# Självständigt arbete på grundnivå

*Independent degree project – first cycle*

**Electrical Engineering**

**FPGA implementation of ROI extraction for visual-IR smart cameras**

**Sajjad Zandi Zand**

Mittuniversitetet
MID SWEDEN UNIVERSITY

**Campus Härnösand** Universitetsbacken 1, SE-871 88. **Campus Sundsvall** Holmgatan 10, SE-851 70 Sundsvall.
**Campus Östersund** Kunskapens väg 8, SE-831 25 Östersund.
Phone: +46 (0)771 97 50 00, Fax: +46 (0)771 97 50 01.

Examiner: Dr. Börje Norlin, borje.norlin@miun.se
**Supervisor**: Dr. Muhammad Imran, muhammad.imran@miun.se
**Author**: Sajjad Zandi Zand, saza1005@student.miun.se
**Degree programme**: International Bachelor's Programme in Electronics, 180 credits
**Main field of study**: Electrical Engineering
**Semester, year**: Spring, 2015

# Abstract

Video surveillance systems have been popular as a security tool for years, and the technological development helps monitoring accident-prone areas with the help of digital image processing.

A thermal and a visual camera are being used in the surveillance project. The thermal camera is sensitive to the heat emitted by objects, and it is essential to employ the thermal camera as the visual camera is only useful in the presence of light. These cameras do not provide images of the same resolution. In order to extract the region of interest (ROI) of the visual camera, the images of these cameras need to have the same resolution; therefore the thermal images are processed in order to have the same size as the visual image.

The ROI extraction is needed in order to reduce the data that needs to be transmitted. The region of interest is extracted from the visual image and the required processes are mostly done on the thermal image as it has lower resolution and therefore requires less computational processing. The image taken from the thermal camera is up scaled by using the nearest neighbor algorithm and it is zero-padded to make the resolutions of the two images equal, and then the region of interest is extracted by masking the result with the related converted version of visual image to $YC_bC_r$ color space.

The feeding data for the program is two streams from two separate cameras and according to the need for parallel processing, FPGA was chosen over microcontroller. Furthermore the power consumption and the design is estimated by using Xpower Analyzer with respect to the behavior of Spartan 6 with device number of XC6SLX4, which is 2.8 miliwatts dynamically. In addition, resource utilization is 21% for logic, 67% for block ram, 50% for DSP and 90% for IOs available on the board.

**Keywords**: ROI, nearest neighbour, up scaling, FPGA, VHDL, MATLAB, Xpower Analyzer, $YC_bC_r$.

# Acknowledgements

I would like to thank the people who made this thesis possible. Firstly, I would like to express my sincere gratitude to my supervisor Dr. Muhammad Imran for his support throughout my studies, for his patience, motivation, and immense knowledge. His guidance helped me to carry out the research and writing of this thesis

In addition, I would like to thank my siblings, Sara and Soheil who helped me a lot during my studies and always had my back emotionally.

Last but not least, I would like to thank my parents for their spiritual support when writing this thesis and in my life in general. I could not have asked God for a better gift.

# Table of Contents

# Table of Figures

# List of Tables

# List of Equations

# Terminology

## Acronyms/Abbreviations

| | |
|---|---|
| NN | Nearest Neighbour |
| RGB | Red-Green-Blue |
| ROI | Region of Interest |
| RTL | Register Transfer Level |
| VHDL | VHSIC Hardware Description Language |
| $YC_bC_r$ | Luminance-chrominance |

# 1 Introduction

Video surveillance systems play a vital role in our daily life. The vast possibilities of the systems allow usability in different areas such as airports, healthcare, banks, offices, railways, etc. Currently, due to development of image processing, all traditional analogue solutions are being replaced with digital video surveillance systems, which increases the precision and decreases the required resources. Creating a smart society has been a target for a long time now, and having smart cameras with possibilities of detection and making decisions can be used in areas where surveillance is crucial.

One of the areas where digital surveillance systems can be used is the monitoring of railways for the sake of safety. According to the Swedish Transport Agency, 109 people were killed in 2013 in the rail accidents [1]. The high number of fatalities requires a robust railway surveillance system. Utilizing a wireless smart camera system will provide useful information for the train operator to take the required action. In the case of this project we are going to study how image processing can help to decrease accidents.

The main goal of this thesis is extracting region of interest in the visual image by using the processed data of the thermal image. The required data processing is done on FPGA.

## 1.1 Problem motivation

The wireless camera system in this project contains a thermal and a visual camera. Referring to the goal of the project, moving objects need to be detected. It is essential to utilize a thermal sensor that can detect a living object because using only a visual camera in the system would be limiting, as it is not useful in darkness. Adding a thermal camera solves the problem as a common characteristic of these objects is the emission of infrared energy (heat); a thermal camera is equipped with sensors which are able to detect heat.

The thermal camera in this project captures frames with resolution of 352x240, whereas the images captured by visual camera have a resolution of 1280x1024. Masking the images from these cameras is

possible only when the resolutions are exactly the same, hence it is essential that the thermal image is processed in order to have higher resolution.

The processes of this project need to be done by a processor that can handle parallel processing as the requirement is to handle two streams of visual and thermal efficiently; therefore using microprocessors will cause latency as they handle processes sequentially. According to the real-time requirements of the project any latency needs to be prevented. Therefore, using FPGA for processing is the best choice in order to handle parallel processes and to minimize latency.

## 1.2 Overall aim

The main aim of this project is to extract thermo-radiating objects from a thermal camera (sensor) and extract regions of interest from the corresponding visual image from the same scenery. Processing the thermal image decreases the processing load as it has lower resolution compared to the visual image. The following figure shows the overall structure of modules of the railway safety project, the highlighted modules are a part of this thesis project.

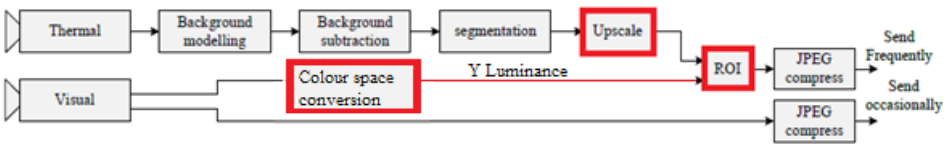


**Figure 1. Outline of the project**

## 1.3 Scope

The camera setup should be able to detect the objects in different lighting and weather conditions. The main focus of this project is to integrate a combination of thermal and visual camera to detect moving objects and to extract the region of interest in the visual image in order to minimize the data transmission. For the sake of extraction, a masking process is required, and in order to perform masking both images need to have same resolution, which means pre-processing is required. The first step is to upscale the thermal image by using nearest neighbour

algorithm. When the upscaling stage is done, the image needs to be zero padded – the borders of the image filled with zeros (black colour) – so that the images have the same size. The next stage is convert the visual image to $YC_bC_r$ color space and extract the luminance component for the masking process. Finally the processed visual and thermal images are masked to extract the region of interest.

## 1.4 Concrete and verifiable goals

The aim of this work is to extract the region of interest in a visual image with the help of a processed thermal image to provide surveillance in areas where it is needed in order to increase security and safety. The following are the tasks of the project.

- Investigate the most suitable algorithm for image scaling
- Implement upscaling
- Implement zero-padding
- Convert visual image to $YC_bC_r$ colour space by investigating the related Xilinx Ip core and extracting luminance
- Develop test bench for modules in order to debug them
- Mask the processed thermal and visual images in MATLAB and VHDL
- Analyse the power consumption and resource utilization by Xilinx tool, Xpower Analyzer.

## 1.5 Outline

This report is organized in different chapters as follows: Chapter 2 gives theoretical information about different terms and algorithms used in this thesis. Chapter 3 describes the methodology and outline of the thesis. Chapter 4 is divided into two sections describing how the points in methodology are implemented in both MATLAB and VHDL simulation. Chapter 5 presents the step by step results of the implementation and compares the visual differences in each stage.

Chapter 6 discusses the problems and limitations and mentions the social and ethical aspects of the project. Chapter 7 brings all the steps of the project together and the achievements are concluded. Chapter 8 suggests future work for better and more efficient results.

## 1.6 Contributions

This thesis is a part of a large surveillance project and most of the work is done by the author based on related research in this area. I would like to mention that the assistance from my supervisor Dr. Muhammad Imran by providing the outline of image reading and writing module has been most appreciated. Furthermore the processed test images were provided by my supervisor and Amiryousef Samaei, my classmate.

# 2 Theory

The camera setup of this system uses a thermal and a visual camera in order to detect moving objects. The raw images from the thermal camera are not the best option for this project as the data need to be transmitted; therefore they need to undergo processing to extract the region of interest. Extraction of ROI means less data to be dealt with in the transmission phase.

Different requirements in terms of resolution means the images need to be scaled using image processing. There are several algorithms that have been used to upscale or downscale images, and we cite a few papers discussing different techniques in this field. In addition to scaling the images, other processes might be required when there is a need to have images with the same resolution. Some of these techniques are briefly described in the following.

## 2.1 Upscaling

Upscaling refers to resizing the original image in order to have a higher resolution. In order to mask the images in this project, the first step is to upscale the thermal image because the resolutions of visual and thermal images are not equal. As the thermal image has lower resolution than the visual, a scaling factor is required in order to upscale the thermal image in both x and y axes. Moreover, an efficient upscaling method needs to be applied to fulfil the requirements of the project. Changing the resolution of an image can be done by using two-dimensional interpolations that reconstruct the set of data regarding a specific algorithm. The outcome of this process is an image with new set of coordinates. The scaling process is anticipated to maintain the contents of the image as far as possible, in addition to providing a good quality [2][3].

## 2.2 Upscaling Algorithms

Upscaling algorithms are chosen because of the requirements of the projects. There are several algorithms available, for instance nearest neighbour, bilinear, bi-cubic, quadratic cubic and Gaussian [2][4]. Some of these methods provide good quality results but need more computational processes. Three most used non-adaptive algorithms are briefly described as follow.

### 2.2.1 Bilinear

Bilinear scaling takes the nearest pixels of the original image and then takes a weighted average of the neighbouring pixels and places the result for the unknown pixel of the upscaled image [5]. The following figure indicates the placement of pixels in this algorithm.

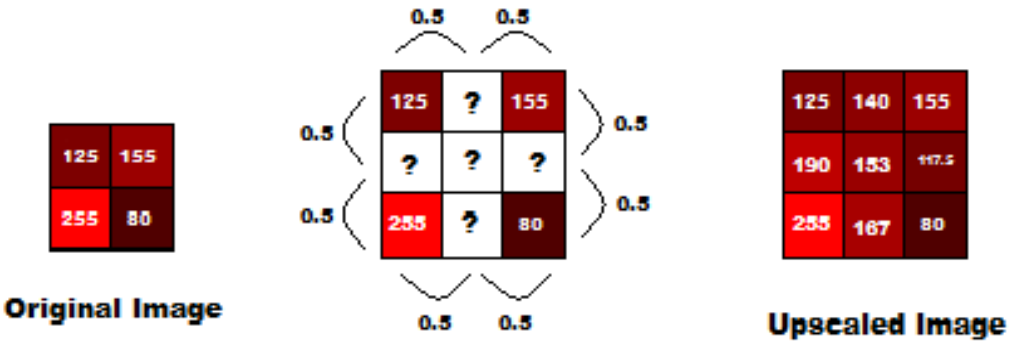


**Figure 2. Bilinear upscaling algorithm**

As the figure above shows, each unknown pixel from the upscaled image is filled with the average of its known neighboring pixels.

### 2.2.2 Bi-cubic

Bi-cubic upscaling is often chosen over other non-adaptive algorithms when speed is not an issue. This algorithm with higher complexity offers a good quality but requires more calculations and computational processes. Bi-cubic takes a further step compared to bilinear by taking the nearest 4x4 neighborhood of known pixels. As the distance to the unknown pixels varies, closer pixels get more weight in the calculations [6].

### 2.2.3 Nearest Neighbour

Nearest neighbor, computationally, has the least complexity of the mentioned methods and offers the highest speed of operation. Instead of calculating the mean value with weighting criteria, this algorithm takes the nearest neighboring pixel of the original image and places it in the upscaled image [7][8].

The following figure shows the placement of pixels in this method.

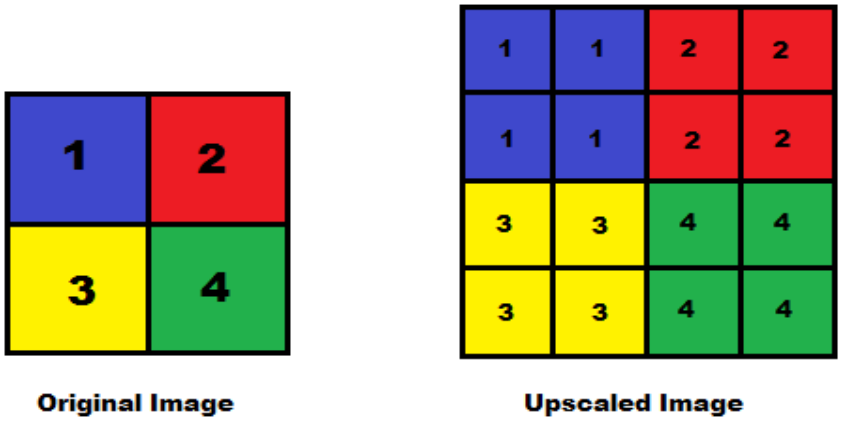


**Figure 3. Nearest neighbor upscaling algorithm**

As the figure above indicates, unknown pixels of the upscaled image get the value (color) of its nearest neighbor. In the example, the 2x2 image is upscaled with the factor 2x2, which creates a 4x4 image. Multiplying the original image by the upscaling factor suggests that each pixel is going to be repeated 4 times in the upscaled image. This method generates objects with mosaic quality which decreases the quality of the image. Although the excellence of the upscaled image is lower than the other two, the speed of performance is considerably higher than the other methods and it is less complex to implement. This method is the best choice in applications where the quality of the image is not the first priority and the speed of execution is more important.

### 2.2.4 Comparison

As mentioned earlier in this chapter, each method has advantages and disadvantages, therefore one method cannot be introduced as the best possible algorithm. Hence, the method is chosen to meet requirements of the application. Nevertheless, due to the unique characteristics and wide application of image scaling, a separate study of their evaluation methods is essential [9]. This section compares different scaling algorithms in some functional characteristics.

The nearest neighbor scaling algorithm has the lowest complexity and the advantage of being fast. However it may bring considerable distortion and come out with mosaic and saw-tooth quality [10].

The bilinear scaling method requires more calculation than nearest neighbor, making it more complex. It creates satisfactory results and has the ability of low-pass filtering, meaning that the high frequency component of the image is faded. Having the ability of creating continuous results, makes this algorithm a better choice than nearest neighbor in terms of visual effects. Although it has better performance in terms of visual aspects, the speed of operation is slightly lower compared to nearest neighbor.

Among the three mentioned algorithms, bi-cubic interpolation offers the best quality of image and consequently requires more calculations and computation processes. The algorithm is perfect for high-level image processing applications such as Photoshop, Avid, etc. [11].

The following graph presents the average running time (seconds) of these algorithms, and as it indicates nearest neighbor is the fastest option of all.

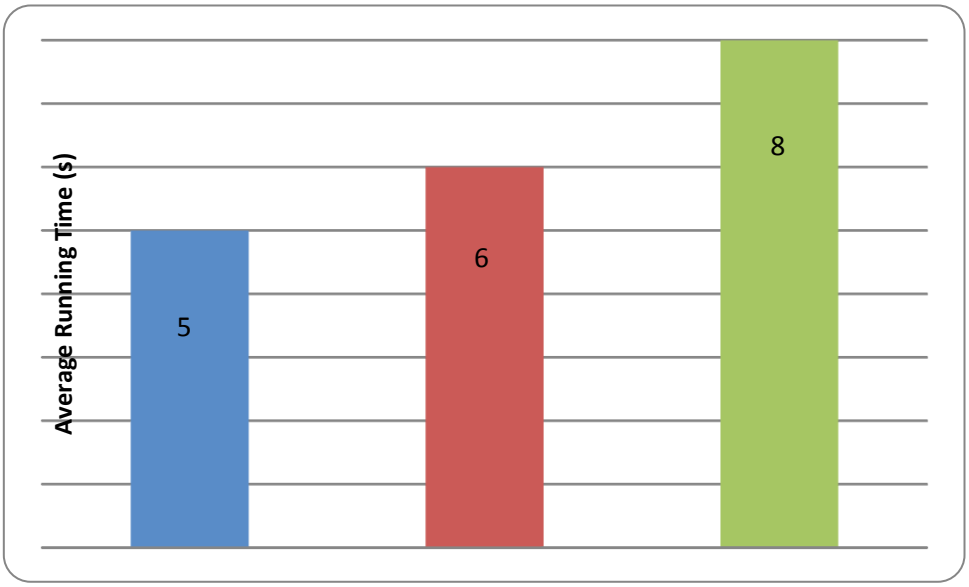


**Figure 4. Running time of different algorithms [11]**

The following table compares the algorithm from different aspects.

| Method | Subjective Feelings | Image Contour | Processing Time (seconds) |
|---|---|---|---|
| **Nearest Neighbor** | obvious mosaic phenomenon | Not clear | 5 |
| **Bilinear** | blur, not sharp | not clear, serrate phenomenon | 6 |
| **Bi-cubic** | fuzzy, sharper | serrate phenomenon has improved | 8 |

**Table 1. Comparison of different upscaling algorithms [11]**

The following fiugre, is a visual comparison of the three methods applied to a similar image.

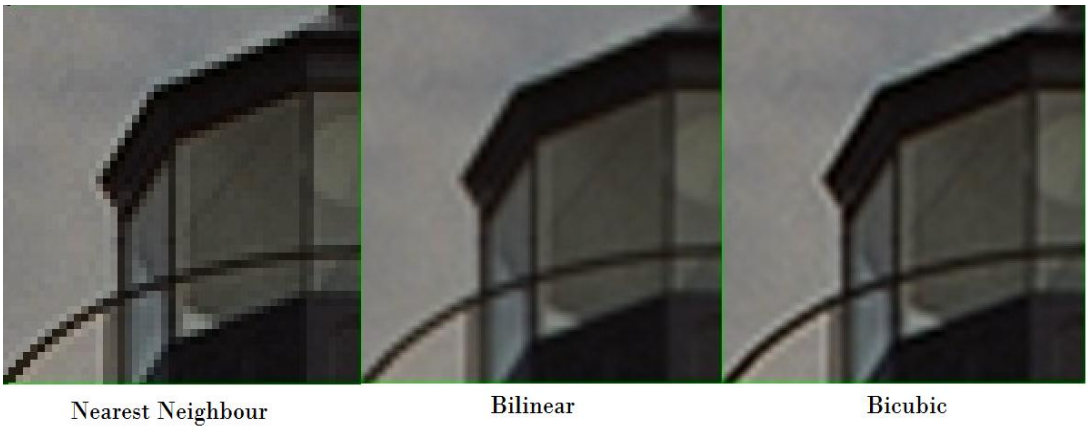


**Figure 5. Visual comparison of different algorithms [12]**

## 2.3 Zero-Padding

Zero-padding refers to extending a signal (spectrum) with zeros. In terms of image processing, performing zero-padding helps us to create borders for the images by filling margins with zeros. The need of zero-padding arises when the input images have been taken with cameras with different focal lengths. Dissimilar focal lengths cause the field of view to be different in the images. The following figure indicates how the focal length is related to field of view, a smaller focal length results in a larger field of view.

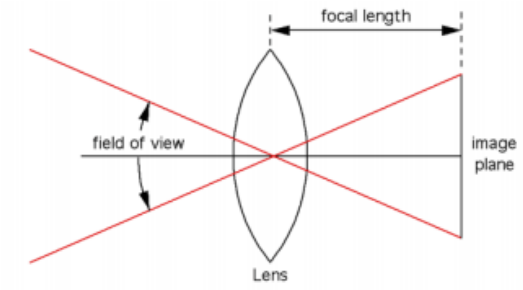


**Figure 6. Relation between focal length and field of view**

## 2.4 YCbCr

YCbCr is a colour space system where Y stands for *luma* and CbCr stands for blue-difference and red-difference *chroma* components. Bitmap images use red, green and blue channels directly to show colour images. According to medical research on the human eye, it has been proven that the human eye has different sensitivity to colour and brightness [13].

These facts suggest converting RGB to YCbCr. As the medical investigations on the human eye suggests, the rods, 120 million in number, are much more sensitive than the cones, which are around 6-7 million in number [13]. The rods in the human eye, which carry out the most tasks in human vision, have a sensitivity to brightness. The cones provide the eye's sensitivity to colours, and they are located close to a central region called the macula.

The luminance channel appears to be very similar to the grayscale version of the original image. $C_b$ is strong where the blue colour is dominant in the image, for instance when the image is taken from the sky. $C_r$ is strong when the image is taken from places where the reddish colours are dominanat and both $C_b$ and $C_r$ factors are weak where the green colour is leading. [14]

The following figure shows the colour difference in different channels and also the difference between RGB and $YC_bC_r$.

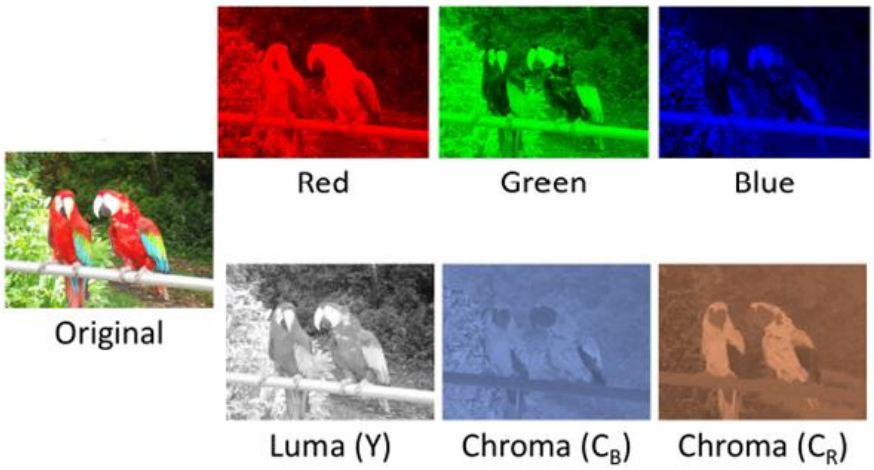


**Figure 7. Difference between RGB and YcbCr [15]**

## 2.5 Processing Platforms

Any application in an embedded integrated system attempts to minimize four factors simultaneously. The number of transistors, the number of clock cycles required, the complexity of application development and nonrecurring engineering costs are the factors that should be reviewed before choosing a processing platform for a certain application.

Different processing units can be used in order to achieve the goals of the project. The following is a review of the strengths and weaknesses of MCU, FPGA, DSP and ASIC.

Microcontrollers (MCU) are devices for information processing and control which can be deployed on a wide range of applications. The development is limited to software only and clock cycle optimization is determined by the optimization of the code. In addition, the code footprint affects the number of transistors needed. It normally uses transistors and clock cycles efficiently but not at optimal level. The data processing takes place in a sequential manner.

Digital signal processors (DSP) optimize the number of transistors and clock cycles required by hard-wiring the basic functions of signal processing algorithms. The complexity of code is lower than for microcontrollers and many MCUs include basic DSP operations in their introduction set.

Field programmable gate arrays (FPGA) limit the effort of development in the coding phase. The power consumption is not optimal and the optimization of clock cycles is limited. The processing takes place in a parallel manner; it is ideal where processes need to be carried out simultaneously.

ASIC is a custom-designed processor for particular applications that contains a mixture of MCU or DSP cores. The customizability allows the

optimization to be at the highest level for transistors and clock cycles but requires more development time and is more expensive [15].

Choosing processing platform for any application is an engineering compromise. Generally it depends on several factors and no single technology is ideal, therefore a combination of different processing units results in an optimized cost and power consumption. In the case of this project, FPGA was chosen due to its ability to handle parallel processes.

# 3 Methodology/Model

The following diagram depicts the top-to-down stages required for the proposed application. Because of the aim of this project, the two images need to be masked in order to extract the region of interest given that the segmented image is provided to the project. When it comes to the different resolutions of the images, the thermal image needs to be upscaled and zero-padded in order to have the same resolution as the visual image. The stages are done using VHDL programming and this chapter discusses each stage in detail.

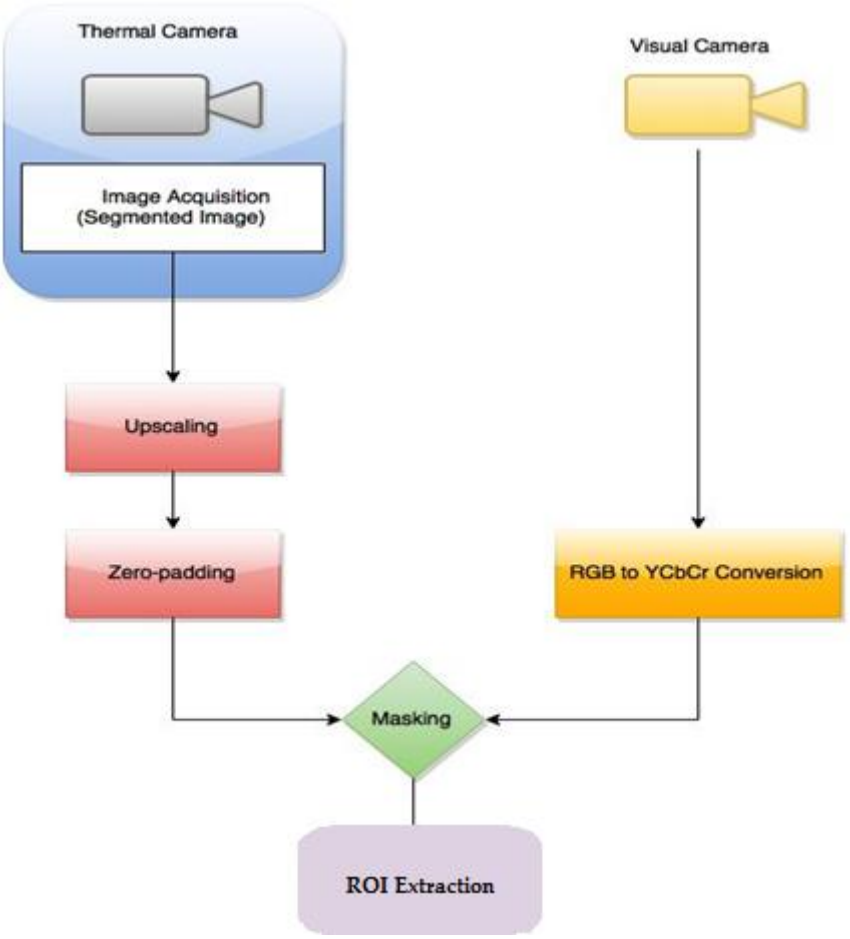


**Figure 8. System overview**

## 3.1 Image Acquisition

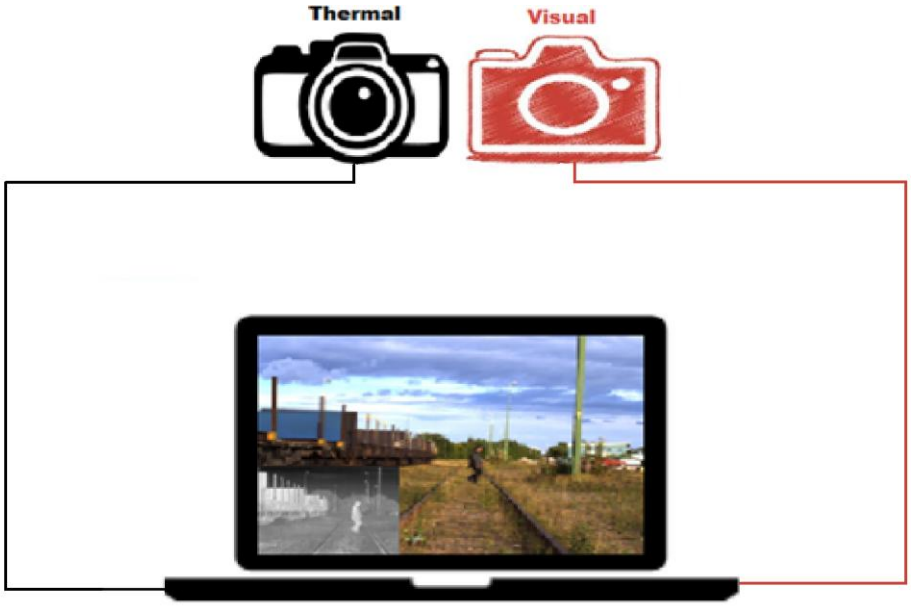


**Figure 9. Image samples of the cameras and camera setup of the system**

The camera setup in this project contains two cameras installed on a plate. The thermal camera uses a 19 mm lens with uncoiled VOX micro bolometer sensor working in the spectral band of 8-14 μm with a resolution of 320x240. The visual camera in this project has a focal length of 12 mm.

According to the description of the project, the image from the thermal camera is processed through segmentation module. This project deals with the test image that has gone through a segmentation module which changes the grayscale image to binary.

**Figure 10. Segmented thermal image [16]**

## 3.2 Upscaling

In terms of application requirements, the speed of implementation plays a vital role. Hence the chosen method for upscaling the image is nearest neighbour. This algorithm offers the highest speed comparing to the other available methods, and in this case it also provides satisfactory quality of image as the image under process is binary.

### 3.2.1 Nearest Neighbor

Although better quality can be achieved with other methods, in this case nearest neighbour is a good compromise between speed and quality. In order to implement nearest neighbour, the first step is to consider what scaling factor is required. The following figure shows the calculation process of the scaling factor for the nearest neighbour algorithm.

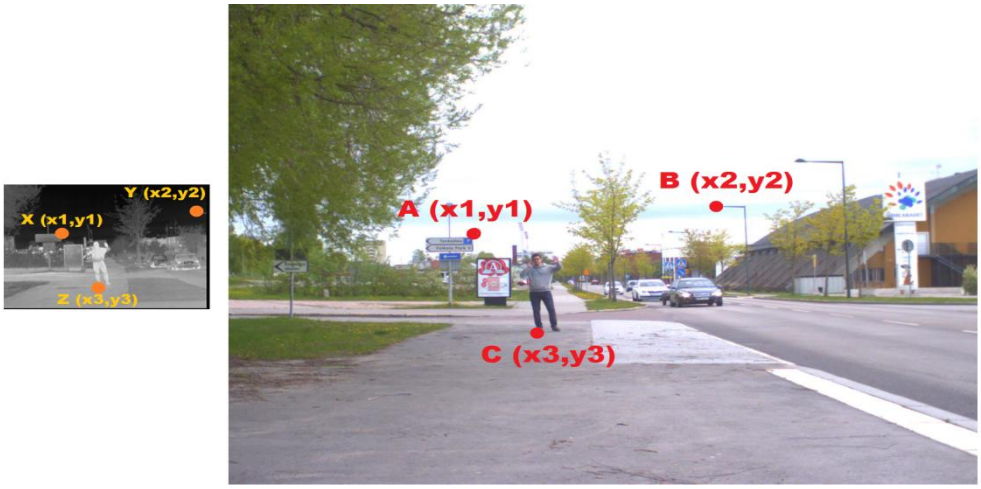


**Figure 11. Sample images from cameras for upscaling calculations**

As the figure above indicates, the highlighted points are used to calculate the areas of zero-padding and also the upscaling factor.

$$\boldsymbol{Horizontal\ scaling\ factor} = \frac{Y_{(x2)} - X_{(x1)}}{B_{(x2)} - A_{(x1)}}$$

**Equation 1. Horizontal scaling factor**

$$\boldsymbol{Vertical\ scaling\ factor} = \frac{Z_{(y3)} - X_{(y1)}}{C_{(y3)} - A_{(y1)}}$$

**Equation 2. Vertical scaling factor**

The number of rows and columns of the upscaled image is calculated according to Eq.3 and Eq.4.

$$Upscaled\ Columns = Horizontal\ scaling\ factor \times Columns\_original$$

**Equation 3. Number of columns in upscaled image**

$$Upscaled\ Rows = Vertical\ scaling\ factor \times Rows_{original}$$

**Equation 4. Number of rows in the upscaled image**

## 3.3 Zero-padding

The upscaling factor is calculated by using the equations above, but the number of blank columns and rows still needs to be calculated in order to perform zero-padding.

The idea is to have the same resolutions in both images, but as the thermal and visual images show, the cameras do not have the same field of view (Figure 11. Sample images from cameras for upscaling calculations). Therefore the upscaled image needs to be zero padded to cover the non-important areas that visual camera is taking. Hence the

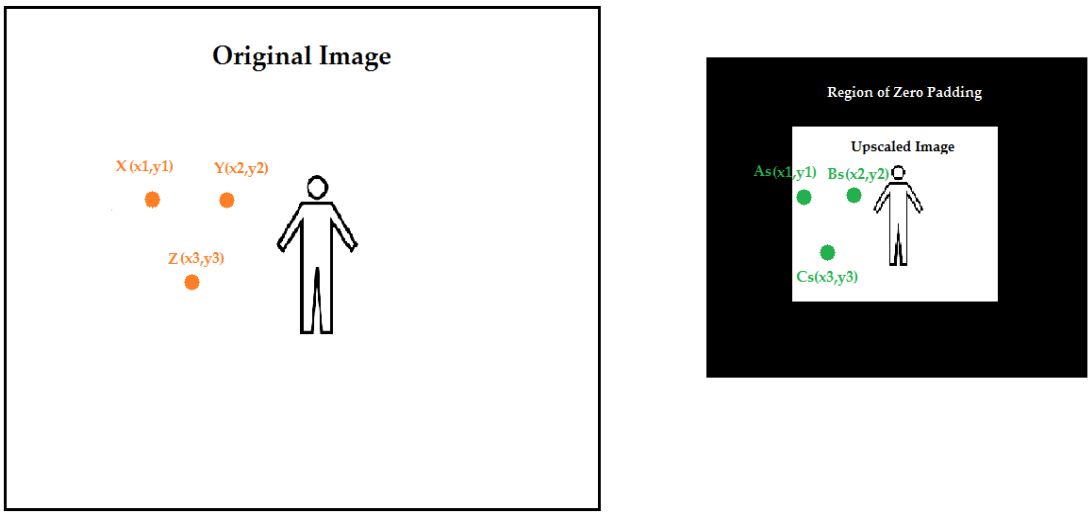


**Figure 12. Zero-padding structure**

regions out of interest need to be filled with black colour so that the images will have the same resolution. The following equations indicate how the region for zero-padding is calculated.

$$Blank\ Columns = Y_{x2} - Bs_{x2}$$

**Equation 5. Number of blank columns**

$$Blank\ Rows = Y_{y2} - Bs_{y2}$$

**Equation 6. Number of blank rows**

## 3.4 $YC_bC_r$

Designers of video systems frequently need to perform conversions through different colour spaces. Xilinx RGB to $YC_bC_r$ Color-space Converter LogiCORE supports 5 formats and 3 range standards. The implementation consists of an abstracted 3*3 coefficient matrix. This coefficient uses 4 multipliers which apply to the coefficients regarding inter-relations of RGB to $YC_bC_r$ [18].

$YC_bC_r$ is more suitable for the human visual system compared to RGB. For the same reason, in the JPEG process, the compression takes place more efficiently in the $YC_bC_r$ plane. Hence, during decoding, the features are directly extracted from the luminance and chrominance components [19].

The following equation shows the conversion from RGB to $YC_bC_r$.

$$\begin{bmatrix} y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.299 & -0.587 & 0.866 \\ 0.701 & -0.587 & -0.114 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$y = (0.299R) + (0.587G) + (0.114B)$$

$$Cb = (-0.299R) + (-0.587G) + (0.866B)$$

$$\boldsymbol{Cr = (0.701R) + (-0.587G) + (-0.114B)}$$

**Equation 7. RGB to $YC_bC_r$ conversion [19][20]**

# 4 Implementation

According to the objectives of this project a high level model for nearest neighboring and ROI extraction in MATLAB needs to be implemented. In addition, the algorithm needs to be designed and implemented in VHDL. According to the description in Chapter 3, the implementation in MATLAB and VHDL require different strategies.

## 4.1 High-level Analysis Performed in MATLAB

Upscaling in MATLAB leads to less development complexity as it takes the whole image as a matrix so that all pixels can be accessed simultaneously. Regarding the tasks of the project, the ROI extraction needs to be done in MATLAB. This is why the first step is to read the visual image and the prepared upscaled image, and then combine these images using logical AND operation.

## 4.2 Hardware Implementation

For implementation on hardware the algorithms are modeled in RTL by using VHDL. The design in VHDL requires more development time than MATLAB, as VHDL deals with images pixel by pixel. As mentioned in Chapter 3, the calculated upscaling factors are decimal numbers, 1.93 for vertical and 1.76 for horizontal factors. According to the description of the algorithm, implementing nearest neighbor is basically repeating the pixels based on the upscaling factors. As one pixel cannot be repeated 1.76 and 1.93 times in hardware, the factors are rounded up to 2. Due to the type of data, the black and white image, and also the application of the project, the rounding up does not involve a significant change.

Other than the modules in the implementation phase, a test bench is designed to feed images to the program and also print the result on screen, which can be described as follows.

### 4.2.1 Upscale

The upscale module consists of a sub-module called memory. In order to perform the nearest neighbor algorithm, each pixel needs to be stored in the memory and read from the memory according to the upscaling factors. In this case, as described in this chapter, the factors are rounded up to two. Therefore each entry of the memory is read two times and when one row is done, the whole row is repeated.

In order to implement the algorithm in VHDL, several counters are used. A writing counter is used to address the data coming into the memory, a reading counter addresses the data from the memory to the upscaled image. The pixel counter is used to make sure that each pixel is repeated twice in a row and finally the line counter is used to ensure each row is repeated twice.

The zero padding and masking of the images have been done in this module as well. In order to perform zero-padding and masking, the number of columns and rows need to be tracked, therefore, as the counter of columns and rows reach a certain number, the zero-padding needs to be stopped and the data stored in the memory is read and fed to the masking process. This means that when the number of columns and rows reaches 172 for columns and 212 for rows, the counter for reading from the memory starts and runs until the columns and rows are out of the zero-padding region.

The following figure shows the schematic of the upscaling module including all inputs and outputs of the module.

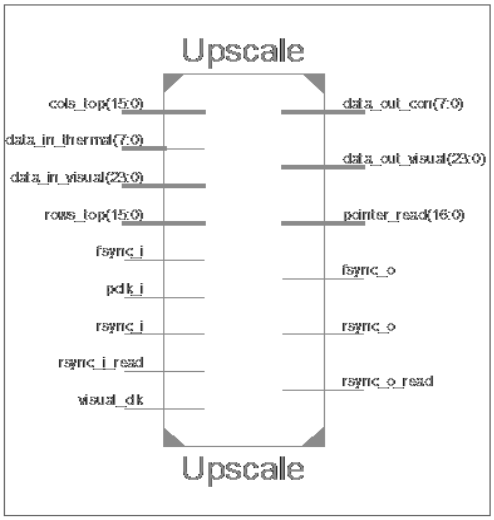


**Figure 13. Upscaling module**

#### 4.2.1.1 Memory

The memory in this project consists of 84,480 entries as the thermal image is 352x240; 84480 pixels are written to the memory and read according to the address of the memory. When a pixel is written to the memory, the writing counter of memory address increments by one. In the reading phase, when the data is read from the memory, the reading counter increments when the entry is read for two clock cycles. In order to ensure that each row repeats twice, the reading counter is subtracted by 352 at the end of each line until the row is repeated twice. For instance when the reading address reaches entry number 704 for the first time, the counter is subtracted by 352 and starts over again. After the subtraction, when the counter reaches 704 again, it continues to the next memory cell 705.

The following figure shows the RTL schematic of the memory including all inputs and output of the module.

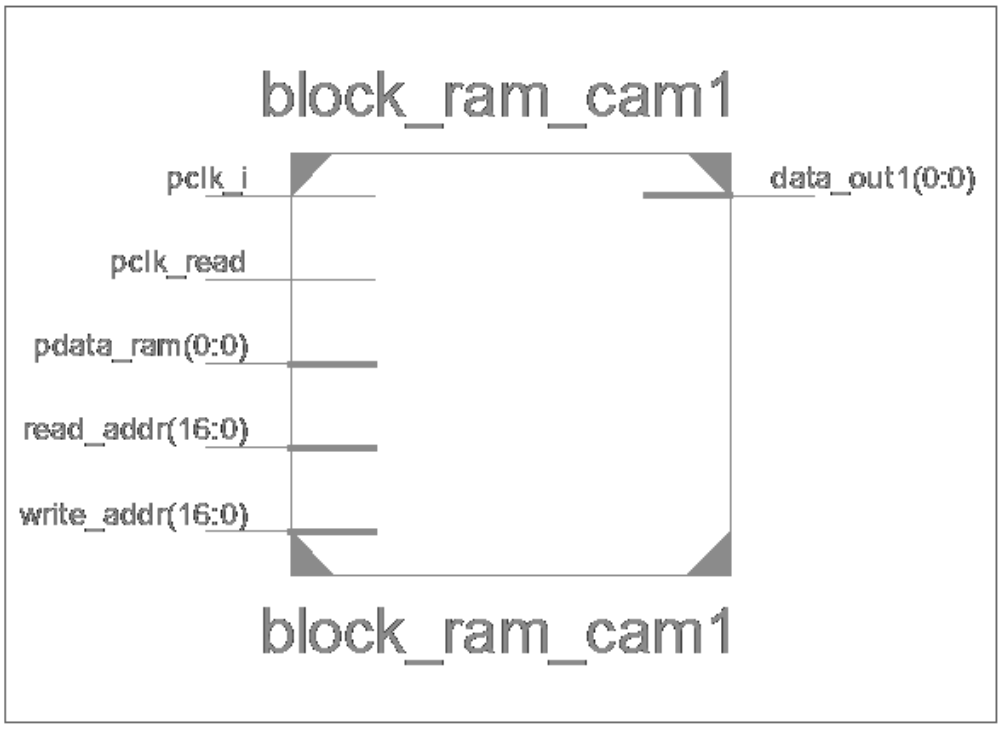


**Figure 14. Block memory module**

According to the figure above, two different clocks are used in this module, first clock *Pclk_i* is generated by the thermal camera and is responsible for writing data into the memory. The other clock *Pclk_read* is generated by the visual camera and it is responsible for reading data from the memory.

The following figure indicates the behavior of the memory when it is used for writing and reading data. The pixel value in this case is either 0 or 255, therefore storing one bit of the value can solve the problem, for example if the value is zero all 8 bits are 0s and if the value is 255, all bits are 1. Hence, by storing only one bit, the amount of memory that is required can be minimized to 1/8.

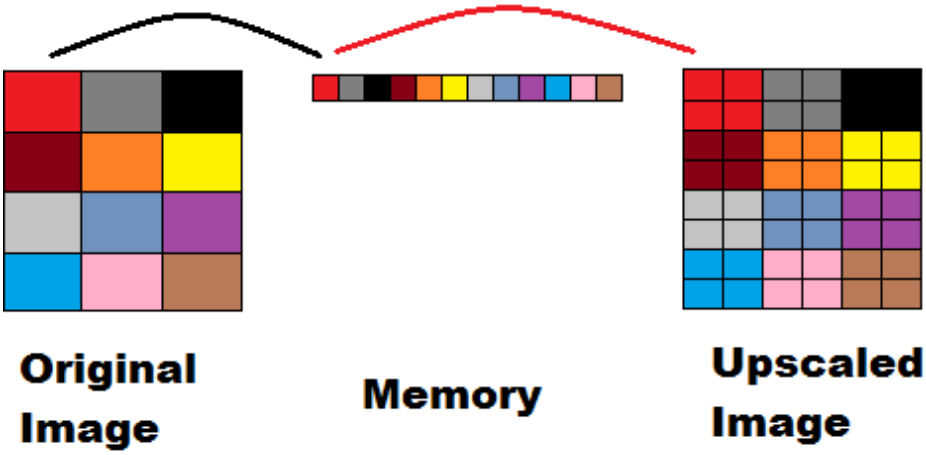


**Figure 15. Block memory behaviour**

As the figure above shows, pixels are written to the memory from the original image, and then each pixel is repeated four times in the upscaled image.

The process of reading from the memory (printing the image on screen) runs with the clock of the visual camera which is adjustable and faster than the thermal camera clock. In order to prevent overwriting the memory cells that have not been read yet, the writing counter stops when the memory is full and starts again when the reading counter reaches the last entry of the memory.

According to Figure 16. Behavioural simulation of block memory the memory entry with an index of 17074 is filled with the value 1 and it has taken one thermal clock cycle for this operation to be completed. However, as the following figure shows, when the reading counter reaches the index 17074, the stored data is read and as mentioned each entry repeats for two pixels (visual clock cycles). When the counter reaches the end of the row, the counter starts again and repeats the whole row one more time.

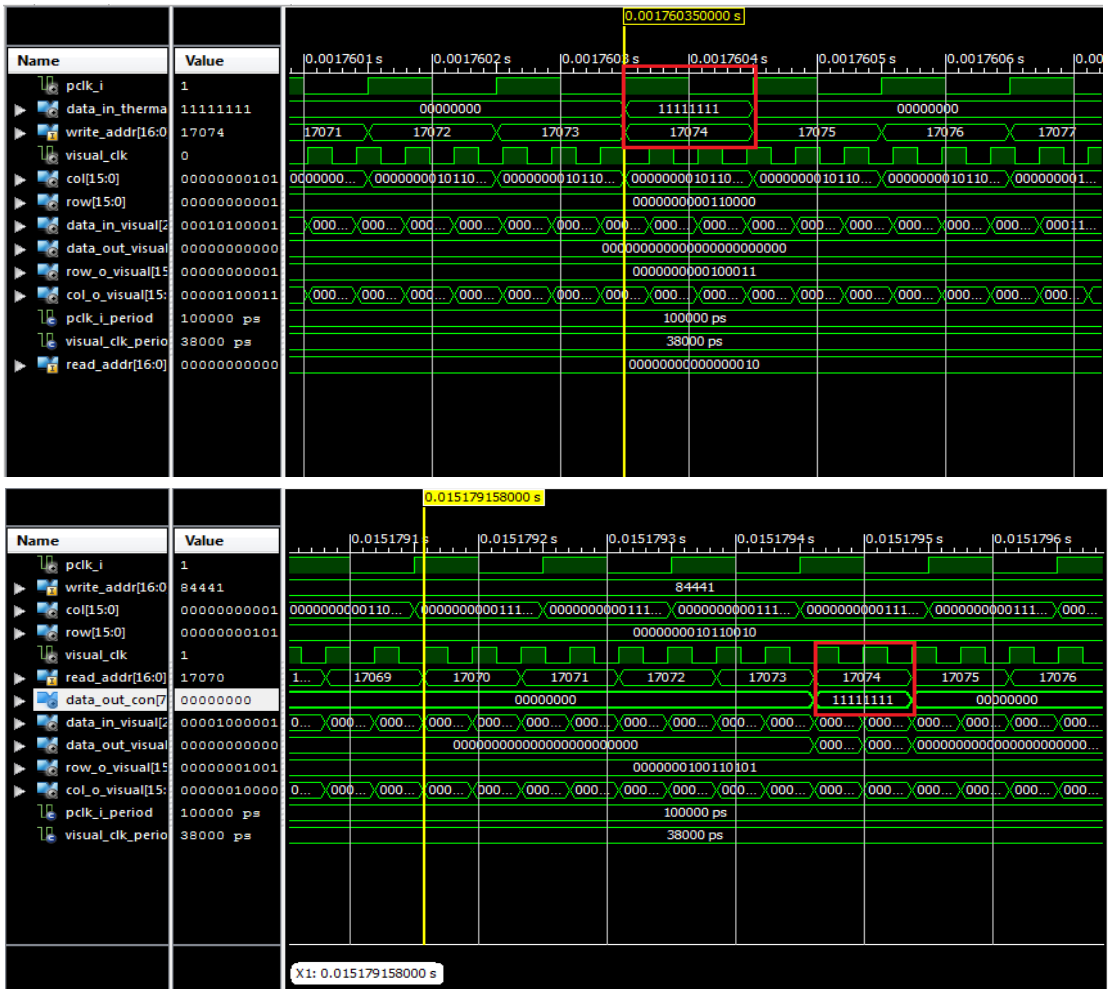


**Figure 16. Behavioural simulation of block memory**

### 4.2.2 Zero-padding

The zero-padding process is also done in this module. The program reads the original image and simultaneously starts writing the upscaled image. As mentioned earlier, the borders of the upscaled image needs to be filled with zeros, therefore, when the image writing process has reached the region where the zero-padding is done, the reader counter starts and data is read from the memory. The following figure shows the

simulation of memory, where the reader counter starts and where the writing counter starts. In order to prevent storing unknown data an extra entry has been created in the memory with an index of 84481 and whenever the data is unknown, the data is stored at this address and the counter reader will never reach there.

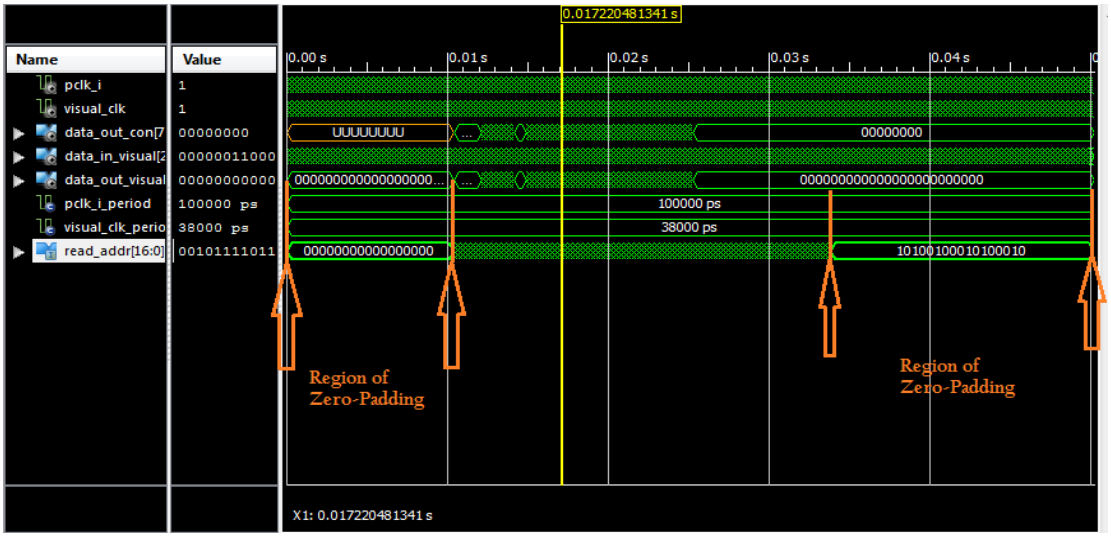


**Figure 17. Behavioural simulation of the zero-padding process**

When the implementation starts, the program starts reading the image and writing the upscaled image. According to the description above, the first rows of the image are filled with zeros, hence, when the image reaches the point where the upscaled image needs to be read from the memory, the writing process on the memory is already finished. In addition to the rows on the top and bottom of the image that are filled with zeros, on each row a certain number of columns are provided with zeros on both the right and left side of the upscaled image.

### 4.2.3 $YC_bC_r$

According to the project outline presented in Chapter 1 (Figure 1. Outline of the project) the visual image needs to be converted to $YC_bC_r$ colour space and then the luminance component needs to be dragged out. The other two components are passed to an averaging module, which is not part of this project.

The conversion of RGB to $YC_bC_r$ is done by using the related Xilinx RGB to $YC_bC_r$ Colour Space Conversion LogiCORE™. The colour space is used by generating the core and choosing the options respectively. According to the information provided in the methodology chapter, the input data from the visual camera is 24-bit RGB, which needs to be converted to luminance-chrominance colour space. Therefore each parameter consists of 8 bits, which needs to be specified while generating the core. Basically, the component gets the output data from the upscale module and runs with respect to row and frame synchronization signals, meaning that when the frame and row synchronization signals are high, the conversation is allowed. The following figure shows the RTL schematic of the conversion module in which the mentioned IP core is added.

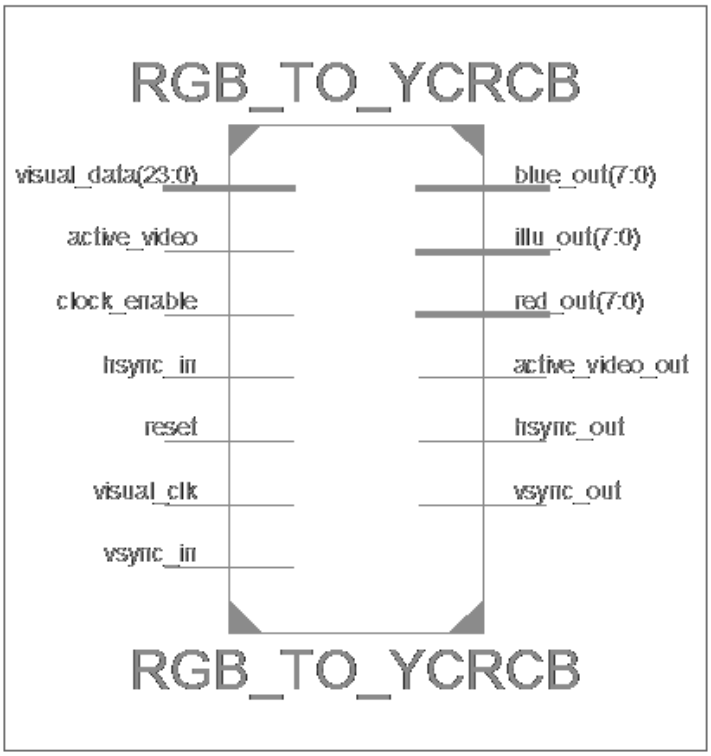


**Figure 18. RGB to $YC_bC_r$ module**

The following figure shows the behavioural simulation of the $YC_bC_r$ module. As can be observed in the figure, the conversion of black colour in RGB (0,0,0) has been converted to YCbCr (128,128,16). The code inside the core is not accessible but the functionality can be monitored by using test bench and simulation. In this design Y (luminance) is written on the result image.

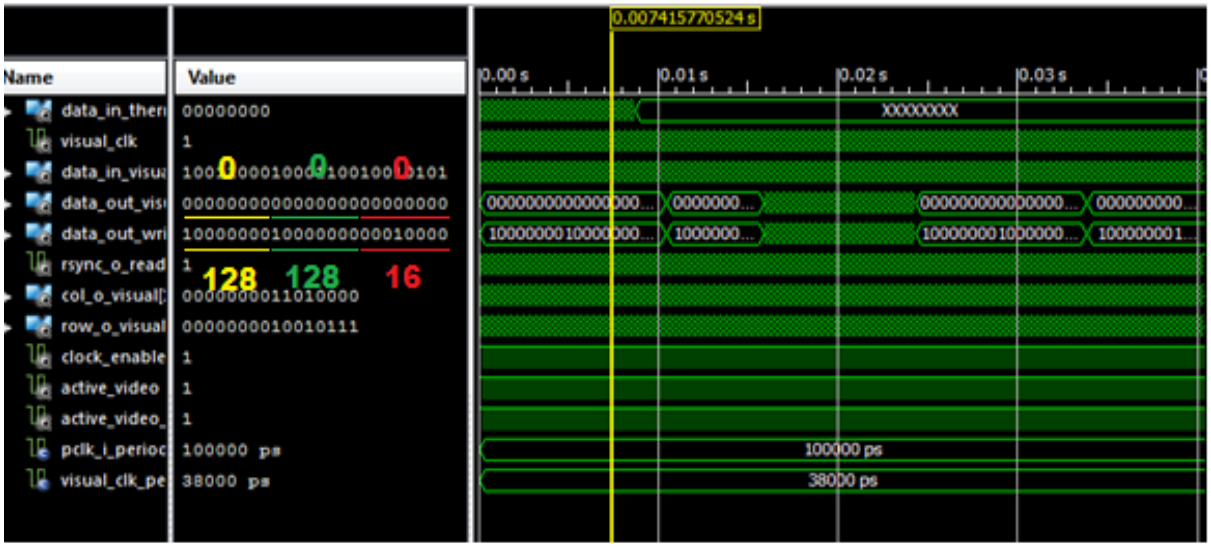


**Figure 19. Behavioral simulation of the $YC_bC_r$ module**

Figure 20 depicts the setting configuration of the colour conversion core. The coefficients of red and blue can be chosen in the range of 0 to $0.9\overline{}$ and the green colour's coefficient of conversion is calculated according to the value of red and blue. After choosing the name of the component and setting the desired configuration, the core is generated and instantiated in the design.

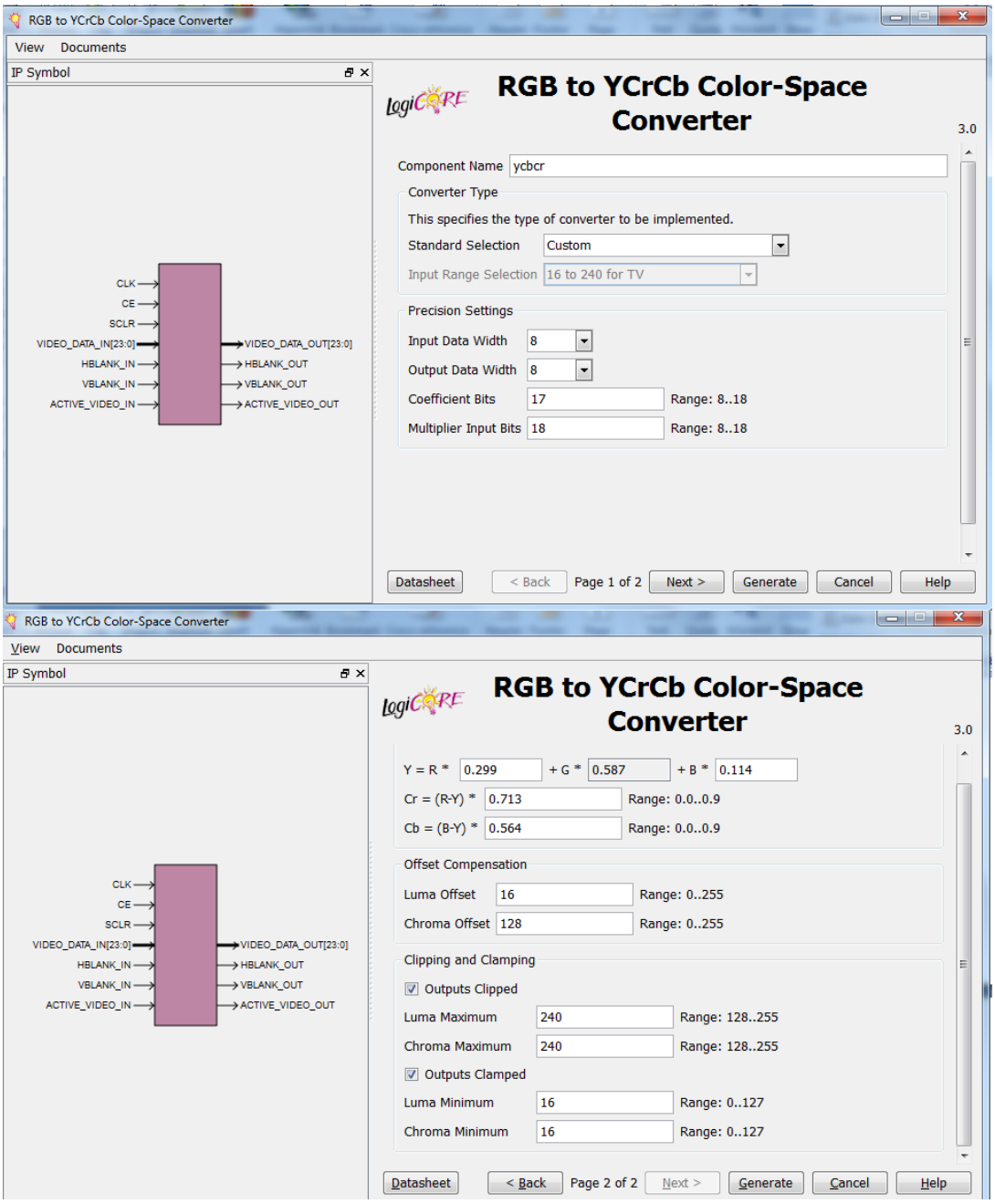


**Figure 20. RGB to $YC_bC_r$ configuration setting**

### 4.2.4 Masking

While the image is being made, each pixel of the upscaled and zero-padded thermal image is combined with the visual image by performing logical AND operation between the pixels. According to the description of the visual camera, each visual pixel is a 24-bit number, which determines the colour of that particular pixel. However, the thermal pixels can only be 0 or 255; therefore each bit of the visual pixel is masked with the thermal pixels. Hence the final result for each pixel is either 0 or the colour of the visual pixel. In this case the object is the only white region of the image, which results in a black background where the object has its actual visual colour. The following figure shows how visual and thermal pixels are operated with respect to each other.

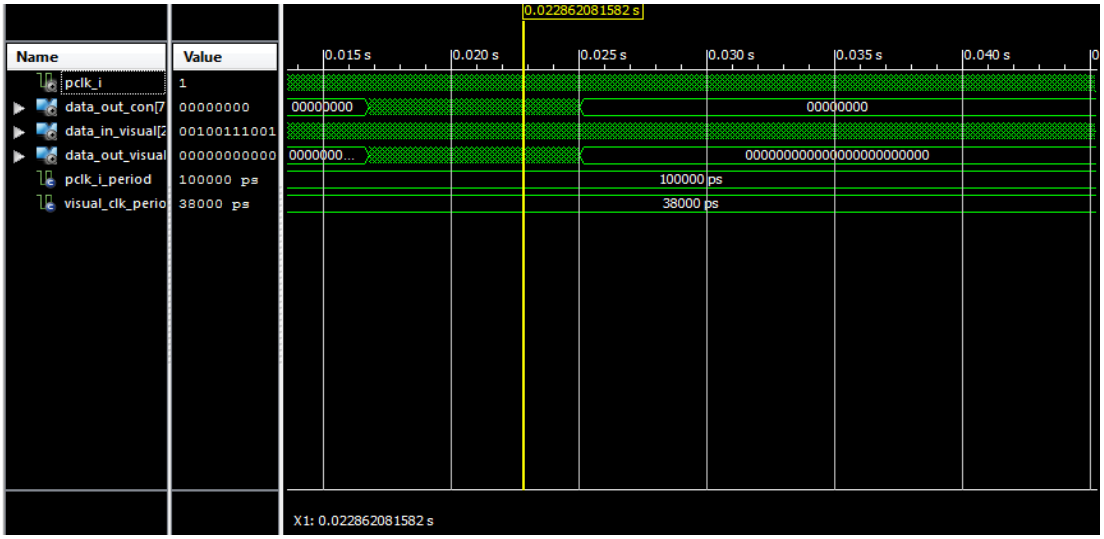


**Figure 21. Behavioural simulation of the masking process**

As the figure above shows, the data from the memory (*data_out_con*) is an 8-bit number. The AND operation is done between this number and data coming from visual image (*data_in_visual*), which results in the final result (*data_out_visual*). According to the figure, whenever the data from thermal image is 0, the final result is also zero and that causes the black colour on the screen.

### 4.2.5 Development of Test Bench

The test bench of this project consists of a module for feeding images to the program and writing the results on the result image. In addition, two clocks are required in this project, which are generated in this module. The clocks have different speeds and sources; one should be from the thermal camera and one from the visual camera. These clock generations are simulated in the test bench module where the thermal clock *pclk_i* has a frequency of 10 MHz and the *visual_clk* has a frequency of 26 MHz.

The image feeder module simulates the behaviour of the camera, by sending the content of the image pixel by pixel to the program. In addition to tracking the number of the pixels sent, row and frame synchronization can be generated in the same way that the camera generates the signals.

Basically, this module reads an image and sends the pixels one by one on each clock cycle to other VHDL modules. Moreover, on each clock cycle the result is passed to this module in order to write it on the image.

According to the description of the project, the modules of VHDL expect one image from the thermal camera and one image from the visual camera. Therefore in the image feeding module, two images need to be sent to the implementation modules. Thus one sample segmented image of (Figure 10. Segmented thermal image) the thermal camera and the related visual image are read and the pixel values are sent to the related modules.

The difference between the real cameras and the simulated module is that images from camera are raw data, and with the help of row and frame signals, the image or the stream is constructed. However, when images are fed to the program using the test bench, they have a header file which contains information about the format of the image. Each format of image contains a certain number of bytes and these header files need to be skipped in order to reach the actual data. In the case of this project, we are dealing with one 8-bit grayscale image and one 24-bit colour image. The header information of the grayscale image occupies 1078 bytes, whereas 54 bytes are occupied by header information in the colour image [21]. In order to read the images, information about the number of columns and rows is essential. This

information is stored in byte number 18 for columns and byte number 22 for rows. By reading these bytes the number of columns and rows become known, therefore the row and frame synchronization can be generated.

# 5 Results

The results of this project are divided into two sections describing the attained results in MATLAB and step by step results of the simulation using VHDL programming. In addition, the power consumption is estimated by using Xpower Analyser which will be discussed in the following.

## 5.1 High Level Analysis

The implementation in MATLAB is to mask the images where the upscaled image of thermal camera is provided. The following image shows the provided images and the masked final result. The provided images have a resolution of 1280x1024, and the task was to mask the visual image with the thermal image.

| a | b |
|---|---|
| c | |

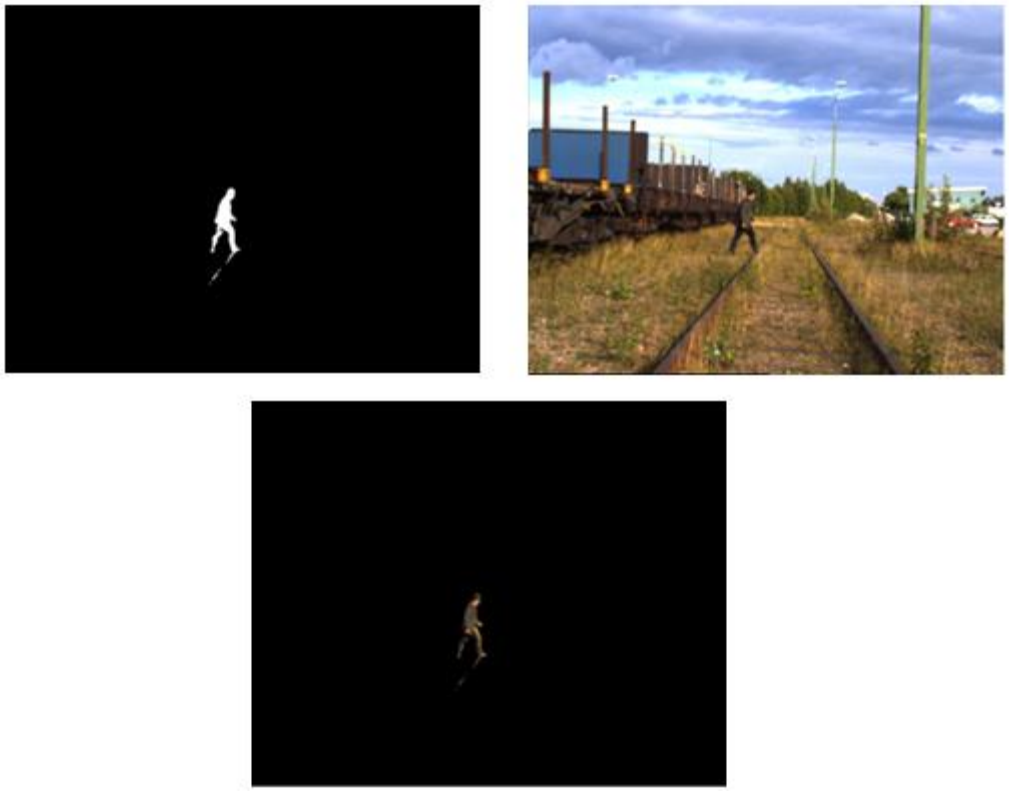

**Figure 22. Results from MATLAB. a) Provided upscaled segmented image. b) Provided visual image. c) Masked image.**

## 5.2 Hardware Implementation

The implementation of the code in VHDL results in two images, RGB and YCbCr. This chapter introduces the step by step results of each stage and how the input image is processed through the program. All following results are made by simulating a behavioural model using Isim. The following image is the provided image which has been taken from the thermal camera and processed in order to be black and white.

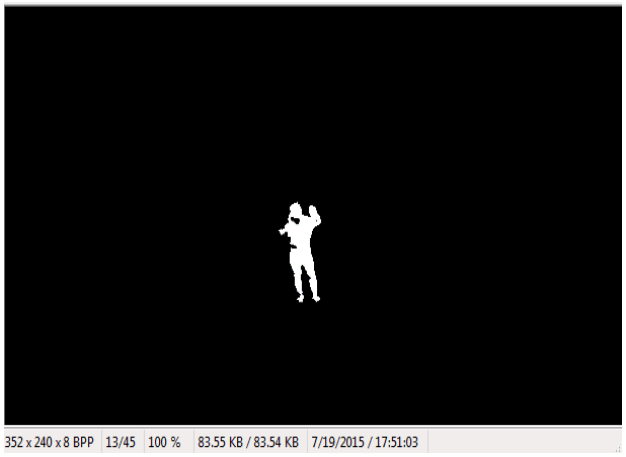


**Figure 23. Original segmented image with a resolution of 352x240**

### 5.2.1 Upscaling

The first step was to upscale the provided segmented thermal image with a resolution of 352x240 by a factor of 2x2, which results in an image with a resolution of 704x480 and an 8-bit colour depth. The following is the result of the upscaling process without zero padding.

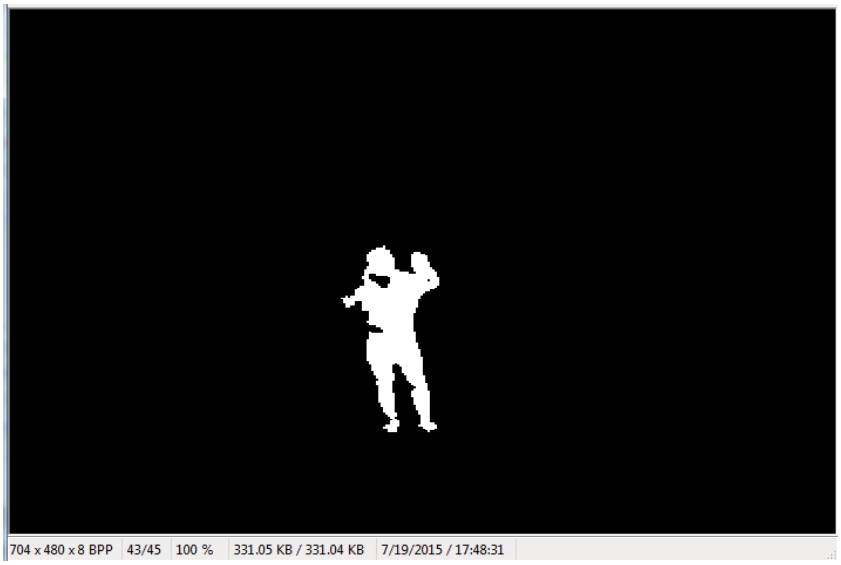


**Figure 24. Upscaled image with a resolution of 704x480**

### 5.2.2 Zero-padding

The image above is passed to the zero-padding process in order to create an image with a resolution of 1280x1024. According to the given resolutions, the difference between the images is filled with zeros, as in the visual image the region is out of our scope of interest. The following figure is the result of zero-padding of the upscaled image. For clarification, in this section, the image is filled with blue colour, to make the zero-padding area easier to see.


**Figure 25. Zero-padded image**

### 5.2.3 YCbCr

The visual image goes through the RGB to $YC_bC_r$ and the RGB pixel values change to $YC_bC_r$ values according to the mentioned formula in Equation 7. RGB to YCbCr conversion [19]. The following figures are the converted version of the visual image to $YC_bC_r$ and only Y.


**Figure 26. Conversion to $YC_bC_r$**


**Figure 27. Y Component**

### 5.2.4 Masking

After performing zero-padding, the image is masked with the related converted visual image taken from the YCbCr module. The following images are the zero-padded and the converted image to Y (luminance) of the visual image that enter the masking process.

a b


**Figure 28. Provided images. a) Converted Y component image. b) Upscaled and zero-padded image**

The following images are the result of the masking process, which is the result of the AND operation between pixels of the upscaled and original images.

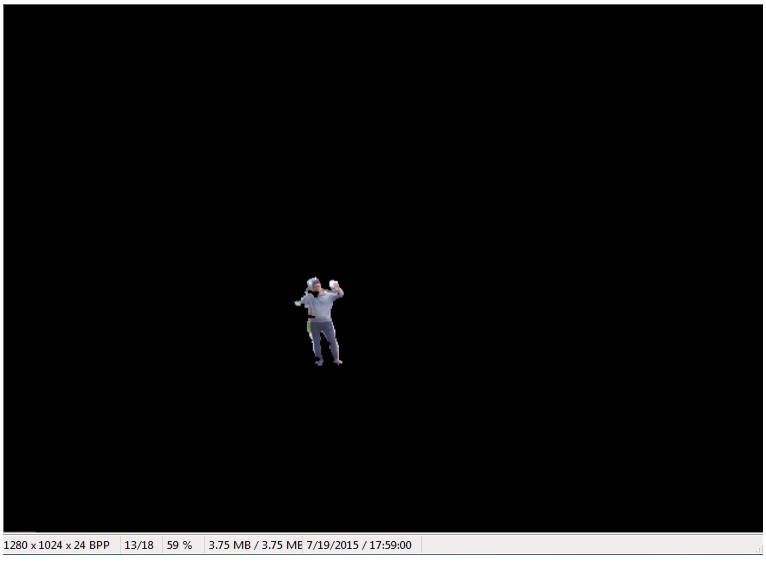


**Figure 29. Masked image**

The following image is the final result of this project where the colour space is converted. The mentioned module is not part of this project. This is the result of masking the upscaled image and the $YC_bC_r$ conversion where only the Y component is used and the remaining two components $C_r$ and $C_b$ are passed to another module to implement an averaging process on the component.

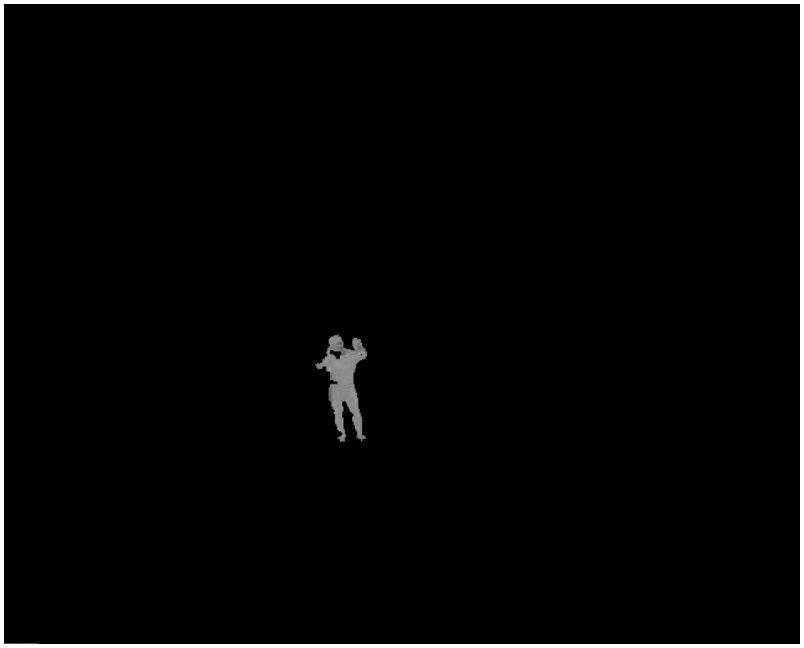

**Figure 30. Masked image converted to Y (luminance)**

## 5.3 Resource Utilization

The following table shows the utilization of resources available on Spartan 6.

| On-Chip | Used | Available | Utilization % |
|---|---|---|---|
| Clocks | 2 | - | - |
| Logic | 511 | 2400 | 21 |
| BRAM 8K | 0 | 24 | 0 |
| BRAM 16K | 8 | 12 | 67 |
| Signal | 777 | - | - |
| DSPs | 4 | 8 | 50 |
| IOs | 92 | 102 | 90 |

**Table 2. Resource utilization**

## 5.4 Power Consumption

The performance of the VHDL design for this project is estimated in terms of the resource utilization, power consumption and maximum running frequency based on the Xilinx Spartan 6 platform running two clocks, a visual clock at 26 MHz and a thermal clock at 10 MHz. The VHDL top module was synthesized and post-route simulation was performed. The parameters are calculated using the Xpower Analyzer, and the Xilinx ISE tool set [22]. The following are the results achieved. The leakage value in Table 3 refers to the device's static power. The static power of the device represents the amount of transistor leakage power when the device is turned on but not configured.



**Table 3. Hierarchical power consumption**

| Name | Power (W) | Logic Power (W) | Signal Power (W) | #FFs | #LUTs | #SRLs | #BRAMs | #DSPs | #CARRYs |
|---|---|---|---|---|---|---|---|---|---|
| Top | 0.00025 | 0 | 0.00025 | 0 | 0 | 0 | 0 | 0 | 0 |
| YC | 0.00045 | 0.00034 | 0.00011 | 221 | 170 | 37 | 0 | 4 | 32 |
| Upscale | 0.00019 | 0.00012 | 0.00006 | 146 | 355 | 0 | 0 | 0 | 38 |
| Memory | 0.00196 | 0.00196 | 0 | 0 | 0 | 0 | 8 | 0 | 0 |
| Total | 0.00285 | 0.00243 | 0.00043 | 367 | 525 | 37 | 8 | 4 | 70 |
| Leakage | 0.014 | | | | | | | | |

The following table indicates the power consumption of different signal types and as can be observed, the data signals require more power to function as their quantity is higher.

| Name | Power (W) |
|---|---|
| Data Signals | 0.00038 |
| Control Signals | 0.00005 |
| Total | 0.00043 |

**Table 4. Power consumption of signals**

# 6 Conclusion

Image processing is a demanding subject in electronic fields. Due to the high capabilities of this field, processing images and analysing the visual results is a reliable source for developing surveillance systems.

A reliable surveillance system should function in all types of lighting and weather conditions especially if the system will be used outdoors. Hence a camera system is presented that uses a thermal and visual camera, since a visual camera alone cannot meet the requirements when there is no light. After image acquisition the data need to be transmitted, therefore the volume of data need to be kept to a minimum. Thus, extracting the region of interest helps to prepare useful information for transmission phase. The presented thesis basically implements a specific scaling method called nearest neighbour and masks two images from different sources in both MATLAB and VHDL. Two different image sources are used, one is the thermal camera, the images go through an upscaling process. The other source is the visual camera, the image from which goes through a conversion from RGB to YCbCr colour space.

The results are represented in both MATLAB and Isim simulation. In addition, the factors in terms of power consumption are represented by Xpower Analyzer to estimate the implementation performance. The outcomes of this work could be used for the ongoing research project at Mid Sweden University where a real-time hardware railway surveillance system is going to be implemented completely on hardware, and the performance will be tested on a railway in real-time.

## 6.1 Social and Ethical Aspects

Safety has been a necessity of human life since day one, and as technology has developed, the need for safety has increased. Hence, safety issues are always inseparable from inventions.

Trains and railways as an invention require safety and surveillance in order to keep the number of injuries and deaths near zero. Increasing life

expectancy by controlling unnatural causes of death is an important service to society. Deploying a smart camera setup can help achieve this goal by increasing personal safety. The research work of this thesis can be used in order to improve the functionality of railway surveillance systems.
In addition to railways accidents, there are people who commit suicide by jumping in front of the moving trains. By improving the surveillance system for railways, they can be detected and stopped. In addition to saving their lives, hours of time can be saved for the passengers.

When it comes to ethical aspects, using images of people passing the railway and storing them could be a security issue, as there is a risk of information theft. Therefore the transmission of data needs to be done in a secure way and handed only to those concerned. However, for this project all processes have been done locally and the final result is transmitted in binary image, which makes privacy a non-issue. In addition, by consulting detected suicidal people, a sense of responsibility can be increased among people.

# 7 Discussion

One of the limitations of using the nearest neighbour algorithm in VHDL is that if the upscaling factor is decimal number it should be either rounded up or down. Hence the algorithm does not create a precise result in VHDL. An alternative is to measure the number of missing pixels or extra pixels and decide a value for them according to their neighbouring pixels. For instance, in the case of this project, the upscaled image was calculated to have a resolution of 619x463 which could not be achieved with an input image resolution of 352x240, therefore 85 pixels were added to the columns and 17 rows were added to the image. Instead of this inaccuracy, the added data could be ignored by deciding the pixel values in groups. For instance one can track 10 pixels of the original image and decide pixel values of 19 pixels in the upscaled image. By doing this, the upscaling factor of 1.9 can be achieved in VHDL.

Although RGB might be the best colour space for representation in many applications (e.g. television) but it has major limitations in activities such as image based surveillance systems. The main drawback of RGB in this case is the fact that the luminance information is embedded into red, green and blue layers [23]. Varying levels of brightness in images shifts the values of RGB which results in instability in images. The sensitivity of the RGB colour space to brightness shows that each layer is affected equally and that the layers correspond to each other [24].

# 8 Future Work

In real-world application, if all the camera setups (each setup containing a thermal and a visual camera) are connected with the same alignment with respect to each other, the zero-padding process can be omitted. A constant region of zero-padding could be created on the monitor of the operator, and hence the data transmission would have to deal with less data.

One of the key factors when it comes to efficiency in a system is power consumption; if the project only uses a thermal camera for object detection the power consumption can be decreased considerably. In this case, the visual camera can be set to hibernate mode and whenever the thermal camera signals the existence of an object, the visual camera can be turned on.

# References

[1] SVT Nyheter, *Mamma och två barn dog i tågolycka*, (2013)[online]. Available: http://www.svt.se/nyheter/regionalt/vasternorrland/person-med-barnvagn-pakord-av-tag-1 (Retrieved January 25, 2015)

[2] E. Aho, J. Vanne, T.D. Hämäläinen and K. Kuusilinna, "Configurable Implementation of Parallel Memory Based Real-time Video Downscaler", *Microprocess. Microsyst*, vol. 31, pp. 283–292, 2007.

[3] Freedman. G and Fattal.R, "Image and Video Upscaling from Local Self-Examples", *ACM Transactions on Graphics*. Vol. 30, Issue 2, Article No. 12, April 2011.

[4] Imran.M, Ahmad.N, Khursheed.K,O'Nils.M and Lawal.N, "Low Complexity Background Subtraction for Wireless Vision Sensor Node", *16TH Euromicro Conference on Digital System Design*, Spain, Sept. 2013.

[5] Tech-algorithm, *Bilinear Image Scaling*. (2009) [online]. Available: http://tech-algorithm.com/articles/bilinear-image-scaling/ (Retrieved March 21, 2015)

[6] Cambrige in Colour, *Digital Image Interpolation.* [online]. Available: http://www.cambridgeincolour.com/tutorials/image-interpolation.htm (Retrieved March 25, 2015 )

[7] Wunschmann.J, Zanker.S, Gunter.C and Rothermel.A, "Reduction of computational cost for high quality video scaling", *IEEE Transconsumer Electron*., vol. 56, pp. 2584-2591, 2010.

[8] Kim.C, Seong.S, Lee.A, and Kim.L, "Winscale: an image-scaling algorithm using an area pixel model", *IEEE Transactions* on Circuits and Systems for Video Technology 13 (6) pp.549–553, 2003.

[9] Amanatiadis.A and Andreadis.I, "Performance evaluation techniques for image scaling algorithms", *IEEE International Workshop on Imaging Systems and Techniques*, 2008.

[10] Studley.H, K. T. Weber, "Comparison of Image Resampling Techniques for Satellite Imagery", *Idaho State University* pp. 185 – 196, 2011.

[11] Han.D, "Comparison of Commonly Used Image Interpolation Methods", *ICCSEE Atlantis Press* pp. 1557-1559, 2013.

[12] Plex, *Video Calibration.* (2010) [online]. Available: http://www.hd-plex.com/blog/tag/yuv-to-rgb-conversion/ (Retrieved April 4, 2015).

[13] Deshpande.G , Borse.M, "Image Retrieval with the Use of Color and Texture Feature", *International Journal of Computer Science and Information Technologies,* Vol. 2 (3),pp.1018-1021, 2011.

[14] The Technical Experience, *Why the RGB to YCbCr*, (2009) [online]. Available: https://makarandtapaswi.wordpress.com/2009/07/20/why-the-rgb-to-ycbcr/ (Retrieved April 22, 2015).

[15] EE Times, *A tradeoff between microcontroller, DSP, FPGA and ASIC technologies,* [online]. Available: http://www.eetimes.com/document.asp?doc_id=1275272. (Retrieved June 13, 2015)

[16] Imran.M, O'Nils.M, Munir.H, Thörnberg. B**, "**Low Complexity FPGA Based Background Subtraction Technique For Thermal Imagery", *ICDSC,* Seville, Spain, 2015.

[17] Microsoft, *JPEG YCbCr Support,* [online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/dn424131%28v=vs.85%29.aspx. (Retrieved April 24, 2015)

[18] Xilinx, *RGB to YCrCb Color-Space Converter*, (2014) [online]. Available: http://www.xilinx.com/support/documentation/ip_documentatio

n/v_rgb2ycrcb/v7_1/pg013_v_rgb2ycrcb.pdf (Retrieved April 25, 2015).

[19] Padmashri. S, Sundaram.A.A, "Feature Extraction in Compressed Domain for Content Based Image Retrieval", *International Conference on Advanced Computer Theory and Engineering*. pp. 190 – 194, 2008.

[20] Equasys, *Color Conversion.* [online]. Available: http://www.equasys.de/colorconversion.html (Retrieved May 2, 2015).

[21] Green.B, *Raster Data Tutorial*. (2002) [online]. Available: http://dasl.mem.drexel.edu/alumni/bGreen/www.pages.drexel.edu/_weg22/raster.html (Retrieved May 3, 2015).

[22] www.xilinx.com, last accessed, 18th June 2015.

[23] Kim. W.-S., D.-S. Cho, and H.M. Kim. "Interplane prediction for RGB video coding". *International Conference on Image Processing.* Vol.2, pp.785-788, 2004.

[24] Sebastian.P, Yap Vooi Voon. R, Comley. R, "The Effect of Colour pace on Tracking Robustness" *3rd IEEE Conference on Industrial Electronics and Applications*. pp. 2512-2516, 2008.

[25] R.C. Gonzales and R.E. Woods, *Digital Image Processing*, Pearson Education, 2008.