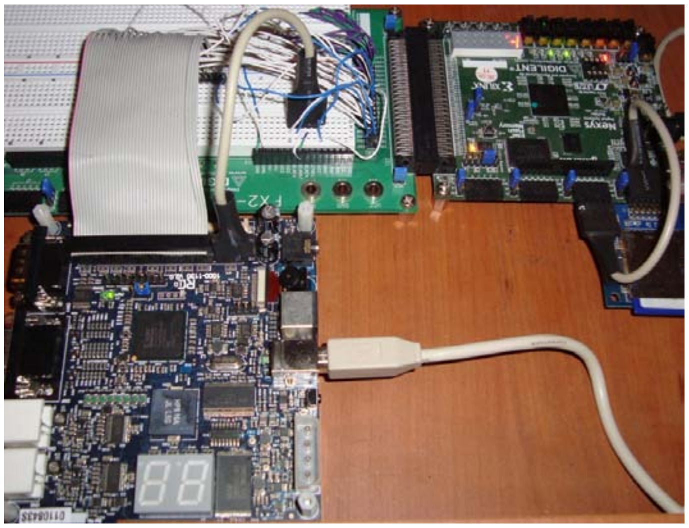


Figure A-2 Celoxica-Nexys Prototype 2

A.2 Final Prototype

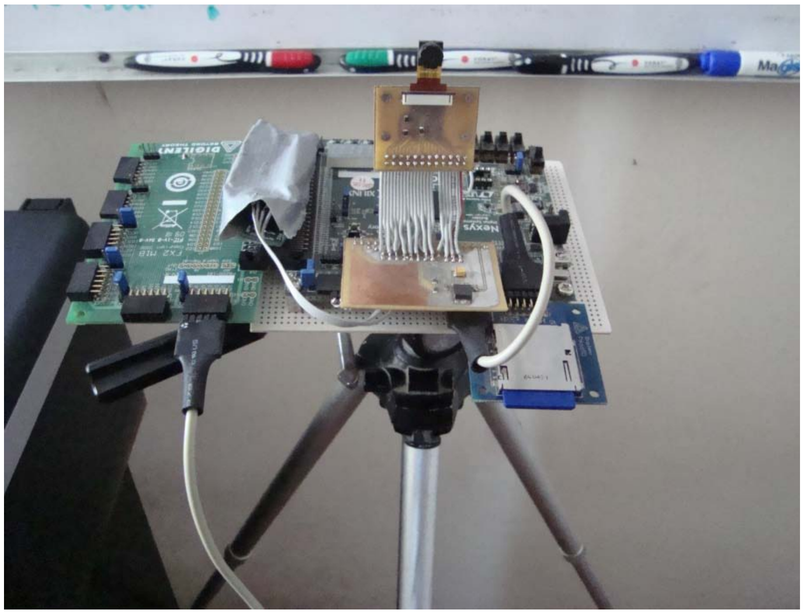


Figure A-3 Nexys only final prototype 1

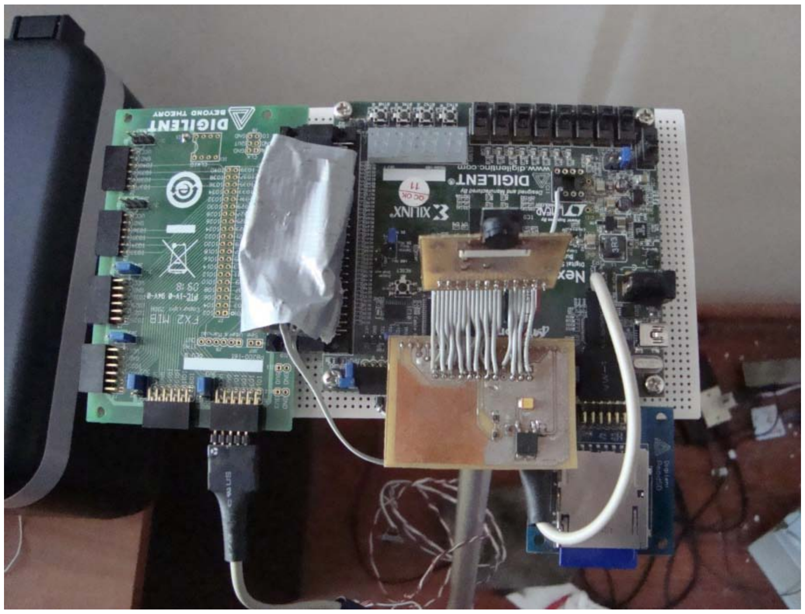


Figure A-4 Nexys only final Prototype 2

A.3 LabVIEW USB Interface

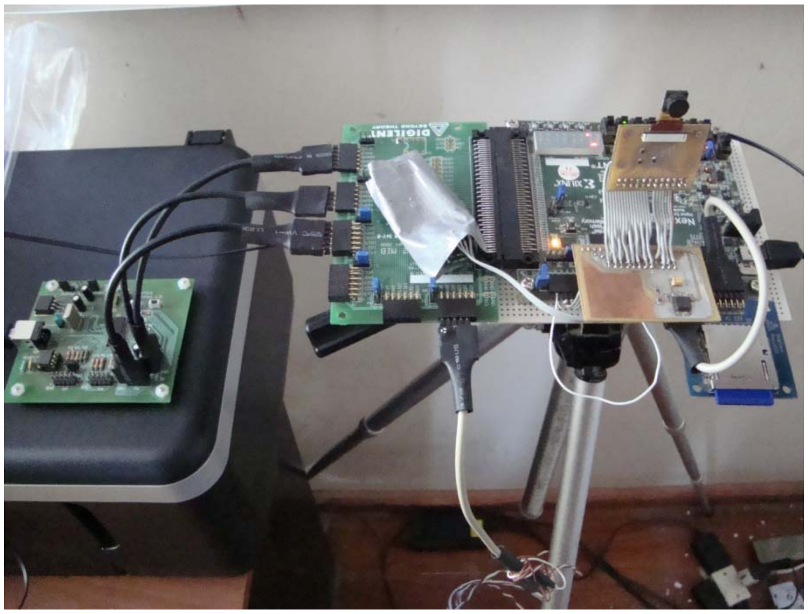


Figure A-5 USB-Nexys Interface

A.4 SD Reading

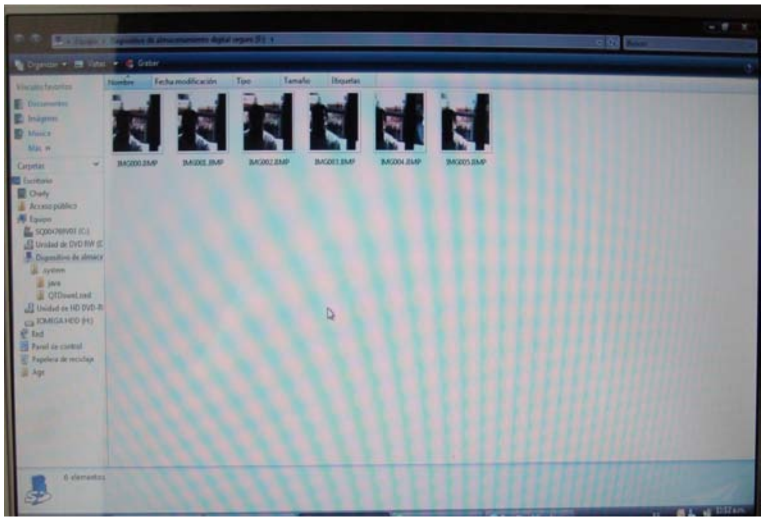


Figure A-6 Windows Vista SD Reading

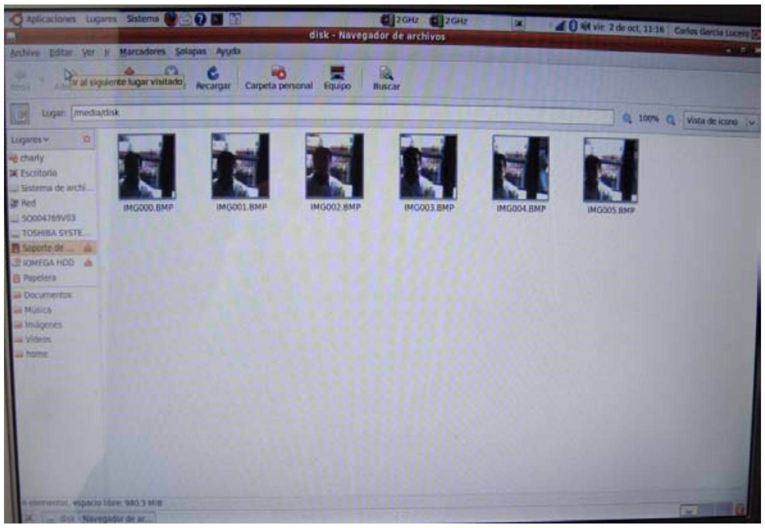


Figure A-7 Linux Ubuntu Operative System SD Reading

A.5 Final Tests

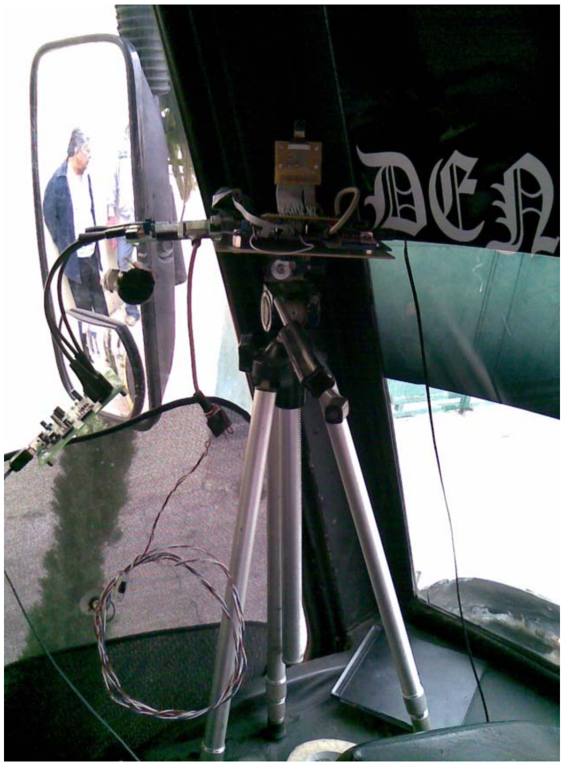


Figure A-8 Public Transport Bus System's Position

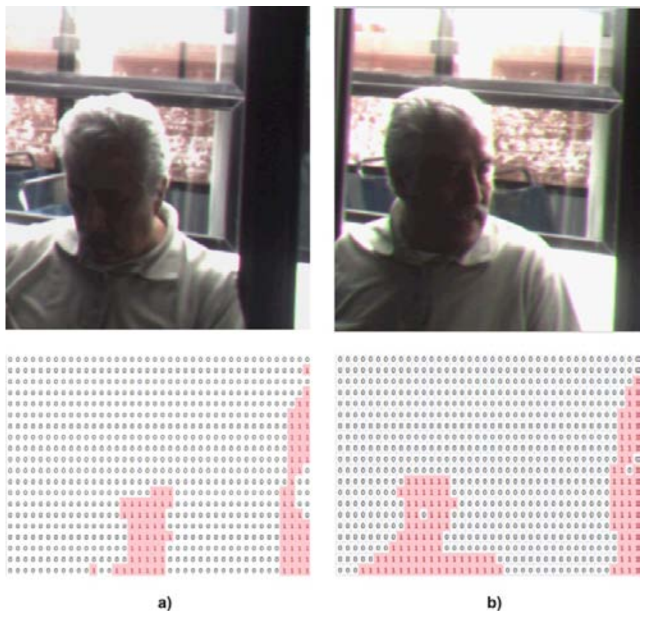


Figure A-9 FSP area 1. a) FPGA and b) MATLAB

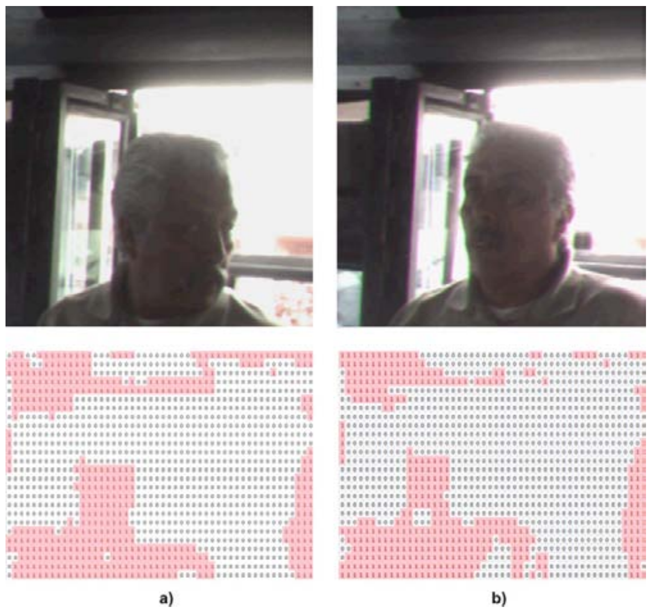


Figure A-10 SSP area 1. a) FPGA and b) MATLAB

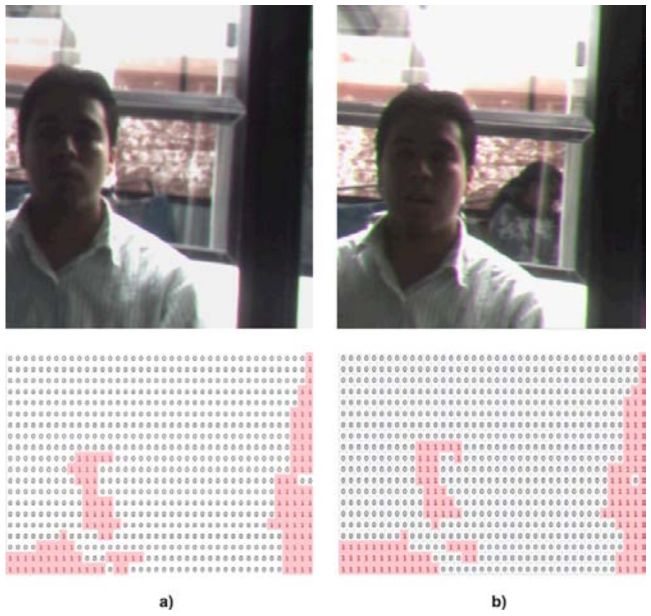


Figure A-11 FSP area 2. a) FPGA and b) MATLAB

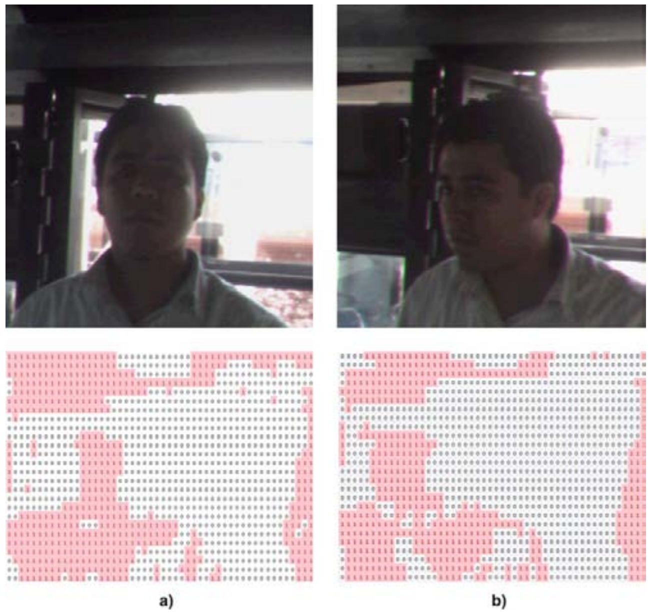


Figure A-12 SSP area 2. a) FPGA and b) MATLAB

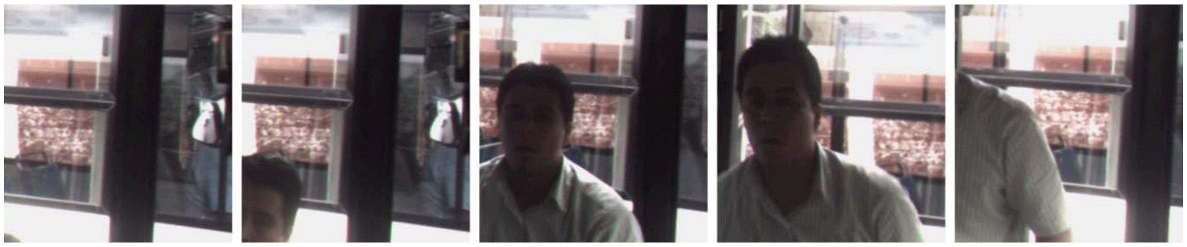


Figure A-13 FSP area Sequence 1 on FPGA

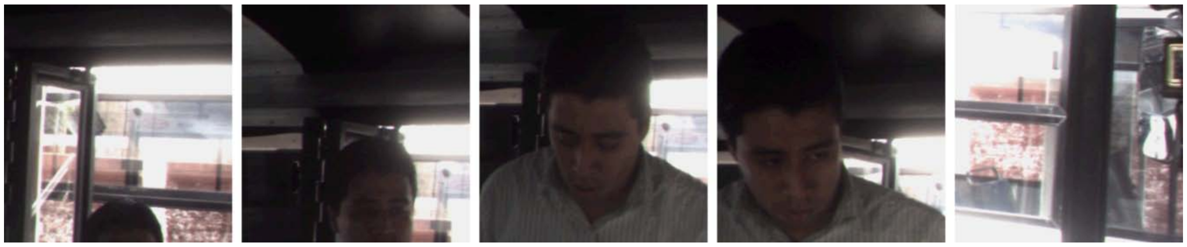


Figure A-14 SSP area sequence 1 on FPGA

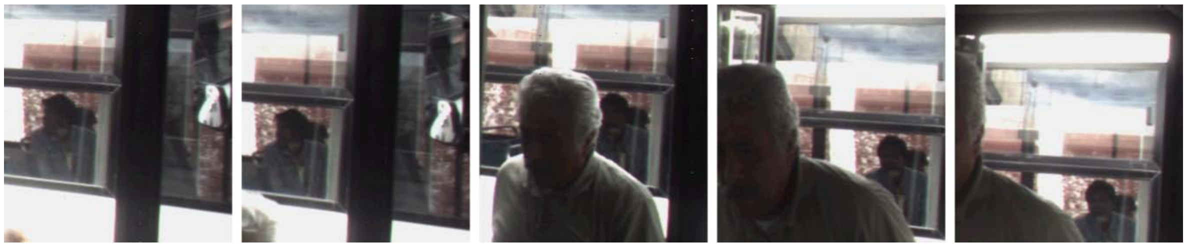


Figure A-15 FSP area sequence 2 on FPGA

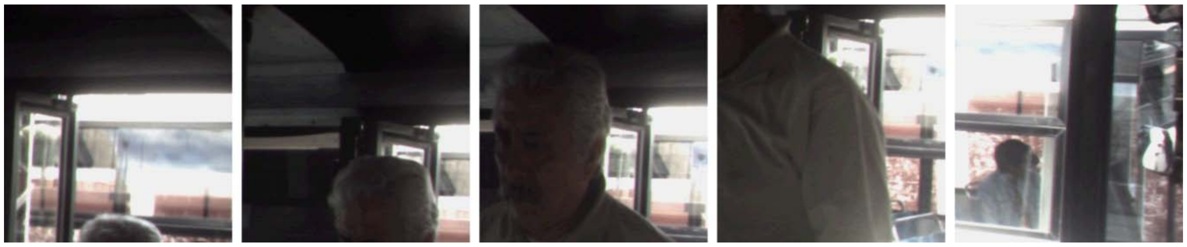


Figure A-16 SSP area sequence 2 on FPGA

Appendix B FPGA’s RTL Schematics

The RTL schematics generated by the Xilinx’s software for each block, including the inner stages from the face detection algorithm are presented within this appendix. The software tool used to synthesize, route and place, and implement the VHDL code into the Spartan3 FPGA was the free software from Xilinx: ISE WebPACK™ 11.2.

B.1 TOP’s RTL

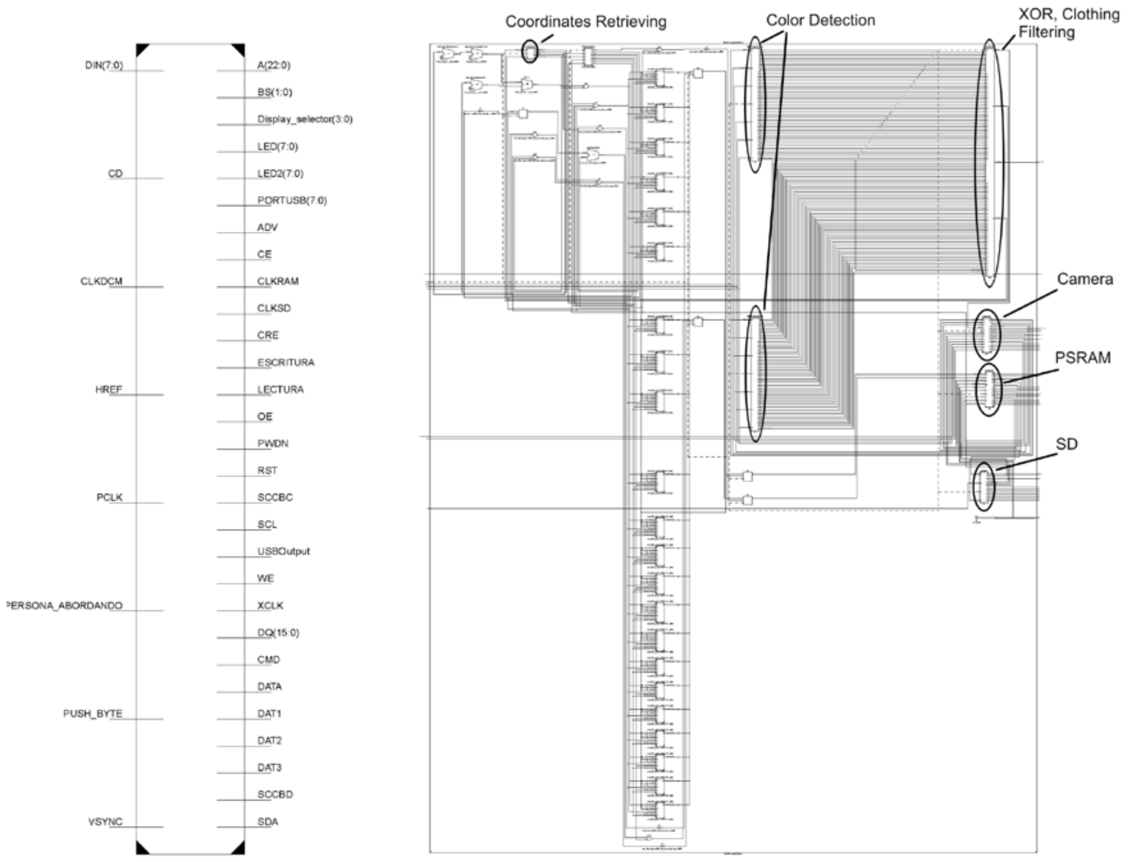


Figure B-1 TOP’s RLT Schematic

B.2 Camera’s RTL

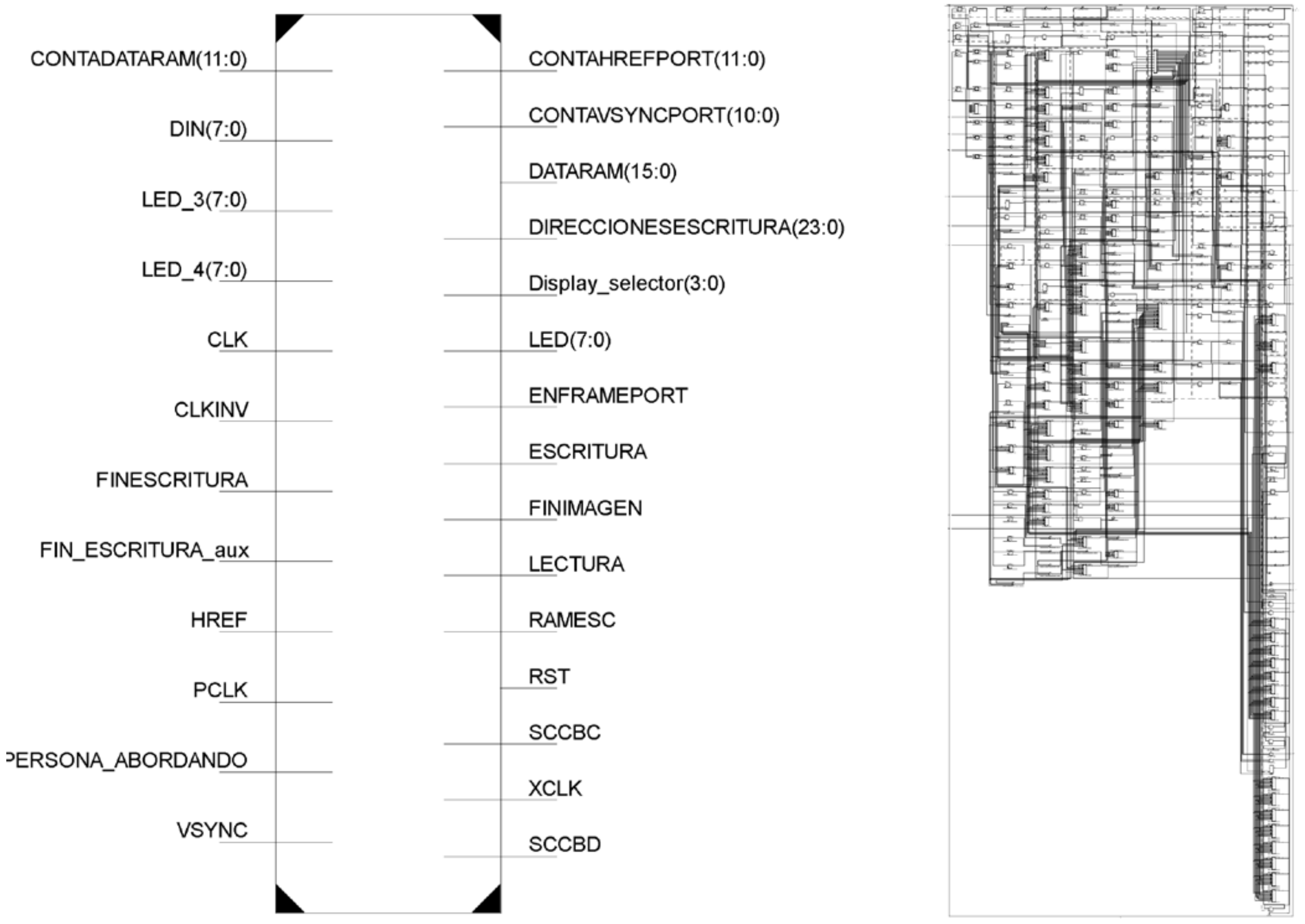
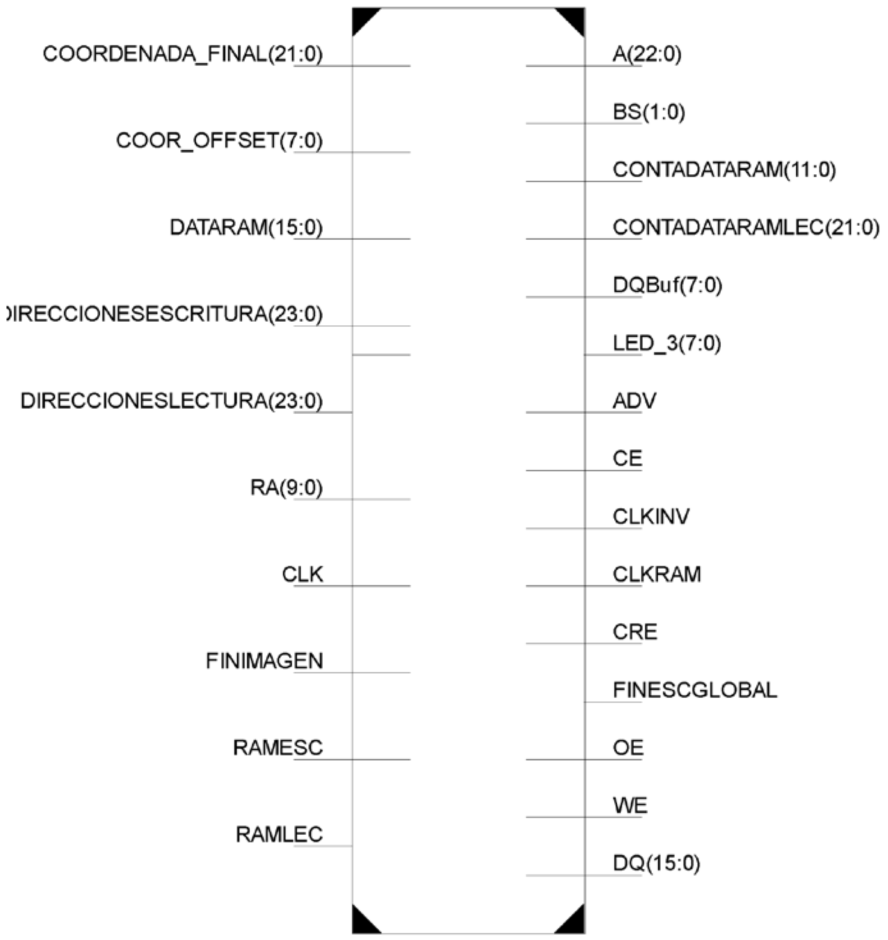


Figure B-2 Camera’s Block RTL Schematic

B.3 PSRAM’s RTL



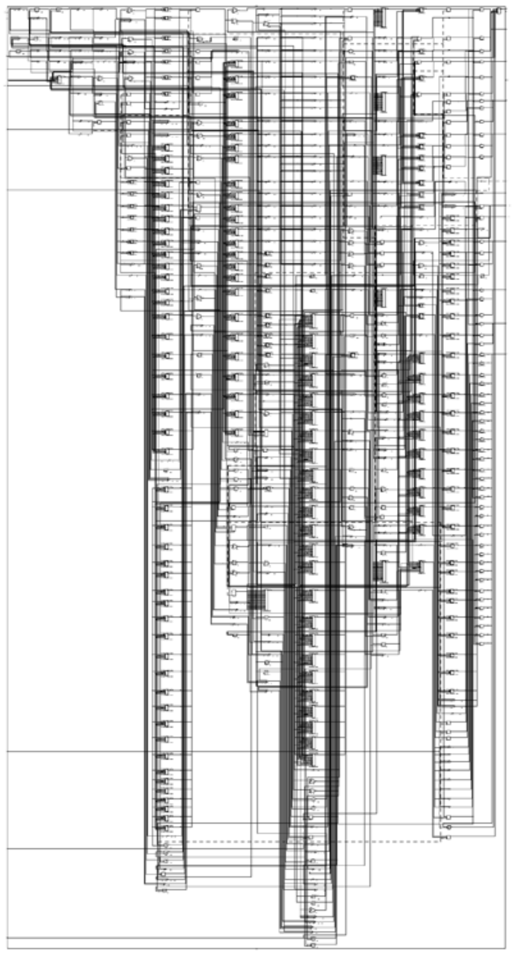


Figure B-3 PSRAM’s Block RTL Schematic

B.4 SD’s RTL

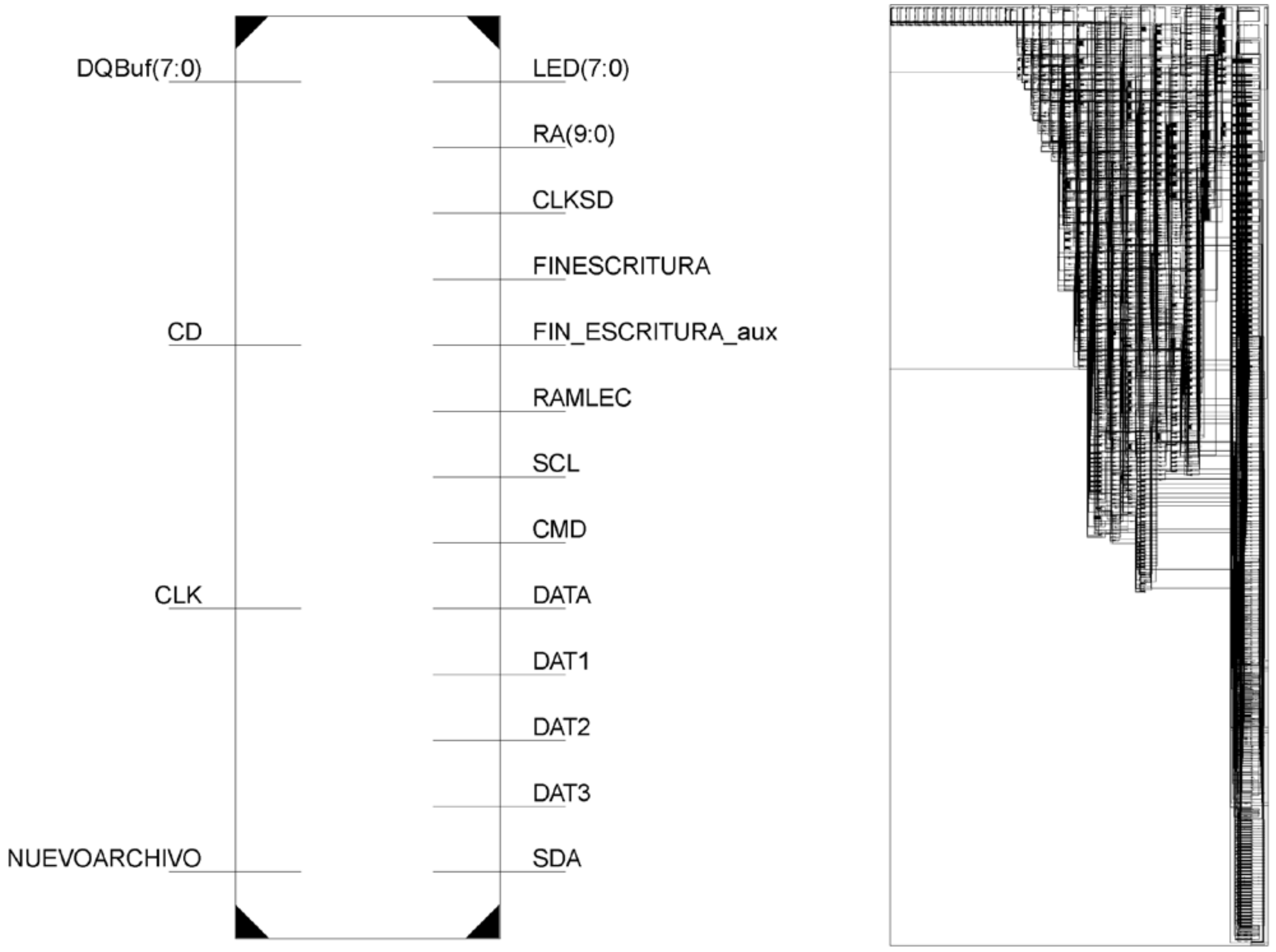


Figure B-4 SD’s Block RTL Schematic

B.5 RTC’s RTL

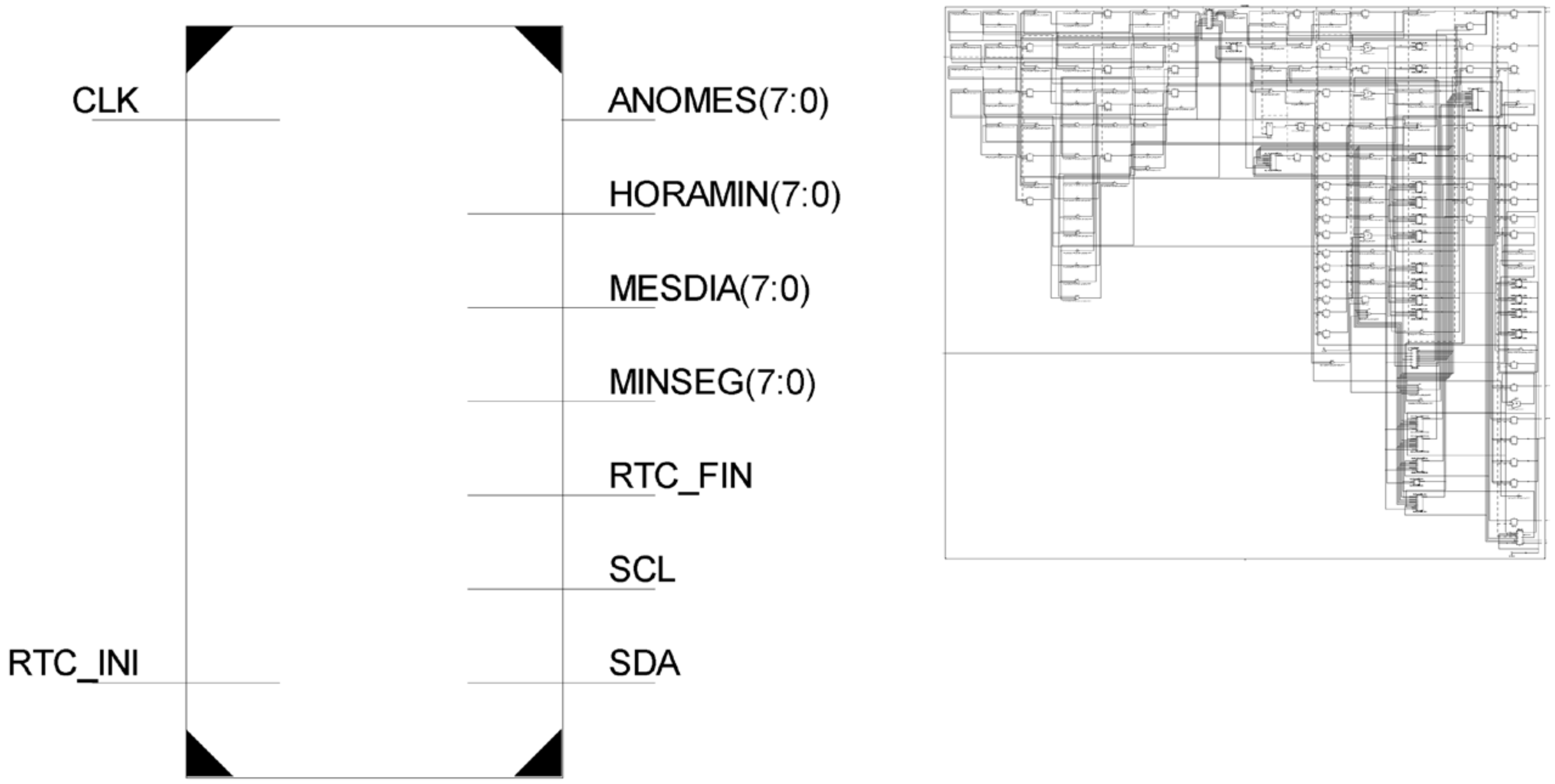


Figure B-5 RTC’s Block RTL Schematic

B.6 Color Detection's RTL

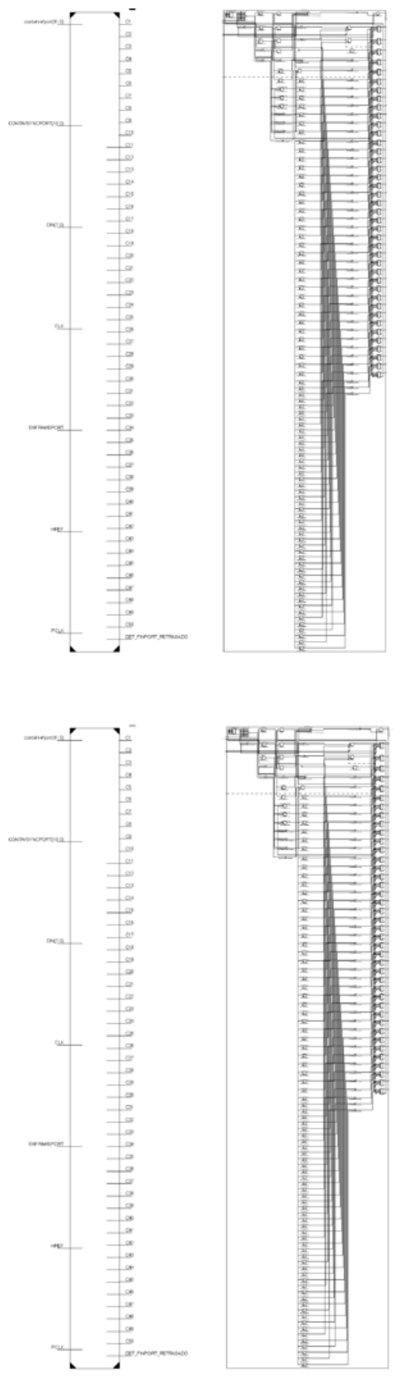


Figure B-6 Color Detection's Block RTL Schematics

B.7 XOR and Cloth Filtering Block's RTL

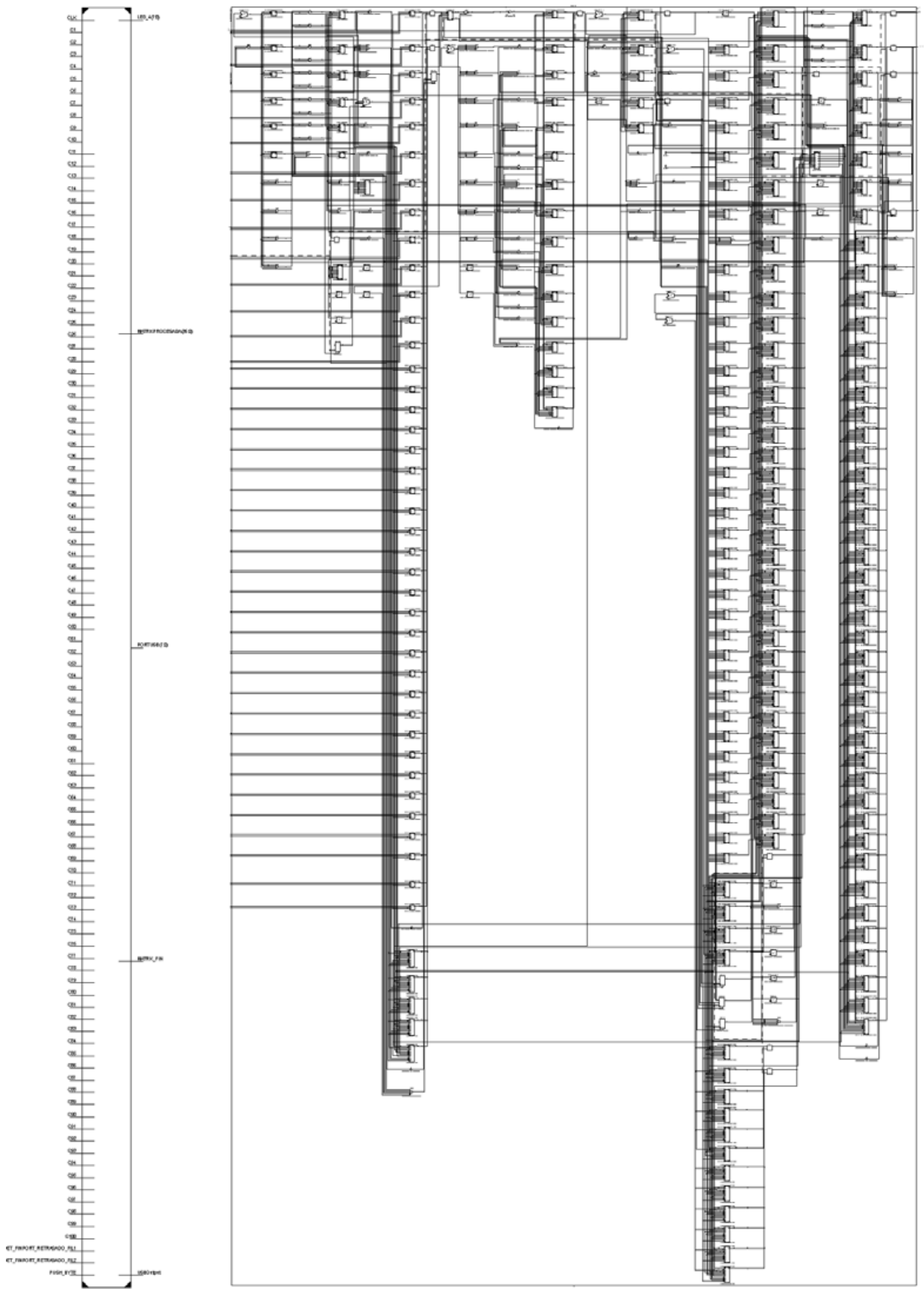
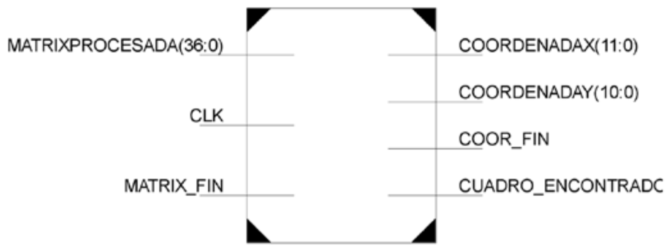


Figure B-7 XOR and Clothing Filtering Block's RTL Schematic

B.8 Coordinates Retrieving Block’s RTL



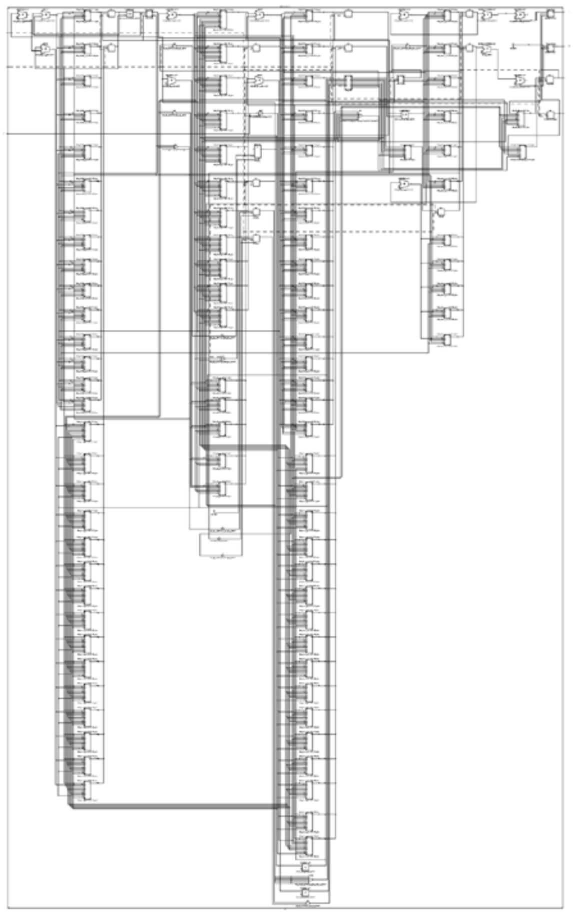


Figure B-8 Coordinates Retrieving Block’s RTL Schematic

REFERENCES

[1] Saúl Vela. *Milenio.com*. March 2009. [Online]. http://www.milenio.com/node/189736. [Accessed: September 03, 2009]

[2] Gerardo Orta. *periodicodigital.com.mx*. June 2009. [Online]. http://www.periodicodigital.com.mx/index.php?option=com_content&task=view&id=92893&Itemid=67. [Accessed: September 03, 2009]

[3] Owen Bowcott. *guardian.co.uk*. May 2008. [Accessed: December 12, 2008]

[4] OmniVision Technologies Inc. "OV9650FSL Color CMOS SXGA (1.3 MegaPixel). Concept Camera Module with OmniPixel® Technology". August 2005. [Online]. www.cs.uni-potsdam.de/techinf/lehre/SS08/./OV9650FSL_DS.pdf

[5] Micron Technology Inc. "1.5, Async/Page/Burst CellularRAM™". 2004. [Online]. http://download.micron.com/pdf/datasheets/psram/128mb_burst_cr1_5_p26z.pdf

[6] SANDISK CORPORATION. "Sand Disk Secure Digital (SD) Card Product Manual, Rev 1.9©". December 2003. [Online]. www.sandisk.com

[7] Matsushita Electric Industrial Co. Ltd (MEI), SanDisk Corporation, and Toshiba Corporation. "SD Memory Card Specifications. Part 2 FILE SYSTEM SPECIFICATION". February 2000.

[8] Philips Semiconductors. "PCF8583 Clock/calendar with 240 x 8-bit". Jul 1997. [Online]. www.nxp.com/acrobat_download/datasheets/PCF8583_5.pdf

[9] Yu-Ting Pai, Shanq-Jang Ruan, Mon-Chau Shie, and Yi-Chi Liu, "A SIMPLE AND ACCURATE COLOR FACE DETECTION ALGORITHM IN COMPLEX BACKGROUND," *IEEE - ICME*, pp. Vols 1-5, Proceedings Pages: 1545-1548, 2006.

[10] Wei-Min Zheng, Zhe-Ming Lu, and Xiu-Na Xu, "A Novel Skin Clustering Method for Face Detection," *IEEE Computer Society*, pp. Vol 3, Proceedings Pages: 166-169, 2006.

[11] Wen-Hsiang Lai and Chang-Tsun Li, "Skin Colour-based Face Detection in Colour Images," *IEEE Computer Society*, pp. paper56 pp 1-6, 2006.

[12] V. Mariatos, K.D. Adaos, and G.P. Alexiou, "Design and Implementation of a Reconfigurable, Embedded Real-Time Face Detection System," *IEEE Computer Society*,

pp. Pages: 65-68, 2007.

[13] Erik Andrén. *LWN*. September 2008. [Online]. lhttp://lwn.net/. [Accessed: October 30, 2009]

[14] Inc. Freescale Semiconductor. "Using the MMA7360L ZSTAR2 Demo Board. Rev 1". 2007. [Online]. www.freescale.com