REAL TIME IMAGE PROCESSING FOR MEDICAL INFRARED IMAGING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

CEMİL KIZILÖZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

DECEMBER 2005

Approval of the Graduate School of Natural and Applied Sciences

Prof. Dr. Canan ÖZGEN
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. İsmet ERKMEN
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Prof. Dr. Nevzat G. GENÇER
Supervisor

Examining Committee Members

Prof. Dr. Cengiz BEŞİKÇİ (METU, EEE) _____

Prof. Dr. Nevzat G. GENÇER (METU, EEE) _____

Assist. Prof. Dr. Yeşim SERİNAĞAOĞLU (METU, EEE) _____

Assist. Prof. Dr. Çağatay CANDAN (METU, EEE) _____

Dr. İpek ZIRAMAN (NUMUNE, RADIOLOGY) _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Cemil KIZILÖZ

ABSTRACT

REAL TIME IMAGE PROCESSING FOR MEDICAL INFRARED IMAGING

Kızılöz, Cemil
M.S., Deparment of Electrical and Electronics Engineering
Supervisor: Prof. Dr. Nevzat G. Gençer

December 2005, 95 pages

This thesis focuses on Medical Infrared Imaging. It deals with the implementation of an infrared imaging system that can be used as a thermograph. A user interface program is also designed in order to control the imaging system.

The system is implemented using Very High Speed Integrated Circuit Description Language (VHDL). Digitizing the data is implemented by Field Programmable Gate Array. Non-uniformities at the detector data are corrected by the two-point correction algorithm. To obtain absolute temperature readings, another system calibration process is also performed. Real-time histogram equalization algorithm and a real-time convolution operation are implemented using the VHDL. Tests of these implementations are performed by comparing the results with the numerical values. A user interface program is developed to allow the operator select any filter type and measure the temperature of any point in the object. Previous studies showed that an infrared system should detect a temperature difference of 500°mK if it is to be used for biomedical applications. Using a black body system with a precise temperature control, it is shown that this specification is satisfied. Clinical evaluations for a few

patients reveal that the implemented medical infrared system can be used for biomedical applications.

Keywords : Medical Infrared Imaging, Thermography.

ÖZ

KIZILÖTESİ KAMERA İLE GERÇEK ZAMANLI MEDİKAL GÖRÜNTÜ İŞLEME

Kızılöz, Cemil
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Nevzat G. Gençer

Aralık 2005, 95 Sayfa

Bu tez, medikal amaçlı kullanılabilen bir kızılötesi görüntüleme sistemi çalışmasıdır. Termograf olarak kullanılabilecek bir kızılötesi kameranın tasarımı üzerine çalışmalar yapılmıştır. Bir arayüz programı da tasarlanarak sistemin çalışması kontrol edilmiştir.

Bu tez çerçevesinde, sistem Çok Yüksek Hızlı Entegre Devre Donanım Tanımlama Dili (VHDL) kullanılarak Alan Programlanabilir Kapı Dizinleri (FPGA) üzerinde geliştirilmiştir. Dedektörlerin eşdeğer olmama problemi iki noktalı düzeltme algoritması kullanılarak çözümlenmiştir. Bir başka düzeltme algoritması ile gerçek değerli sıcaklık ölçümlerinin yapılabilmesi sağlanmıştır. Gerçek zamanlı dağılım eşleme algoritması ve gerçek zamanlı katlama operasyonu da sistem üzerinde gerçekleştirilerek çeşitli filtreleme uygulamaları yapılmıştır. Bu algoritmaların test ve simülasyonları hesaplanan değerlerle karşılaştırılarak gerçekleştirilmiştir. Kullanıcının denek üzerinde nokta sıcaklık ölçümü yapabilmesi ve uygulanacak filtre türünü seçebilmesi için bir arayüz programı tasarlanmıştır. Eski çalışmalar, 500 °mK hassasiyetinde kameraların medikal amaçla kullanmak için yeterli olduğunu

göstermektedir. Sistem kalibrasyonu yapıldıktan sonra, karacisim ile gerçekleştilen testler ile sistemin medikal amaçlı kullanılabilecek kadar hassas olduğu görülmüştür. Birkaç hasta üzerinde gerçekleştirilen klinik çalışmalar ile sistemin medikal amaçlı olarak kullanılabileceği gösterilmiştir.

Anahtar Kelimeler : Medikal Kızılötesi Görüntüleme, Termografi.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor Prof. Dr. Nevzat G. Gençer for his encouragement, valuable comments and continued support throughout this study.

I am also thankful to Dr. İpek Zıraman for her contribution to the study through the clinical applications.

Special thanks to ASELSAN Inc. Microelectronics, Guidance and Electro-Optics division for providing technical support and laboratory environment in which we could develop our design.

I am also thankful to all my friends in ASELSAN and METU who have helped me throughout this thesis.

Finally, I would like to thank my family for their patience and moral support.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1
INTRODUCTION

1.1 Overview

Any object whose temperature is above –273C (0˚K) emits infrared radiation which can not be sensed by human eye [24], [25]. To convert thermal infrared radiation into visible light, infrared (IR) or thermal imaging systems are employed. A pathological tissue has a metabolic rate which is different from the other parts of the body, yielding temperature differences of about 500 mK on the body surface [22]. Recently developed cooled infrared systems can detect temperature differences of 10mK. Uncooled infrared cameras are capable of measuring temperature differences of 100˚mK. Thus, infrared imaging systems may find a use in biomedicine. Depending on the application, it is possible to prefer uncooled systems due to their lower cost.

In this study, the performance tests of an uncooled infrared imaging system are investigated. The system is also assessed by exploring clinical applications in the Radiology department of the Numune hospital in Ankara.

1.2 Infrared Basics

Figure 1-1 is the block diagram of a basic thermal imaging system. The IR Imager part is the optical system that collects the infrared radiation and sends to the IR detector. The IR detector part converts the radiated infrared power into electrical signals. These electrical signals must be processed before it is monitored. Front-end electronics is the interface between the processing and the detector parts. This part is responsible for producing the analog signals of the detector and digitizing the detector output. The system calibration and digital signal processing algorithms (to

enhance the details in the image) are performed at the digital signal processing part. The output of that part is sent to a standard monitor.

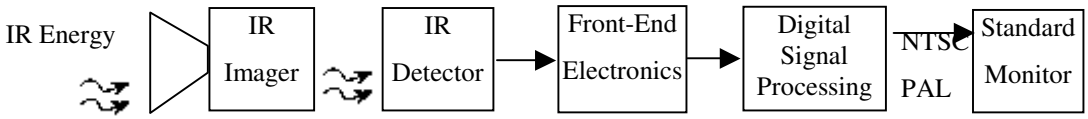

Figure 1-1 Block diagram of basic infrared imaging system

1.2.1. Infrared and Dedector Physics

Infrared radiation is radiated electromagnetic energy. It is same as visible light or radio waves. However, its oscillation frequency is different than the other wave types. IR extends from 0.7 μm to 1000 μm range of the electromagnetic spectrum. However, the whole spectrum cannot be used for IR imaging due to the transmission limits of the atmosphere. Atmosphere absorbs the radiated power at some regions of the spectrum. IR detectors are designed in order to operate where the transmission is maximum. The 0.7-1.1 μm range (NIR –Near Infrared), the 1.5-1.8 and 2.0-2.4 μm range (SWIR - Short Wavelength Infrared), the 3-5 μ range (MWIR - Medium Wavelength Infrared) and the 8-14 μm range (LWIR - Long Wavelength Infrared) are the regions where the transmission is maximum and detectors are designed to operate. Figure 1-2 shows the atmospheric transmittance of these regions [26].

Infrared detectors can be divided into two groups according to their operating conditions, namely, cooled and uncooled infrared detectors.

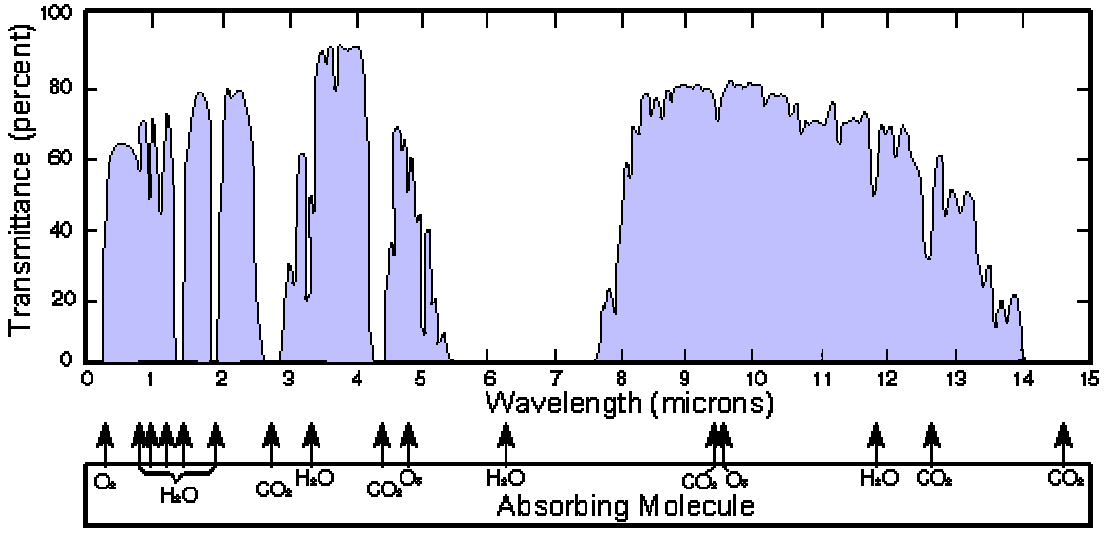

Figure 1-2 Atmospheric Transmittance[26]

Cooled infrared detectors typically operate at a temperature of 70 K, 80 K or 110 K. Operating temperature is determined by the detector material. In cooled type detectors, the incoming radiation is absorbed by the detector to produce electron and hole pairs which leads to charge on the detector. However, without cooling, electron hole pair production due to own temperature of the detector material is very high. Hence a cooling mechanism is needed. Indium antimonide, indium arsenide, HgCdTe, lead sulfide, lead selenide are some of the metarials that are used as cooled detector materials [26].

Uncooled infrared detectors operate at room temperature. But in most cases a thermal stabilization process at room temperature is required. The radiated infrared power leads to some electrical changes at detector material. In case of microbolometers, the resistance of the material changes when exposed to infrared radiation. This change is due to the increase at the temperature of the detector. Vanadium oxide and amorphous silicon are the materials that are mostly used for uncooled infrared detector production [26].

Cooled detector are more sensitive detectors. However, they need cooling mechanisms, they are bigger and more expensive with respect to uncooled detectors. On the other hand uncooled detectors are less sensitive but, they are cheaper smaller and can operate at room temperature.

1.2.2 Thermographic Evaluations

Infrared imaging systems provide information about the surface temperature of an object. An pathology in a living tissue makes metabolic rate changes at that part of the body, for example, the number of blood vessels is increased to support the growth of a tumor. This may increase the surface temperature and can be detected by an infrared system. Advantage of an IR imaging system is that it imposes no radiation on the object. It detects the naturally emitted radiation. So it is safe to use on any person, including applications on pregnant women.

A list of the illnesses that IR imaging can be used is given in Table 1-1 [23].

Table 1-1 IR Thermography Applications[23]

A Thermographic Evaluation is recommended for any of the following indications:	
Altered Biokinetics Arteriosclerosis Brachial Plexus Injury Biomechanical Impropriety Breast Disease Bursitis Carpal Tunnel Syndrome Causalgia Compartment Syndromes Cord Pain/Injury Deep Vein Thromosis Disc Disease Disc Syndromes Dystrophy External Carotid Insufficiency Facet Syndromes Grafts Headache Evaluation Herniated Disc Herniated Nucleus Pulposis Hyperaesthesia Hyperextension Injury Hyperflexion Injury Inflammatory Disease Internal Carotid Insufficiency Lumbosacral Plexus Injury Ligament Tear Lower Motor Neuron Disease Malingering Median Nerve Neuropathy Morton's Neuroma Myofascial Irritation	Nerve Impingement Nerve Pressure Nerve Root Irritation Nerve Stretch Injury Nerve Trauma Neuropathy Neurovascular Compression Neuralgia Neuritis Neuropraxia Neoplasia (melanoma, squamous cell, basal) Nutritional Disease Peripheral Nerve Injury Peripheral Axon Disease Raynaud's Referred Pain Syndrome Reflex Sympathetic Dystrophy Ruptured Disc Somatization Disorders Soft Tissue Injury Sprain/Strain Stroke Screening Synovitis Sensory Loss Sensory Nerve Abnormality Superficial Vascular Disease Skin Abnormalities Thoracic Outlet Syndrome Temporal Arteritis Trigeminal Neuralgia Trigger Points

The clinical evaluations are mostly focused on the diagnosis of breast cancer, pain level recognition, applications during coronary bypass surgery, detecting diabetic foot, imaging burn trauma, nerve pressure and nerve trauma [23].

First studies on medical IR Imaging started on late 60's and early 70's. Since the technology was not developed enough, and the staff used for evaluations were not qualified, earlier studies on the biomedical applications of Infrared Imaging were not successful. After 70's, for about a decade, there were no studies on medical IR imaging [1], [2]. Investigations on IR imaging continued for military purposes. However, in the last 5-10 years, due to recent advances in the detector technologies, investigators started to work for medical IR applications. It was observed that it is possible to get results comparable to mammography [3].

Currently, there are three main approaches used in the diagnosis of illnesses: Direct Infrared Imaging, Tau Imaging and Dynamic Area Thermography. Apart from them, there are also methods that utilize the symmetry in the human body (like subtracting left half of the body from the right half [8], [11]). In the next sections, these approaches will be introduced.

1.2.3 Direct Infrared Imaging

Direct Infrared Imaging is the simplest test method used by medical infrared imaging applications. An IR camera is placed in front of the subject and direct *temperature outcomes* are investigated. The outcomes are 1) the average temperature differences, 2) hot spots, and 3) asymmetries.

For the average temperature difference analysis, the body of the patient is divided into four segments. The average temperatures of these parts are investigated. It was shown that normal breast quadrants has a temperature difference ranging from 0.14°C to 0.29°C [22]. It was also claimed that 0.5°C is the best threshold for

detecting cancer in tissues. Differences of 1°C represent the high risk category for the breast cancer. It was claimed that choosing the threshold as 0.5°C results in an increase in false positives [16]. A threshold value of 1°C, however, eliminates the false positives.

Another abnormality used for diagnosis is the hot spots in the image. Hot spots are the high local temperature differences in the image. Local changes of 1°C up to 2°C are considered as hot spots. It was shown that when the results of infrared system are combined with mammography [14], the sensitivity rate of the imaging system increases from 85 % to 95 % [7], [17].

Temperature map of human body is almost symmetric. Asymmetric thermographs are considered as abnormal. Temperature differences of 0.5°C up to 1°C are the temperature values that lead to asymmetric images [7], [14].

At some studies, the patient is first taken to a cold environment. The abnormalities in the temperature distribution after that cold period are also not desired.

In the Direct Infrared Imaging approach, output of the imaging system is investigated considering an a priori determined threshold value. This threshold value varies among different studies. Most commonly used threshold value is 0.5°C. If temperature of some part of tissue is higher then the threshold value, it is considered as an illness sign. Asymmetries and hot spots are also two other subjects that direct infrared imaging method takes into account.

1.2.4 Tau Imaging

Tau imaging method discusses the heating responses of body by taking subject temperature measurements at cold and hot environments. Usually, the object is placed at a cold environment (actually 10°C -15°C) for a specific period. Thereafter, the temperature is increased slowly. Images, which are usually captured with a period

of 30 seconds, are stored. The stored images are subtracted from each other and heating response of the subject is discussed [13]. Like Direct Imaging method, temperature differences, hot spots and asymmetries are considered as abnormal findings. It was claimed that, 80% true positive rate can be achieved by this imaging approach [13]. It was also reported that, in some patients, although abnormalities were not recognized by the Direct Infrared Imaging approach, they can be detected by Tau imaging [9]. Hence, Tau imaging method increases the true positive rate. When heating responses of body is discussed, better results were obtained [18], [21].

1.2.5 Dynamic Area Thermography

Dynamic Area Thermography (DAT) analysis gives the best result among the three approaches, but it needs more sensitive and fast processing cameras.

NO is a substance that is related with the development of the cancerous tissue.. It causes heat changes on the body surface. The frequency of these changes gives information about the existence of NO [4]. It was reported that “Independent observations had shown that skin temperature is modulated in a periodic manner, and this modulation is a function of neural modulation of blood flow. As NO functions as a chemical messenger in neuronal modulation, it is expected that this modulation will be altered in the presence of cancer-associated extravascular NO. The modulation of skin temperature can be monitored and quantitatively assessed by DAT.“ [4]. This method requires the Fast Fourier Transform of the time sequence of each pixel. To ensure correct temperature measurement for each pixel, 1000 images must be taken in 10 seconds. The Fourier Transform of the images are analyzed. This transform gives information about the frequency change of each pixel, as NO value for each pixel is reached [4]. Using this approach, the abnormalities in the frequency are investigated. Up to 90% sensitivity, 95% specificity values can be reached [5] [6].

1.3 FPGA Structure

Field programmable gate array (FPGA) is a general purpose integrated circuit. Application specific integrated circuit (ASIC) performs similar functions but it can not be reprogrammed. FPGA can be reprogrammed after it has been deployed into a system. It is programmed by an FPGA system designer.

Programming is performed by downloading configuration data (bit stream) into static on-chip random-access memory. The configuration data is the product of compilers. These compilers translate the high level abstractions produced by the FPGA system designer into a something equivalent, but low level, and executable code. There are many compilation tools in the industry. Most popular of them are Precision, Leonardo Spectrum and XST .

FPGAs are high performance signal processing devices. They provide to construct highly parallel architectures for processing signal. FPGA performance is originated from this ability. Microprocessor or DSP processor performance is limited to the clock rate at which the processor can run. However, the FPGA performance is limited to the amount of parallelism employed to implement the algorithms of a signal processing system. FPGAs can now operate with clock frequencies of up to 500 MHz. Although this may seem to be slow, FPGAs operate with parallelism. FPGA and DSP represent two very different approaches to signal processing. Each one is good for different tasks. There are many high-sampling-rate applications that FPGA can do easily, while DSP can not. Equally, there are many complex software problems that FPGA cannot address. As a result, an ideal system often splits the work between FPGAs and DSPs.

FPGAs are implemented with a regular, flexible programmable architecture of configurable logic blocks (CLBs), interconnected by versatile routing resources (routing channels), and surrounded by programmable input/output blocks (IOBs), as seen in Figure 1-3. This implementation is a basic structure.

Most of the logic in FPGA is implemented by configurable logic blocks. Internal structure of CLBs changes with FPGA family and FPGA manufacturer. Basic diagram of CLB for XC4000 family Xilinx FPGA that is used in this thesis is shown in Figure 1-4. There are two 4-input function generators (Function generator is called as look-up table (LUT) in some documents.) which are labeled as F and G in Figure 1-4 The third function generator (H) is also provided. H function generator has three inputs as shown in Figure 1-4

Each CLB contains two storage elements (d type flip-flops (ff)) that can be used to store function generator outputs and direct inputs coming from outside the CLB as shown in Figure 1-4. Well known FPGA producers are listed in Appendix B.

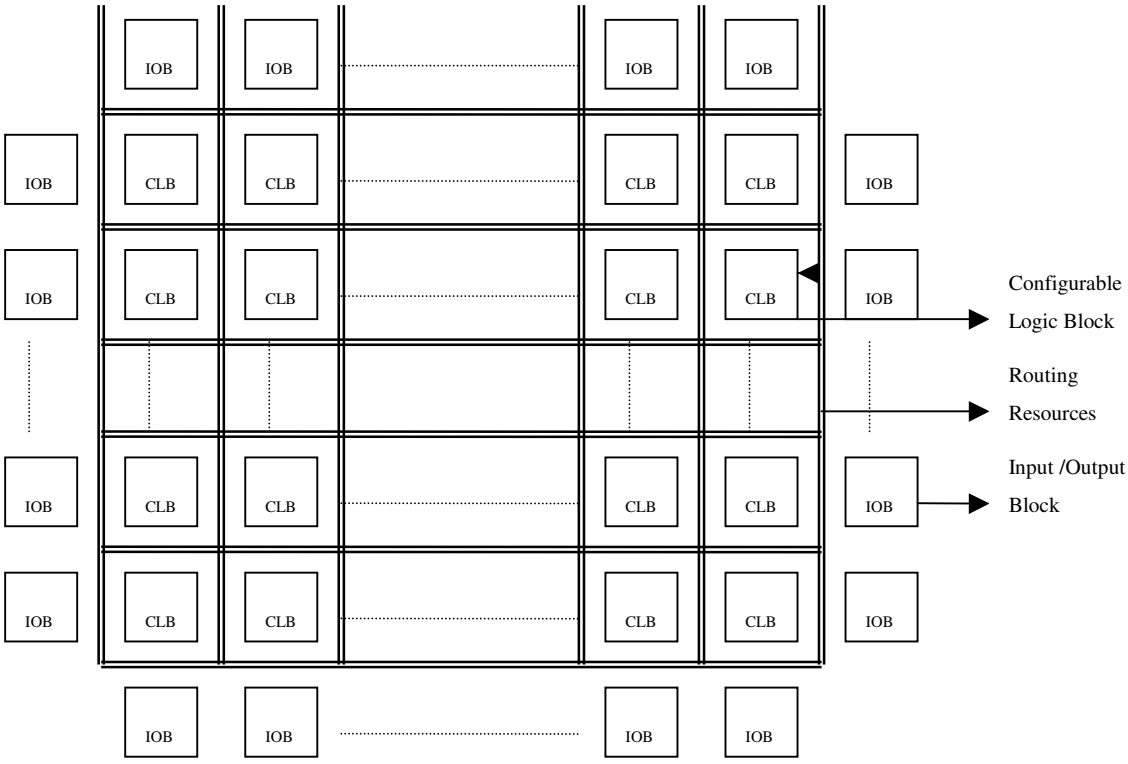

Figure 1-3 IO Blocks and Logic Resources in a FPGA Structure

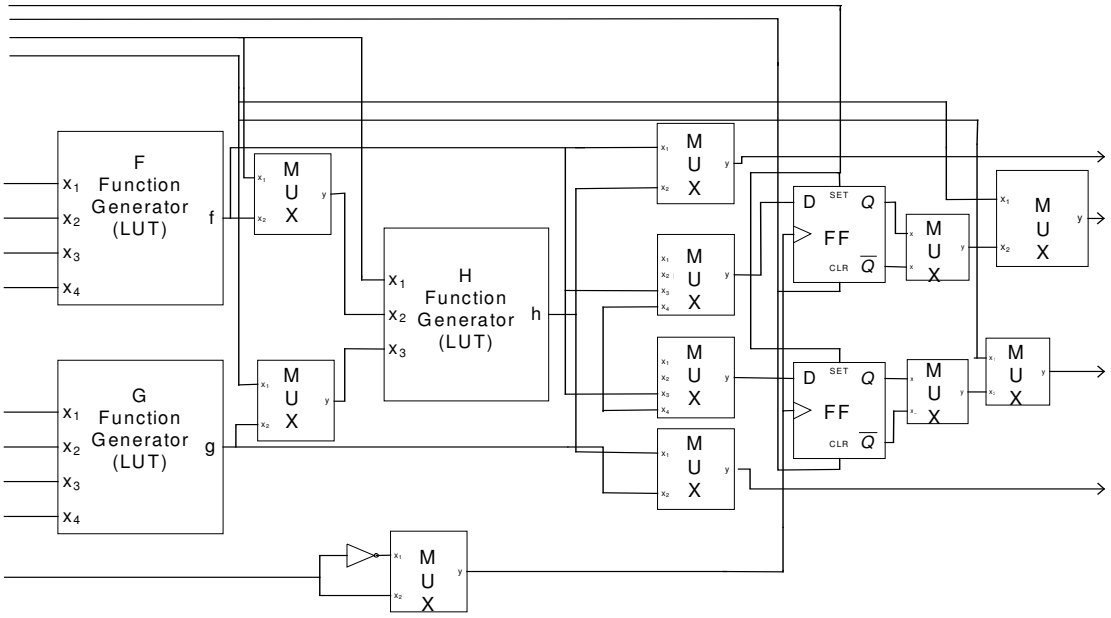

Figure 1-4 Configurable Logic Block Structure in a FPGA with Three Function Generators Eleven Multiplexers and Two Flip Flops

1.4 Aims

During this thesis study, an uncooled military infrared camera is developed in Aselsan AŞ. The author of this thesis study has taken part in this development period and tested the systems's specifications for biomedical applications. Specific goals of this thesis study are listed below:

To implement a two-point calibration using VHDL,

To design data capture communication blocks for the calibration,

To implement a real-time offset correction algorithm,

To calibrate the system for absolute temperature measurements,

To implement a real-time Histogram Equalization algorithm using VHDL,

To implement a convolution block to apply various filters to the image.

To assess the performance of the resulting system for biomedical applications,

To explore possible biomedical applications in a clinical environment.

1.5 Organization of the Thesis

This thesis consists of six chapters. Chapter I provides the background information on infrared imaging systems, FPGA structure and the current status of the medical infrared imaging. Chapter II explains non-uniformity correction and image processing applications. Chapter III focuses on the filtering structure. And also shows functional and timing simulations. Chapter IV describes the approaches adapted in order to measure true temperature. Chapter V deals with the clinical applications and shows experimental results . Finally, chapter six concludes the overall work and outlines the directions of future work.

CHAPTER 2

NON-UNIFORMITY CORRECTION and IMAGE PROCESSING

2.1 Non-uniformity Correction

Output of the focal plane array detectors are not uniform. Although each pixel of the array behaves in a similar way, the output of detector without non-uniformity correction is not meaningful. When the temperature of the scene increased, the output of each detector pixel also increases. However, the increase at each pixel is not same due to pixel size and noise effects. Figure 2-1 shows the raw output of the detector used in this study.. This image reveals severe non-uniformity in the response of the detectors. (In addition to horizontal and vertical stripe like artifacts due to production procedure of the detector.)

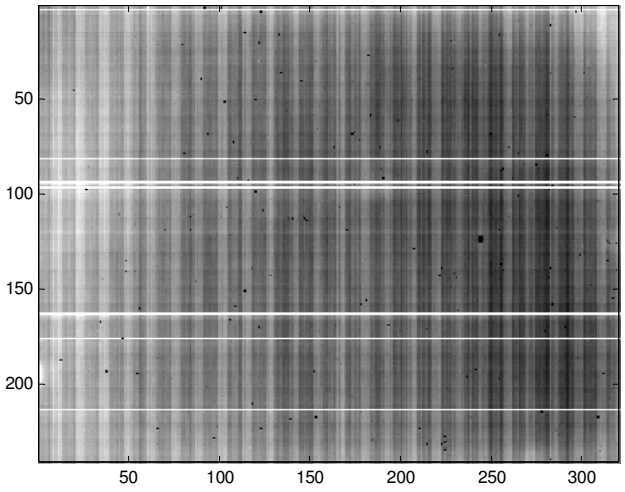

Figure 2-1 Raw Detector Data

Thus, the output of each pixel has to be normalized to have same response for an uniform radiation. This is achieved by a correction algorithm. In order to perform non-uniformity correction, the detector data when exposed to two different temperature scenes are measured and gain and offset coefficients for any pixel are calculated.

Non-uniformity correction procedure is as follows.

- Camera is placed in front of a uniform scene at a low temperature value T1. Temperature values for any pixel of the camera Tp1(i,j) are measured.
- Temperature of the uniform surface is increased to a higher value (T2) and Temperature values for any pixel Tp2(i,j) are recorded..
- Gain Gp(i,j) and offset Op(i,j) values of any pixel are calculated.

The temperature of the corrected pixel Cp(i,j) is obtained as follows:

$$Cp(i,j)= Tp(i,j)*Gp(i,j)+Op(i,j),$$

Non-uniformity correction is performed so that the corrected pixel values, when exposed to a radiation from the scene of T1, are mapped to the average of the pixels at temperature T1. The corrected pixel, when exposed to a radiation of the scene of T2, are mapped to the average of the pixels at temperature T2. The relevant expressions related to these operations are given below,

$$\frac{\sum_{i,j} Tp1(i,j)}{i*j} = Tp1(i,j)*Gp(i,j)+Op(i,j)$$

$$\frac{\sum_{i,j} Tp2(i,j)}{i*j} = Tp2(i,j)*Gp(i,j)+Op(i,j)$$

Thus we obtain,

$$Gp(i,j)=\frac{\frac{\sum_{ij} Tp1(i,j)}{i*j} - \frac{\sum_{i,j} Tp2(i,j)}{i*j}}{Tp1(i,j) - Tp2(i,j)}$$

$$Op(i,j)=\frac{\sum_{ij} Tp1(i,j)}{i*j} - Tp1(i,j)*Gp(i,j)$$

Normalized form of the image in Figure 2.1 is shown at Figure 2-2.

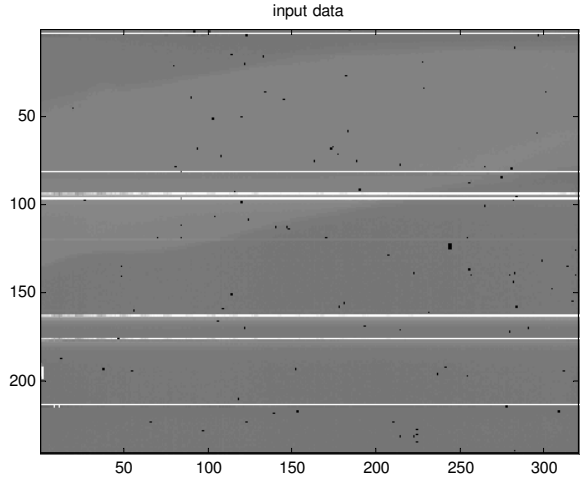

Figure 2-2 Normalized Image

2.2 Image Processing

The normalized output of the detector is a low-contrast image. This image has to be processed in order to show the details in the object. This low contrast image is due to the high precision in the analog-to-digital (AD) converters. For example, a 12 bit AD converter has 4096 digital output levels. When detector data is sampled, it has 400 or 500 digital levels. However, since it is scaled over 4096 levels, the image is low-

contrast and the details on the image cannot be observed (although the image carries this information). In fact, the human eye is sensitive to about 200 levels. Thus, it is sufficient to display images using 8 bits. In this study, details on the image are enhanced by transferring the low-contrast 12 bit image to 255 level (8 bit) images. The Histogram Equalization technique can be applied to solve this problem. Details of that algorithm are explained in the following sections.

2.2.1 Histogram Equalization

Histogram equalization is an algorithm to enhance the details on the image. For low-contrast images, the algorithm starts by analyzing the histogram of the image which shows the number of pixels at each gray level. Figure 2-3 shows a sample low-contrast image. Figure 2-4 shows the histogram of this data.

Figure 2-3 Low Contrast Detector Data

The algorithm first calculates the histogram of the low-contrast image, and finds an inverse function which flats the histogram of the output image. According to the mapping function, lowest histogram value is mapped to “0” digital level and highest histogram value is mapped to “255” digital level. The levels between lowest and highest values are mapped with respect to the histogram value. If the histogram value

is higher they are mapped to a wider area, as there are lots of information in this area. If the histogram value is lower, the image data is mapped to a smaller area, as there is less information in this area.

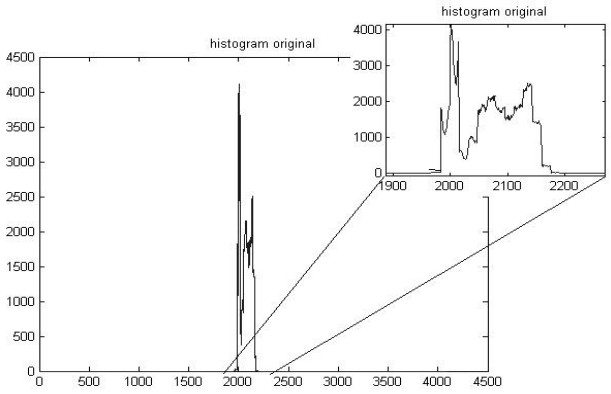

Figure 2-4 Low Contrast Detector Data Histogram

Figure 2-5 Histogram Equalized Image

Figure 2-5 shows the output image after histogram equalization. Figure 2-6shows the output histogram and Figure 2-7 shows the mapping function. The timing simulations of the algorithm and the camera output (Figure 2-5) have the same values pixel by pixel.

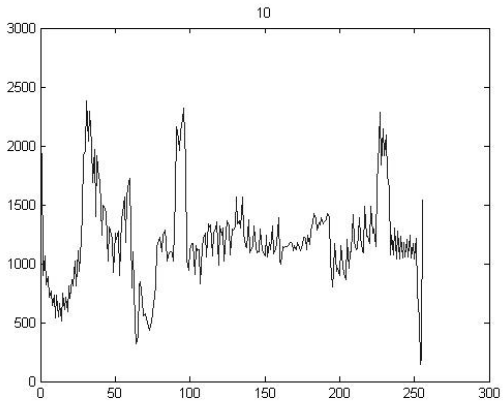

Figure 2-6 The Histogram of the Equalized Image

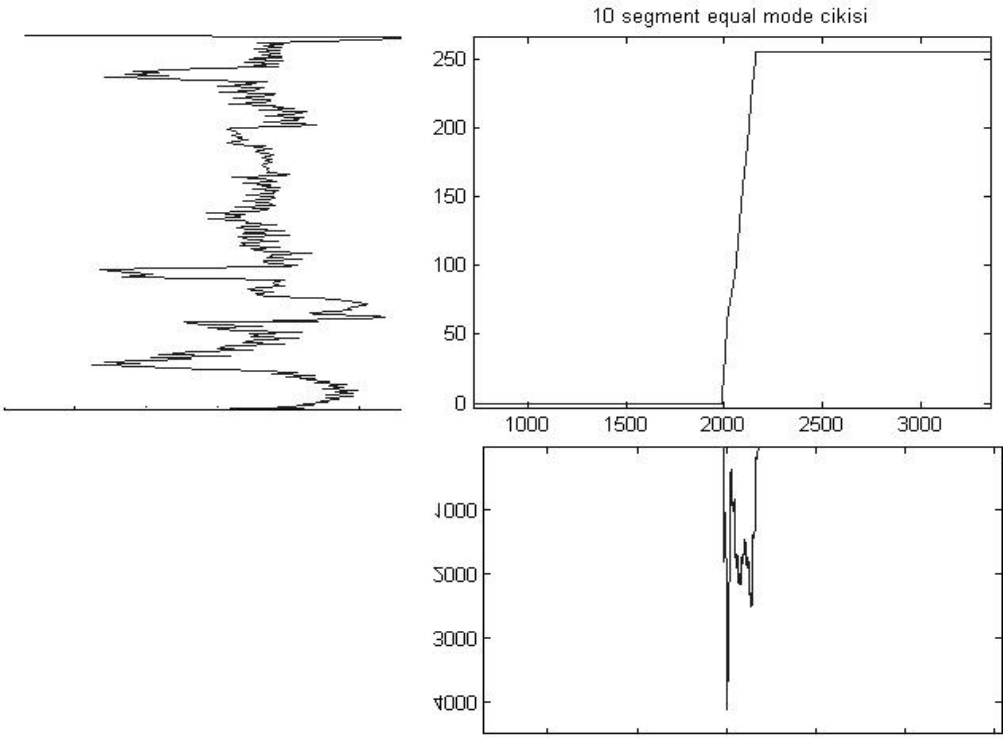

Figure 2-7 Histogram Equalization Mapping Function

Figure 2-8 shows the FPGA architecture designed to implement the Histogram Equalization algorithm. Top level block consists of 4 blocks. Statistics block receives

the image data and calculates the histogram of the image and stores it to a dpram4kx18. The coefficient-finder and gain-finder blocks calculate the inverse mapping function using the saved histogram information, and write the coefficients to a dpram1kx18. Compression block maps the input image to an 8-bit output image according to the mapping coefficients which are present at dpram1kx18.

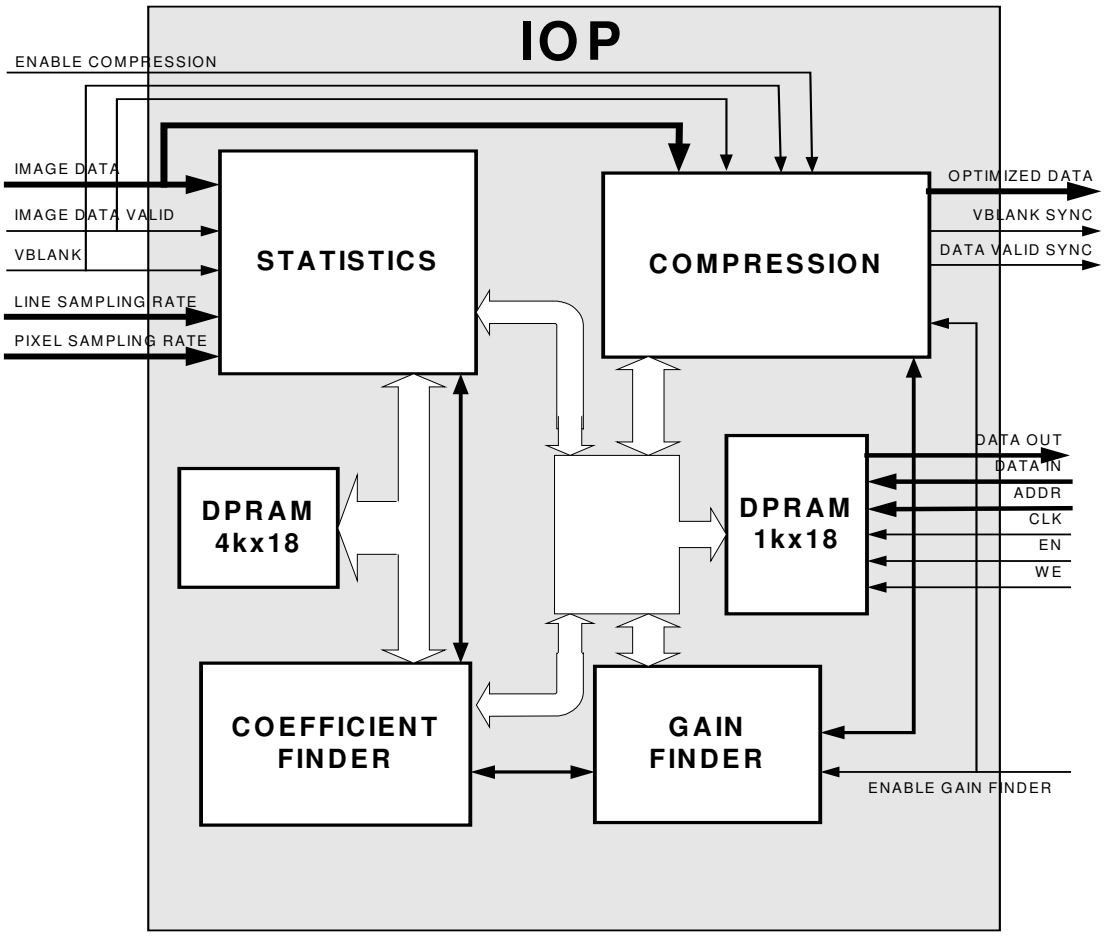

Figure 2-8 FPGA Structure for the Histogram Equalization Algorithm

CHAPTER 3

FILTERING

3.1 Introduction

Convolution is the basics for the image processing applications. Edge finding, smoothing, or some other applications can be implemented by FIR filters. Thus a convolution operation to apply FIR filter is implemented through this thesis work. Convolution operation is performed by scanning a finite size and shape window (FIR filter) across the image. The output pixel is the weighed sum of the pixels on the scanning window. The applied filter in this thesis work is a 5x5 filter. A user interface program is designed to send the filter coefficients to the system. The filter coefficients can be updated in real-time by the developed interface program. A scene of the interface program is shown in Figure 3-1. Using this interface, the type of the filter can be chosen through a combo-box. The filtering operation is applied to the image produced by the system. In the rest of this chapter, first the convolution block design is explained in the structure part. The output and the simulation results are then explained in the simulation part.

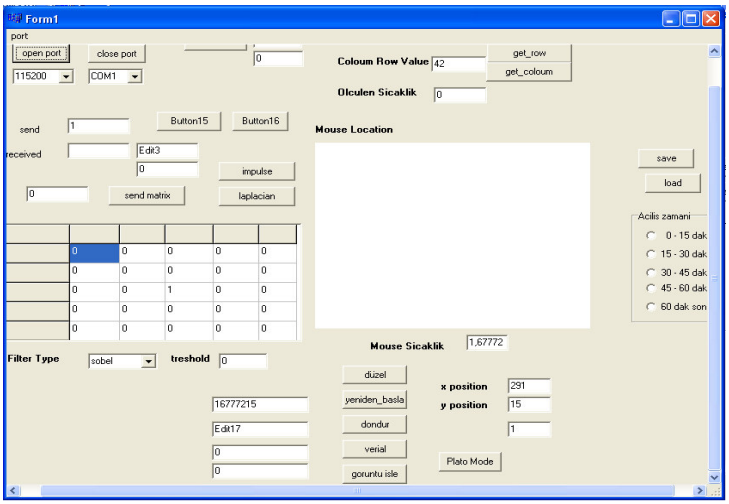

Figure 3-1 The User Interface Program

3.2 Structure

Convolution operation is performed by the hardware implemented in the FPGA by the VHDL. The filter coefficients (the 5x5 matrix) are sent to the system by the interface program. The communication is performed through the RS232 protocol at 115200 baud rate. The convolution operation hardware can be divided to 3 main blocks. First part is the communication block, which performs the transmission of the coefficients to the coefficient RAM. Second part is the FIFO implementation part which creates the necessary delay at the incoming video stream. Third block takes care of the convolution process. Figure 3-2 is the block diagram of the filtering hardware.

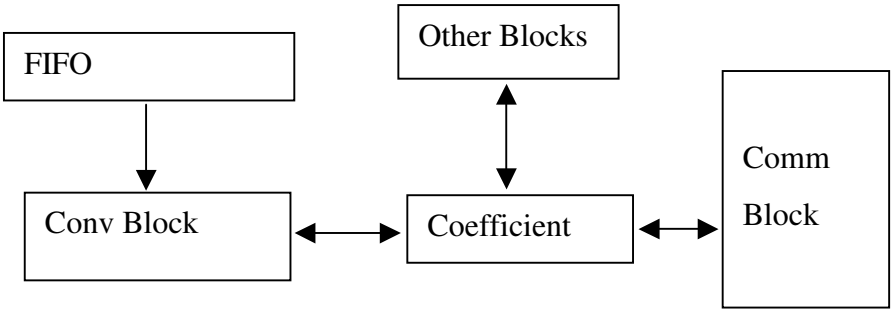

Figure 3-2 Filtering Structure

3.2.1 The RS232 Communication Block

The RS232 communication block receives the commands sent from the interface program. It consists of three blocks, an 8-bit asynchronous receiver block, an 8-bit asynchronous transmitter block and a controller block. The implemented RS232 protocol is as follows: First the start and command bytes are sent to the system. Then the parameters of the commands with a checksum are sent to the hardware. The receiver block decodes the incoming bytes and sends them to the controller block. The controller block decodes the command, checks if it is a valid command or not, and enables the transmitter block to send whether the command received is a valid message or not. The filter coefficients are received by this communication block. The received filter coefficients are stored at the where the image-processing coefficients are stored. That RAM is not accessible by communication block for all the times. The image processing coefficients are calculated for each field, at the vertical blanking time (time interval between the two fields). The controller block cannot reach to the coefficient RAM while the image-processing coefficients are being calculated. Controller block checks the status of the RAM and stores the filter coefficients after the image processing coefficients are calculated. When coefficients are written to the RAM, the controller block enables the transmitter block to send the

acknowledge signal. The RS 232 transmitter and receiver blocks' VHDL codes are presented at Appendix C. Figure 3-3 explains the communication block.

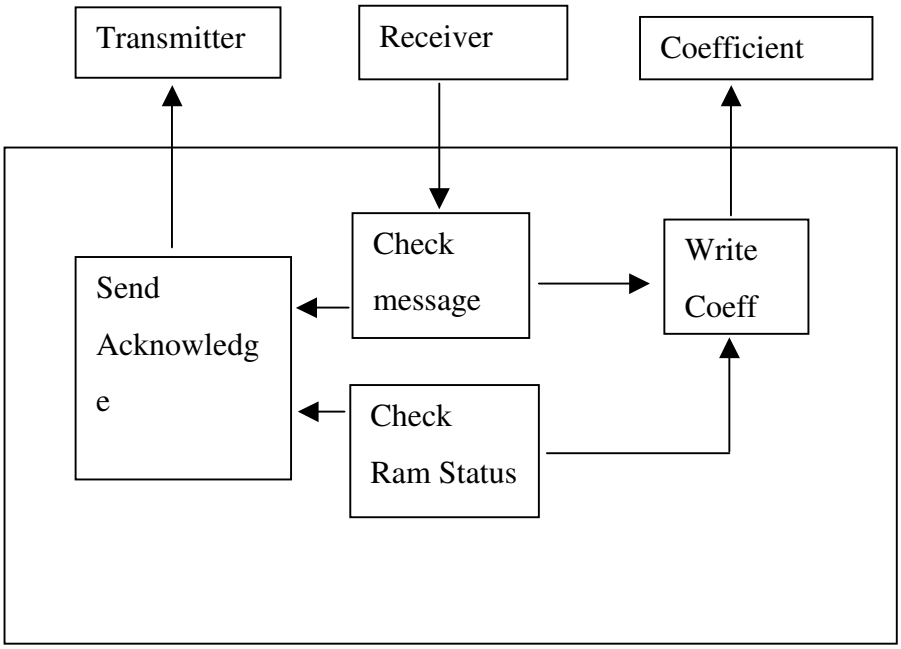

Figure 3-3 Communication Block Structure

3.2.2 The FIFO Controller Block

In order to implement a 5x5 filter, 5-line delay is needed. Figure 3-15 shows 5 line delayed structure signals. Five FIFOs are implemented using RAMs. According to video control signals, the incoming video signal is written to FIFOS. For this purpose, 5 FIFO structures are cascaded. The FIFO controller block is responsible for writing the image stream and read the image data at correct time instants. The output of each FIFO introduces one line delay, which enables us to implement the filter. For each FIFO structure, read operation is one clock before the write operation (actually write operation is one clock delayed), in order to read the correct data and then write new data to the same location. This structure helps to use all locations of the RAM efficiently.

3.2.3 Convolution Block

The ordinary convolution equation is

$$c[m,n] = a[m,n] \otimes h[m,n]$$

$$= \sum_{j=0}^{J-1}\sum_{k=0}^{K-1} h[j,k]\,a[m-j,n-k] = \sum_{j=0}^{J-1}\sum_{k=0}^{K-1} h[m-j,n-k]\,a[j,k]$$

which means that the filter window is scanning the image. Scanning is obtained by the FIFO structure. Control timings are arranged so that the last five output of each FIFO is the scanning row of the image. At each rising edge of the clock signal, five pixels disappear and five new pixels appear (one pixel from each FIFO representing each line data), as five lines are stored at five FIFOs. So the scanning operation is performed. Convolution block receives the delayed line information, reads the coefficients from the coefficient RAM and performs the convolution operation. The coefficients are read at the correct time instants (at start of each field) when the RAM is not used for image processing purposes. Each line is also delayed in order to have 5 pixels at the same time from each line. Each coefficient is multiplied with the corresponding pixel value. In order to have a fast operation output, sum of each line is first calculated. And then the sum of lines are added together. This operation shortens the delay due to the reduced addition operation. If we need faster operation we should add two products at each time. This would perform a faster operation, but meantime it would increase the time needed to have the first pixel output. In our case, first output is generated at the 8th clock rising edge, second and third pixel values are generated at the following clock rising edges (9th and 10th clock rising edges). This can be seen at the simulation part (Table 3-7). The hardware structure of the convolution block is explained in Figure 3-4. The VHDL code for these blocks are presented at Appendix A.

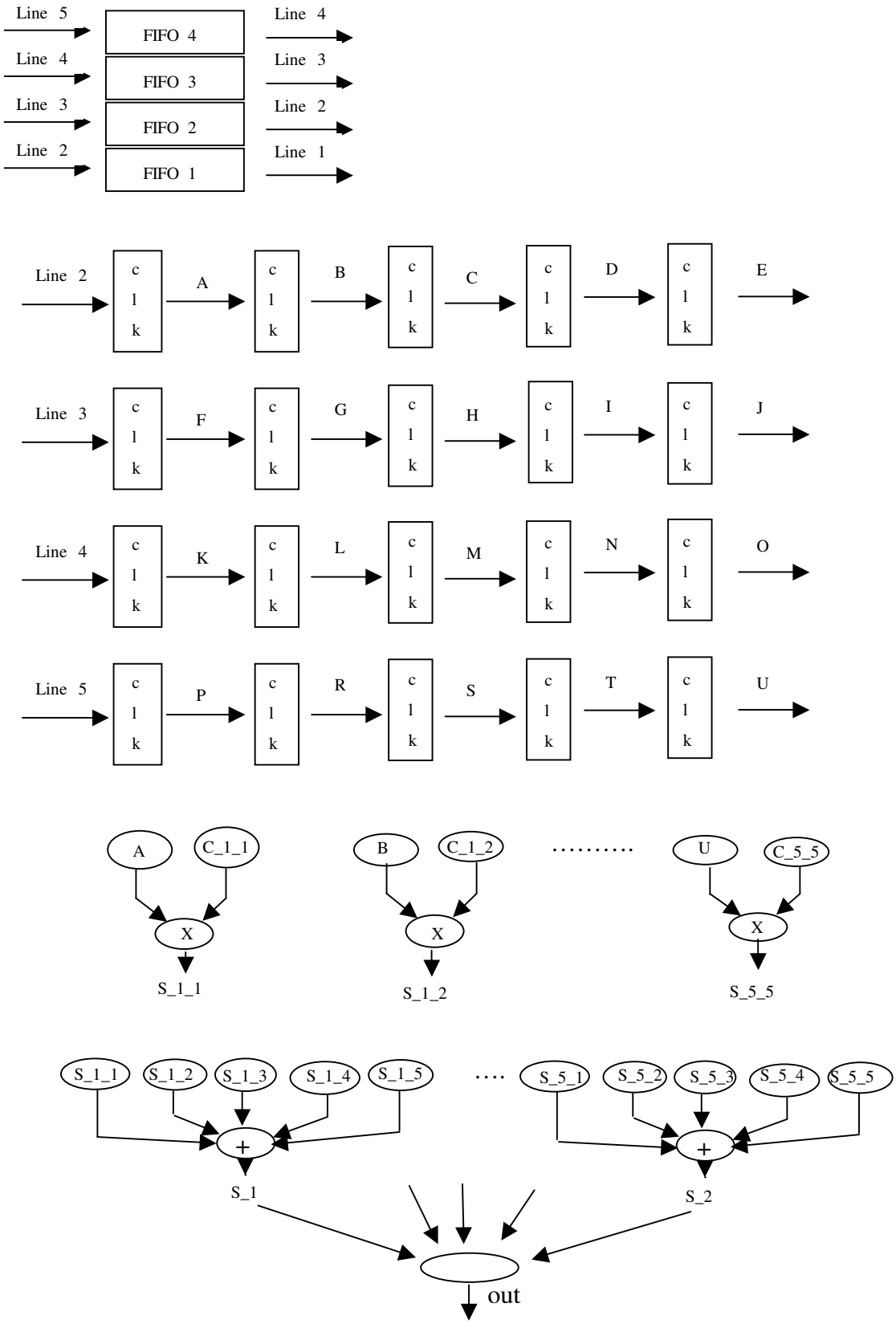

Figure 3-4 FIFO Implementation Convolution Block

3.3 Implementation

Several filtering applications can be performed by the system. The user interface program can choose the filter type through a combo box. Corresponding filter coefficients are sent to the camera. The output image, which is produced after convolution is sent to the display. As always, in order to have the original image an impulse response filter is applied. In order to show the edge effects, the Laplacian, LOG, Sobel, and the Prewitt operators can be selected. Apart from these operators, any filter can be applied by user interface program as the filter coefficients can be modified manually.

3.3.1 The Impulse Response

The impulse response of the system is the original video signal. The default filter coefficients are set to the impulse response coefficients and the original image is introduced. The impulse response filter coefficients are seen in Table 3-1.

Table 3-1 Impulse Response Filter Coefficients

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

The input and output image in case of the impulse response filter is shown in Figure 3-5 and Figure 3-6.

Figure 3-5 Input of the Impulse Response Filter

Figure 3-6 Output of the Impulse Response Filter Same as Input

3.3.2 Laplacian Filter

The Laplacian operator is a high-pass operator. It is useful to examine the edges in the image. Laplacian filter coefficients can be seen in Table 3-2.

Table 3-2 Laplacian Filter Coefficients

0,01	0,01	0,01	0,01	0,01
0,01	0,1667	0,6667	0,1667	0,01
0,01	0,1667	-3	0,1667	0,01
0,01	0,1667	0,6667	0,1667	0,01
0,01	0,01	0,01	0,01	0,01

The input and output image in case of a Laplacian filter is shown at Figure 3-7 and Figure 3-8.

Figure 3-7 Input of the Laplacian Filter

Figure 3-8 Output of the Laplacian Filter Showing the Edges at the Input Image

3.3.3 The LOG Filter

The LOG operator consists of two operators. First a Gaussian operation (low pass) is performed and then a laplacian (high pass) operation is applied. The Gaussian operation reduces the noise in the image as it is a low-pass operation and the Laplacian operation enhances the edge structures in image. The LOG filter coefficients are shown in Table 3-3. The input and output image for a LOG filter is shown in Figure 3-9 and Figure 3-10.

Table 3-3 LOG Filter Coefficients

0,0448	0,0468	00,0564	0,0468	0,0448
0,0468	0,3167	0,7146	0,3167	0,0468
0,0564	0,7146	-4,9048	0,7146	0,0564
0,0468	0,3167	0,7146	0,3167	0,0468
0,0448	0,0468	00,0564	0,0468	0,0448

Figure 3-9 Input of the LOG Filter

Figure 3-10 Output of the LOG Filter Showing the Edges at the Input Image

3.3.4 The Sobel Filter

The Sobel operator is a high-pass filter operator. It can show the edges in the image. The Sobel filter coefficients are shown in Table 3-4.

Table 3-4 Sobel Filter Coefficients

0	0	0	0	0
0	1	0	-1	0
0	2	0	-2	0
0	1	0	-1	0
0	0	0	0	0

The input and output images of a Sobel operator is shown at Figure 3-11 and Figure 3-12.

Figure 3-11 Input image of the Sobel Filter

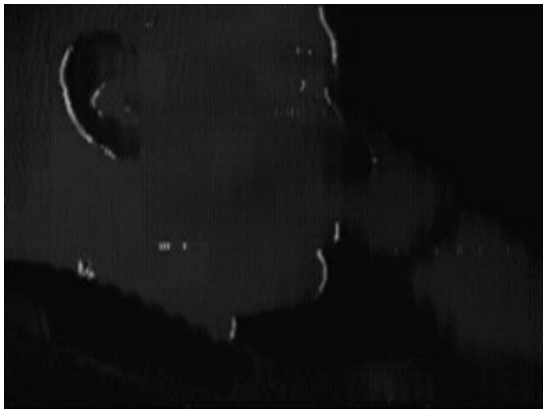

Figure 3-12 Output image of the Sobel Filter Showing the Edges at the Input Image

3.3.5 The Prewitt Filter

The Prewitt operator is another high-pass operator, it is nearly same with sobel operator. It is also used to see the edge effects at the image. Prewitt filter coefficients are shown in Table 3-5. The input and output images for the Prewitt filter is shown at Figure 3-13 and Figure 3-14.

Table 3-5 Prewitt Filter Coefficients

0	0	0	0	0
0	1	0	-1	0
0	1	0	-1	0
0	1	0	-1	0
0	0	0	0	0

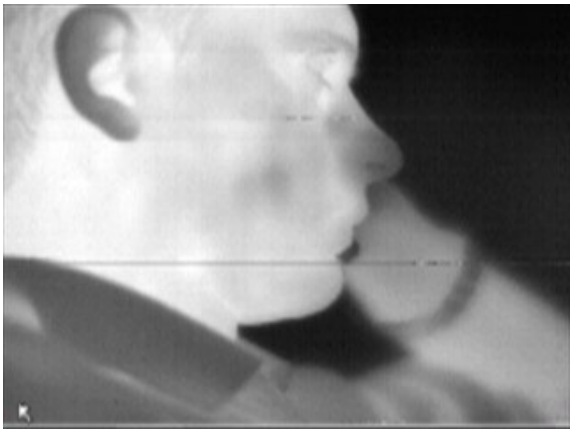

Figure 3-13 Input image of the Prewitt Filter

Figure 3-14 Output Image After the Prewitt Filter Showing the Edges at the Input Image

3.4 Simulations

Figure 3-15 and Figure 3-16 show the coefficients and the simulation results of the implementation. All sign signals are '0' apart from coefficient_3_3_sign and coefficient_5_1_sign, which are '1'. Which means that all the multiplication results will be added, only the multiplication result at the intersection of the third row and

the third column and intersection of the fifth row and the first column will be subtracted. Figure 3-16 shows the coefficients and the convolution input and output. The detailed explanation of the operation is presented by Table 3-6 and Table 3-7. Coefficients are 18 bits. First bit is the sign bit, following four bits are the decimal part of the number, and the following bits are the rational part of the coefficient. Table 3-6 shows the simulated coefficients and their rational equivalents.

Table 3-6 Coefficients

coefficient number	sign	coefficient value	rational value
1_1	0	00000000000000000	0
1_2	0	00000000000000000	0
1_3	0	00010000000000000	1
1_4	0	00100000000000000	2
1_5	0	00000000000000000	0
2_1	0	00000000000000000	0
2_2	0	00000000000000000	0
2_3	0	00000000000000000	0
2_4	0	00000000000000000	0
2_5	0	00000000000000000	0
3_1	0	00000000000000000	0
3_2	0	00000000000000000	0
3_3	1	00010000000000000	-1
3_4	0	00000000000000000	0
3_5	0	00000000000000000	0
4_1	0	00000000000000000	0
4_2	0	00000000000000000	0
4_3	0	00000000000000000	0
4_4	0	00000000000000000	0
4_5	0	00000000000000000	0

Table 3-6 (cont'd)

5_1	1	00000100000000000	-0,25
5_2	0	00000000000000000	0
5_3	0	00000000000000000	0
5_4	0	00000000000000000	0
5_5	0	00000000000000000	0

The above table shows that coefficient_1_3, coefficient_1_4, coefficient_3_3 and coefficient_5_1 are non-zero. When we consider the values and signs, for the first output pixel value, two times first lines fourth pixel value will be added, first lines third pixel value will be added, third lines third pixel value will be subtracted and one fourth of the fifth lines first pixel value will be subtracted. Table 3-7 shows the inputs and outputs of the convolution.

Table 3-7 Inputs and Outputs of Convolution

line1	8309	8264	8257	8260	8256	8309	8204	8193	8266	8192	8288	8192
line2	8266	8243	8269	8195	8280	8197	8315	8251	8248	8376	8235	8305
line3	8273	8309	8258	8314	8199	8297	8220	8306	8307	8264	8309	8239
line4	8270	8277	8271	8260	8311	8235	8308	8292	8272	8282	8215	8303
line5	8208	8260	8294	8282	8291	8289	8208	8193	8260	8261	8381	8264
output									14522	14484	14605	14343

Output1=8282*2+8294-8258-8309/4=14522,75

Output2=8291*2+8282-8314-8264/4=14484

Output3=8289*2+8291-8199-8257/4=14605

Output4=8208*2+8289-8297-8260/4=14343

These results are also shown at Figure 3-16. Simulation results show that convolution block results are successful.

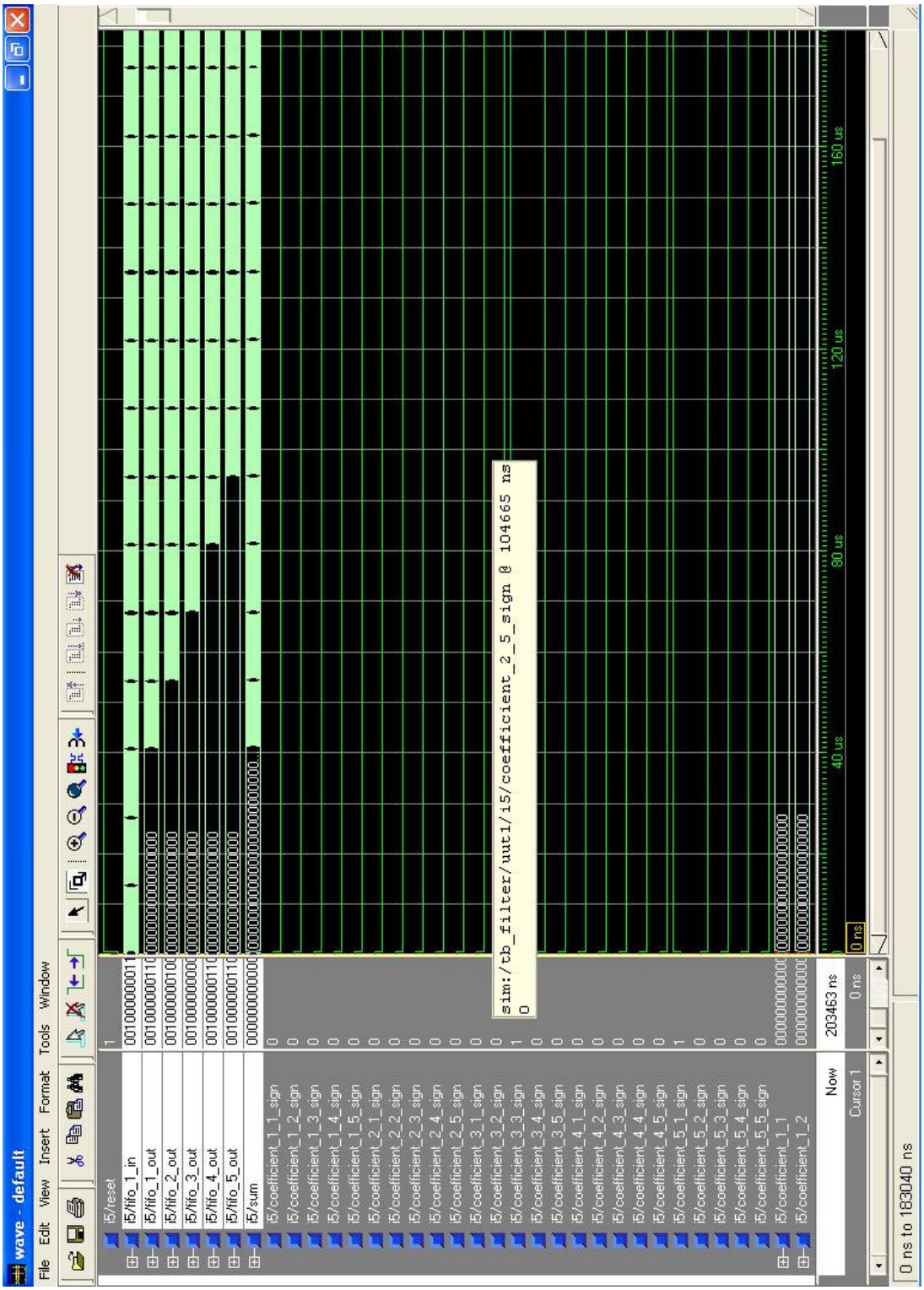

Figure 3-15 Line Delays and Coefficient Signs for Convolution Operation

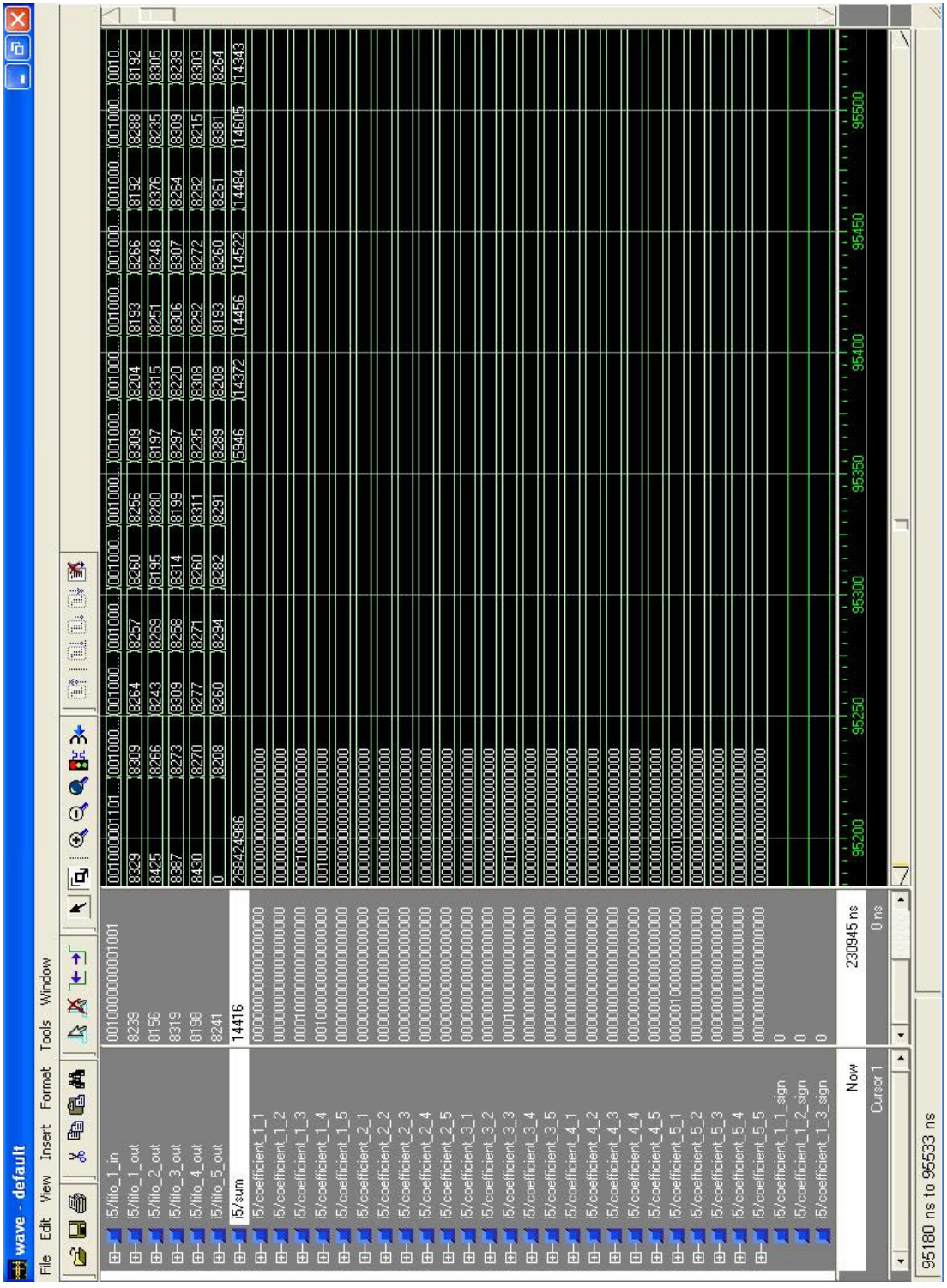

Figure 3-16 Coefficients and Convolution Operation Input and Output Signals

CHAPTER 4

TEMPERATURE MEASUREMENT

4.1 Introduction

Infrared imaging is the remote sensing of infrared radiation emitted from objects. The collected energy is displayed on the screen. The true temperature is possible with specific detector types. Microbolometers are one of the detectors that can be used to measure the true temperature. However, some detector types, such as pyroelectrics, cannot be used to measure true temperature. In this work we have used a microbolometer that let us to have a DC bias and give a prediction about the absolute temperature. The radiation emission physics is explained in the Formulation section. For a true temperature measurement, a calibration process has to be performed for the system. The graphs and the calibration process is explained in the Temperature calibration and data collection section. The system blocks implemented to measure absolute temperatures are explained in the Structure part of this chapter.

4.2 Formulation

Any object whose temperature is above 0 ˚K radiates infrared energy. The amount of radiated energy is a function of the object's temperature and its relative efficiency of thermal radiation, namely, its *emissivity*. The amount of radiated thermal power is determined by the following equation:

$$W = E * B * T^4 \quad \text{Watt cm}^{-2} \qquad (1)$$

where W is the spectral radiant exitance (radiation), E is the emissivity, B is the Stefan Boltzmann Constant ($5.67x10^{-12}$ Watt cm^{-2} $°K^{-4}$), and T is the Temperature (°K).

Radiated power is proportional to the body's temperature, raised to the 4th power. The emissivity value is a distinct property for each material and it changes from object to object. For biomedical applications, human skin emissivity is an important parameter. The emissivity value of the human skin is nearly 0.98. Hence, in some papers human skin is called a perfect radiator (The emissivity value of a perfect radiator is 1.).

In the system designed for this thesis work, the incoming radiation is scaled and shifted for non-uniformity correction purposes, and also due to input requirements of the ADC used in the system. For these reasons, the input data is shifted and scaled. Hence, the above equation has to be modified before calculations, that is

$$W'=A_1*W+B_1 \quad \text{Watt cm}^{-2}$$

due to ADC requirements where A_1 and B_1 are constants,

$$W''=A_2*W+B_2 \quad \text{Watt cm}^{-2}$$

due to normalization requirements. Here A_1,B_1 and A_2, B_2 are constants to be determined.
Hence, the overall equation is

$$W''=A_2*(A_1*E*B*T^4+B_1)+B_2 \quad \text{Watt cm}^{-2}$$
$$W''=A_2*A_1*E*B*T^4+A_2*B_1+B_2 \quad \text{Watt cm}^{-2}$$

As B and E are constant for same kind of objects then

$$W''=A_3*T^4+B_3 \quad \text{Watt cm}^{-2} \qquad (2)$$

Thus absolute temperature can be calculated as

$$T=((W''-B_3)/A_3)^{1/4} \quad °K \qquad (3)$$

Calculation of A_3 and B_3 are explained at the Calibration and Data Collection section.

4.3 Temperature Calibration and Data Collection

To perform a calibration process, a *blackbody* is used. A blackbody is a system, whose surface temperature can be adjusted manually. For the manually adjusted temperature value, the output surface of the blackbody becomes uniform. Using such a test system, measurements are obtained for temperatures in the range of 19 °C to 50 °C (292 °K to 323 °K). The measurement results, taken from the blackbody system, are shown at Figure 4-1.

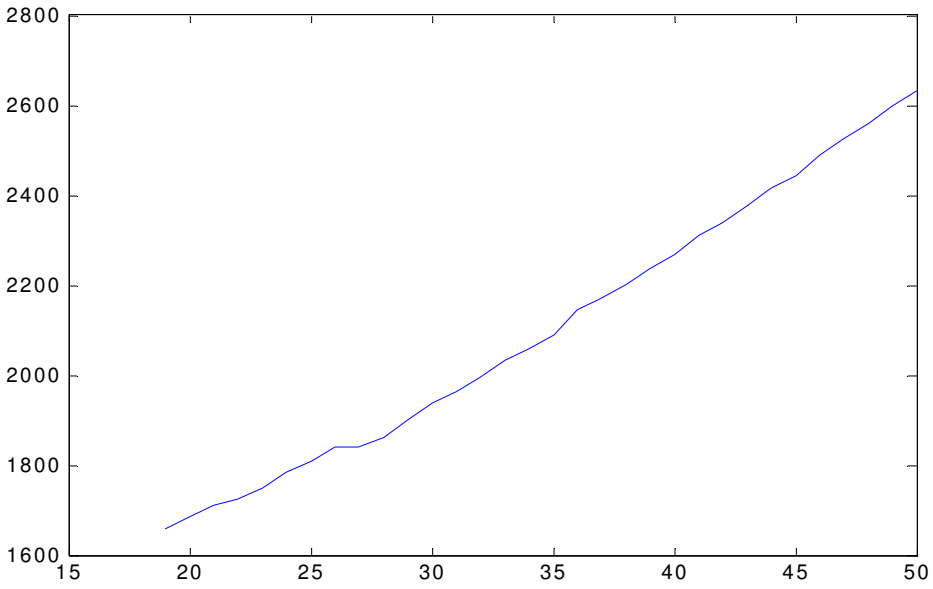

Figure 4-1 Calibration Data Average

Figure 4-1 is the data taken in order to calculate the coefficients in equation (2). Equation (2) is a fourth order equation with two unknowns. So, we need two equations in order to calculate the unknowns. We have chosen 19 °C and 50 °C for the calculation:

$W''(19)=1657$ and $W''(50)=2634$

$W''(T_1)=A_3*(T_1+273)^4+B_3$

$W''(T_2)=A_3*(T_2+273)^4+B_3$

$A_3=(W''(T_2)-W''(T_1))/((T_2+273)^4-(T_1+273)^4)=2{,}70293408*10^{-7}$

$B_3=W''(T_2)-A_3*(T_2+273)^4=308$

Hence the true temperature can be calculated as

$T=(W''-308)/\ 2{,}70293408*10^{-7}$ in Kelvins

4.4 Test of Equipment

The value that a thermograph can measure has to be tested before it is used for biomedical applications. As it is explained at part 1.1.3 there are threshold values for biomedical applications. In a number of studies, 500˚ mK was chosen as a threshold value, while some others use greater threshold values. To check whether the developed system satisfies this condition, a temperature measurement test is perfromed. The test is done using a system with two blackbodies, one of which has a hole on the surface. The temperature of the first blackbody is assigned to a value. The temperature of the second black body (with the hole) is adjusted to a temperature slightly different from the first blackbody and the image is examined. First blackbody is seen through the hole of the second blackbody. The image taken from the test system shows that infrared camera can show temperature differences lower than 500˚ mK. This verifies that the uncooled infrared camera developed during this thesis study can be used for biomedical purposes. Figure 4-2 shows the image with

temperature difference of 200˚ mK, and Figure 4-3 shows the image with temperature difference of 500˚ mK.

Figure 4-2 Temperature Difference of 200 mK Between Two Black Bodies

Figure 4-3 Temperature Difference of 500 mK Between Two Black Bodies

A second test is also performed to see whether the infrared imaging system can measure the true temperature values. For that test, the blackbody system at different

environments are used. Temperatures from 20°C to 40°C are measured. The results of this test is shown in Table 4-1 and Figure 4-4.

Table 4-1 Measured Temperatures

Black Body	measured temp	measured temp	measured temp	measured temp
20	24,2	23,8	16,9	21,4
21	25,2	24,8	17,9	22,4
22	26,1	25,9	18,8	23,4
23	27,2	26,9	19,8	24,5
24	28,2	27,8	20,9	25,5
25	29,1	28,8	21,9	26,5
26	30,1	29,8	22,9	27,4
27	31,1	30,9	23,9	28,4
28	32,2	31,9	24,8	29,6
29	33,2	32,8	25,9	30,5
30	34,2	33,8	27	31,4
31	35,1	34,8	27,9	32,5
32	36,2	35,9	28,9	33,4
33	37,2	36,8	29,9	34,4
34	38,2	37,9	30,8	35,5
35	39,2	38,8	31,9	36,6
36	40,3	39,8	32,9	37,4
37	41,3	40,8	33,9	38,4
38	42,2	42	34,8	39,5
39	43,2	42,8	35,9	40,5
40	44,2	43,9	36,9	41,5

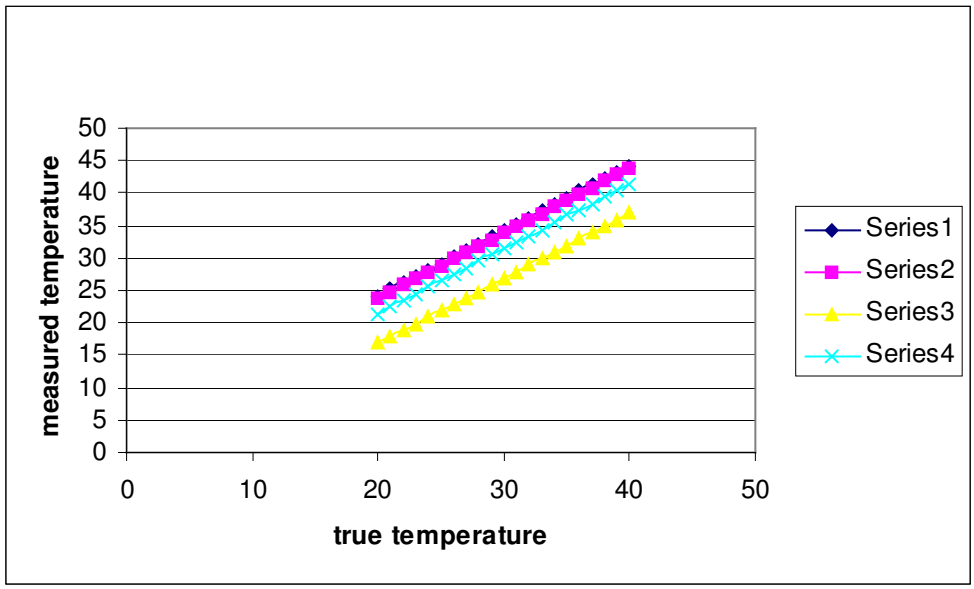

Figure 4-4 Measured Temperature

Results show that the measured temperature is relatively correct. It shows that the output changes when the environment changes, but the measured temperature is only shifted, but not scaled. Once the system is calibrated according for different environments, it should give the correct temperature values.

4.5 Structure

Temperature measurement is done by means of an interface program. The interface program sends the position of the measurement point through the RS232 channel with 115 200 baud rate. An FPGA implementation is done in order to get the position and send the measured temperature. To understand the position a “mouse image” is also implemented on the scene.

When the user moves the mouse on the user interface program, the position of the mouse is sent to the camera. Camera moves the implemented mouse image to the position sent from the interface program. When a “click” event occurs on the

interface program, the measured temperature is sent to the interface program. The structure of these events is explained in Figure 4-5.

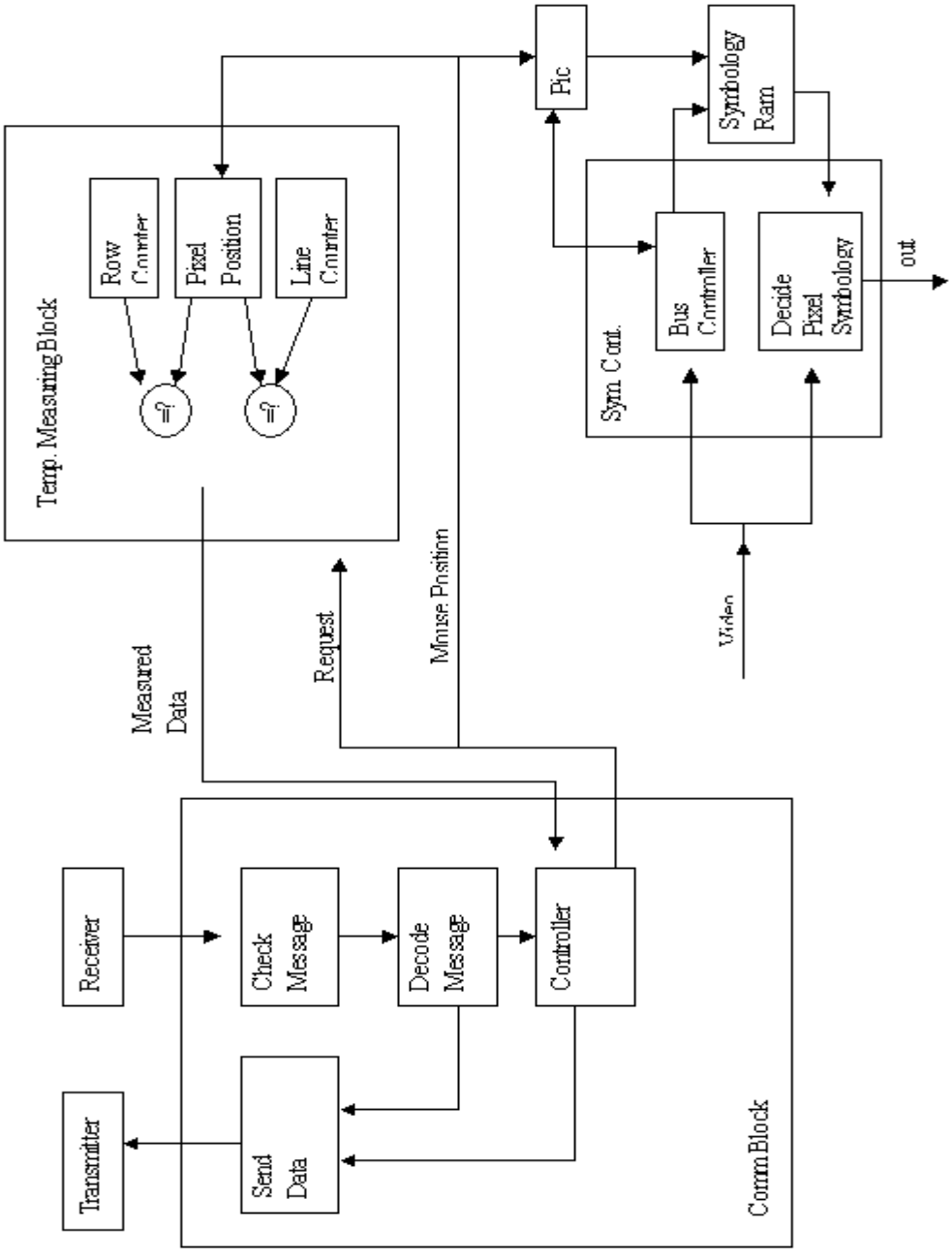

Figure 4-5 Temperature Measurement Structure Block Diagram

4.5.1 RS232 Communication Block

RS232 communication block receives the commands sent from the interface program. The same communication block, which is designed for filter coefficients, is used for the temperature measurement blocks.

The receiver block decodes the incoming bytes and sends them to the controller block. The controller block decodes the command, checks if it is a real command or not, and enables transmitter block to send whether the command received is valid or not. If the command is a valid command it decodes the position of the mouse and send this position to the pic and the temperature-measuring block. The mouse image is written to a symbology RAM by means of PIC. As, PIC is not fast enough, only address and data of the mouse image is adjusted by the PIC. The bus control signals of the symbology RAM are controlled by the symbology controller block at the FPGA.

When a temperature measurement command is received the controller block requests a measurement from the temperature measuring block and sends an acknowledge to the block when it receives the measurement. Then the controller block enables the transmitter block to send the measured temperature. The temperature is also sent with the same protocol. A start byte, a reply command, the measured temperature and the checksum are sent to the interface program.

4.5.2 Mouse Implementation

Implementing a mouse image using FPGA is not a good approach, as an FPGA should be used for more intelligent works. Instead, addressing of the mouse image is implemented by a PIC and also with a symbology controller block at the FPGA. PIC introduces the correct data address bus for the symbology, according to the

coefficient information sent by the transmission controller block. Symbology controller block produces the RAM bus controller signals to write the symbology on the RAM. This RAM is a dual port RAM. This block also reads the symbology information from the RAM and combines it with the image and displays the mouse icon on the display. For any pixel of the icon, it decides whether to send a white image, black image or a transparent image.

4.5.3 Temperature-Measuring Block

Temperature-measuring block is enabled by the transmission controller block. It is enabled when a temperature measurement request is done. This block has two counters. One counter counts the columns and the other one counts the rows from the incoming video signal. When the requested position is reached, the measured temperature is sent to the transmission controller block and tells that the measured temperature is ready.

CHAPTER 5

CLINICAL APPLICATIONS

5.1 Introduction

For biomedical applications of the designed uncooled infrared imaging system, data is collected at the Ankara Numune Hospital. Data taken from different patients are examined. For clinical applications, a black body is also taken to the hospital. Before the clinical applications, the system is calibrated to obtain true temperature measurements.

First case was a patient with rheumatism in her hands and legs. Second one was a breast cancer patient after surgery. The third case was also a breast cancer patient. In the fourth case, the blood pressure is purposely changed using a measuring device and the resulting temperature differences are recorded.

Patients are examined with the Direct Infrared Imaging method, as uncooled system is only suitable for this method. Due to uncontrollable experiment environment we could not examine using the Tau Imaging Method and due to system's limited RAM sources we could not implement DAT Imaging Method.

5.2 Case I

The first case was a patient with rheumatism in her hands and legs. As the problem is symmetric, there was no difference with respect to asymmetry analysis. Right and left hands were nearly the same, but there was a temperature increase at both hands. The measured temperature values were approximately 38°C. Also there was an

other region at the right hand of the patient with higher temperature value, namely 39,4°C. This part is the most problematic part at this hand. Figure 5-1 and Figure 5-2 show the measurements taken from two wrists.

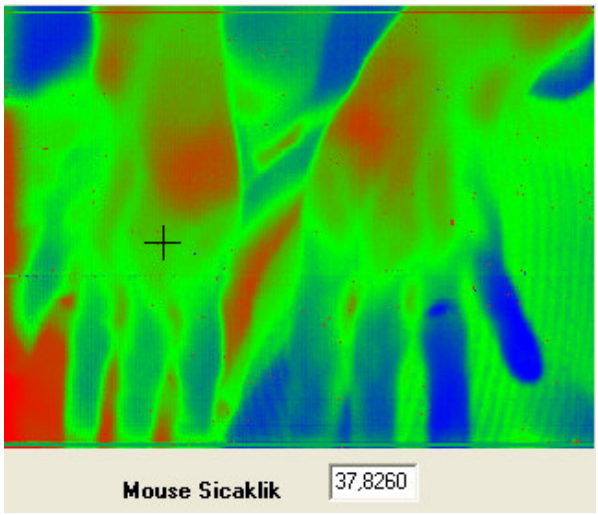

Figure 5-1 Right Wrist Temperature Measurement

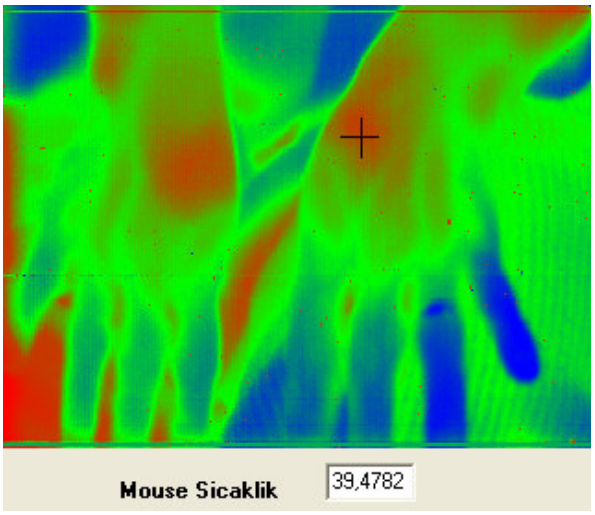

Figure 5-2 Left Wrist Temperature Measurement

The same patient has a problem in her legs, too. Both legs have problems, however, an asymmetry can be observed at the legs. Temperature values were also measured on the two legs (Figure 5-3 and Figure 5-4).

Figure 5-3 Right Leg Temperature Measurement

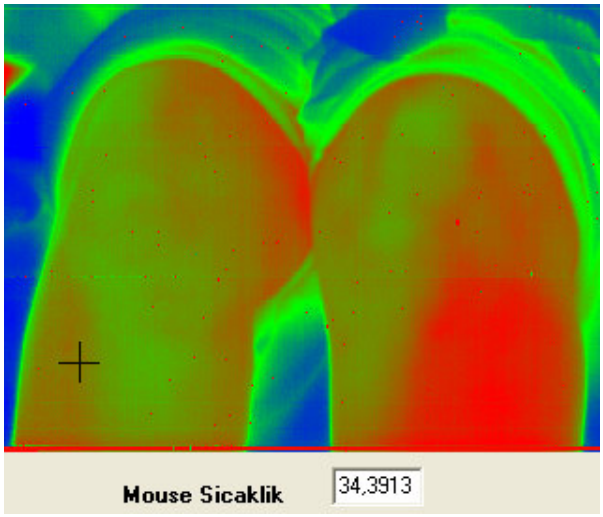

Figure 5-4 Left Leg Temperature Measurement

5.3 Case 2

The second patient was a woman after breast cancer surgery. There was a problem with the surgery area. Thus, temperature measurements were specifically obtained for these areas. Figure 5-5 and Figure 5-6 show the temperature values of the surgery and non-surgery areas.

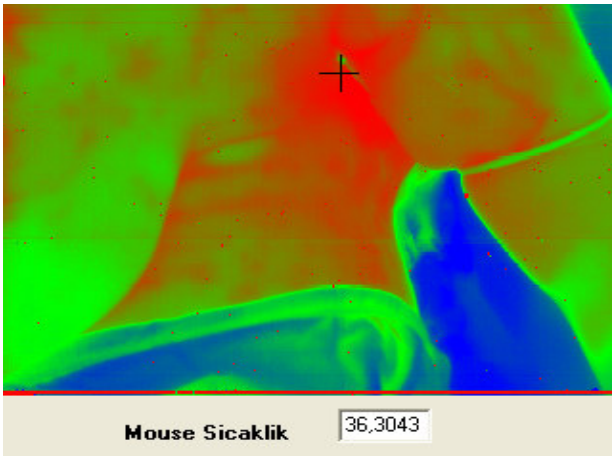

Figure 5-5 Temperature Measurement of the Surgery Area

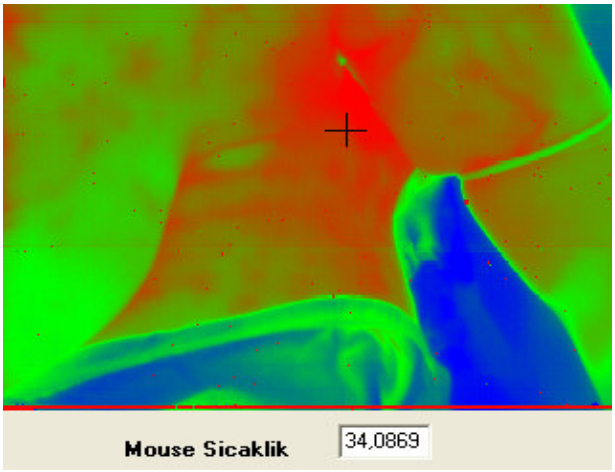

Figure 5-6 Temperature Measurement of the Non-Surgery Area

5.4 Case 3

In this case, the right breast of the patient has a cancer risk. However, the temperature of the risk area is measured cooler with respect to the normal side. The biopsy result of the patient is not reached yet. The lower temperature measurement may be due to non-heating tumor inside the breast or a skin problem. Temperature measurement of the breast with cancer risk is shown in Figure 5-7 and the temperature measurement of the normal breast is shown in Figure 5-8.

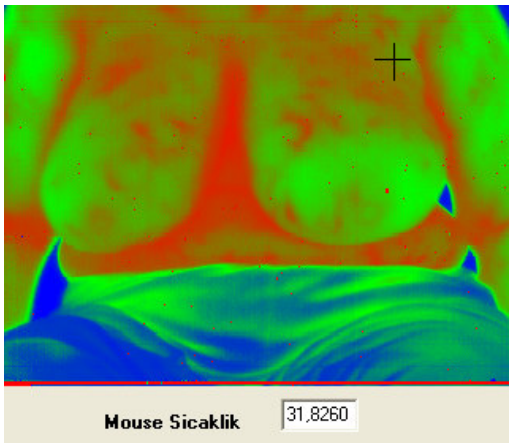

Figure 5-7 Temperature of the Breast with Cancer Risk

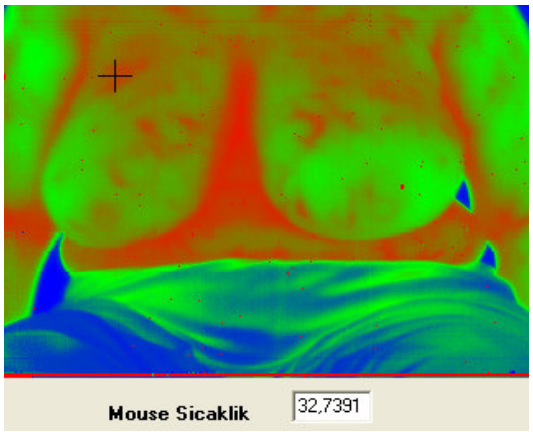

Figure 5-8 Temperature of the Normal Breast

5.5 Case 4

In this case, the temperature values the patient's arms are measured. In order to introduce a change, one arm is fastened by a blood temperature measuring device and temperature is measured. Then the arm is released and its temperature is measured again. Quick temperature changes are observed at the finger tips. These steps can be seen at Figure 5-9, Figure 5-10 and Figure 5-11.

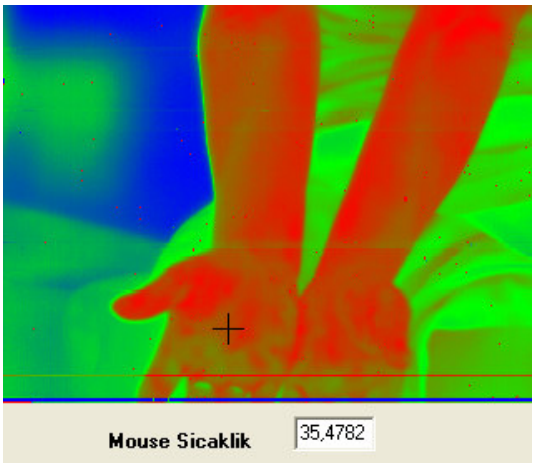

Figure 5-9 Normal Arm Temperature

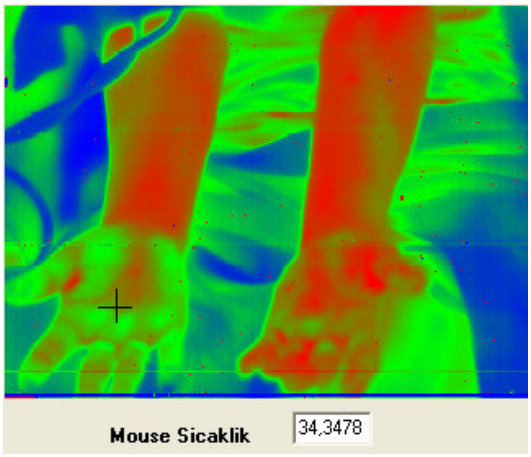

Figure 5-10 Fastened Arm Temperature

Figure 5-11 Released Arm Temperature

5.3 Handicaps of the System

Direct Imaging Method is based on temperature measurement of the object. Hence system is affected by anything affecting the temperature of the object. So measurements must be taken with great care. Object should be taken to the measurement environment 20 minutes before the test in order to obtain correct measurements. Clothes should not be let to heat symmetric parts of the body. Patient should not be let to touch the measurement area in order not to increase the temperature.

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this thesis study an uncooled medical infrared imaging system is designed and implemented. Results show that this system can be used for biomedical applications. A user interface program was also designed to control the imaging system. Work done throughout this study can be summarized as below:

- The output of the detector data is not suitable for imaging applications as it is. In order to obtain an image, non-uniformity correction algorithms are implemented.
- A histogram equization algorithm is implemented in order to show the details of the object.
- A convolution operation is also implemented. An interface program is also designed to control the filter outputs.
- Medical systems require 500°mK measurement sensitivity. Black body tests have shown that system can be used for biomedical applications.
- System is tested in the Ankara Numune Hospital for four cases.. The outputs are saved by means of the interface program.

Results show that different applications of the uncooled microbolometer for biomedical purposes should be investigated. In this thesis study, we could only consider the Direct Infrared Imaging Method. Tau Imaging Method and the DAT Method should be considered as a future work.

In order to consider the DAT Method, some changes in the system are required. This system is designed for a microbolometer. As DAT Method requires a more sensitive dedector, design and implementation of a front end electronic is needed. There will be no system change need for calibration, normalization and image processing algorithms used in the digital part. System is capable of storing images. As DAT

analysis needs storing 1000 images in 10 seconds. A bigger RAM should be replaced with the one employed in this design and can be used in order to store images.

The system needs to be calibrated for different environments. Actually system has a flag in front of the detector. As flag's temperature is not known, calibration is made by a black body. By using a thermistor, flag's temperature can be measured and real time calibration can be made possible.

The tests at the hospital showed that the data storage time is too long. In a revised system, a USB interface should be used for the data transfer.

REFERENCES

[1] J. R. Keyserlingk, P.D. Ahlgren, E. Yu, N. Belliveau, M. Yassa, "Functional Infrared Imaging of the Breast", 2000

[2] Jonathan F. Head, Fen Wang, Charles A. Lipari, Robert L. Elliot, "The Important Role of Infrared Imaging in Breast Cancer", 2000

[3] Jonatthan F. Head, Robert L. Helliot, "Infrared Imaging: Making Progres in fulfilling Its Medical Promise", 2002

[4] Michael Anbar, Cheryl Brown, Lorin Milescu, John Babalola, Laura Gentner, "The Potential of Dynamic Area Telethermography in Assessing Breast Cancer", 2000

[5] Michael Anbar, "Modalities and Clinical Applications of Dynamic Infrared Imaging", 2001

[6] Michael Anbar, Lorin Milescu, Cheryl Brown, Aleksey Naumov, Emily Bachman, Khaldoon AlDulaimi, Christine Geronimo, Terry Button, "Diagnosis of Breast Cancer with Infrared Dynamic Area Telethermography(DAT)", 2000

[7] Hisasi Usuki, Hajimr Matea, Hisao Wakabayashi, Fuminori Goda, Yukihiko Karasawa, Atsushi Misawa, Seiji Mori, Keichi Okano, "Stardardization of Thermographic Brest Cancer Dedection- Role of Finding Qualitative Finding And Quantative Findings", 2000

[8] Junji Wakamiya, Kunuhiko Mabuchi, Iwao Fujimasa, Shinichi Nakagawa, Hitshi Miyake, Kimiyoshi Arimura, Mitsuhito Osame, Akihiro Igata, Yukio

Takizawa, “Data-processing Method for Standardisation of Thermographic Diagnosis”, 2000

[9] A. Meria, L. Di Donota, G. L. Romani, “Tau Image: A diagnostic Imaging Technique Based on the Dynamic Digital Thermography”, 2000

[10] Hisashi Usuki, Tadashi Ikeda, Yoshiaki Igarashi, Takao Yokoe, Hiroshi Sonoo, Kazuaki Asaishi, Hisaki Fukushima, “Efficacy of Thermographic Examination for Minimum Breast Cancer”, 2001

[11] C. L. Herry, M. Frize, “Digital Processing Techiniques for the Assesment of Pain with Infrared Thermal Imaging”, 2002

[12] A. S. Boyd, S. K. Maloney, “Digital Infrared Thermal Imaging As Biofeedback Tool: Monitoring Chemotherophy Response in a young Female with Breast Cancer Mediastinal Secondaries”, 2002

[13] Yasuhiko Ohashi, Isao Uchida, “Applying Dynamic Thermography In the Diagnosis of Breast Cancer”, 2000

[14] Jong Keyserlingk, Paul Ahlgren, Maram Yassa, Normand Belliveau, “Overwiew of Functional Imaging as a Part of a multi-imaging strategy for breast Cancer Dedection and Therapeutic Monitoring”, 2002

[15] A. Merla, V. Romano, F. Zulli, R. Saggini, L. Di Donato, G. L. Romani, “Total Body Infrared Imaging and Postural Disorders”, 2002

[16] Monique Frize, Christophe Herry, Roger Roberge, “Processing of Thermal Images to Dedect Breast Cancer: Comparison with Previous Work”, 2002

[17] Phani Teje Kuruganti, Hairong Qi, “Asymetry analysis in Breast Cancer Dedection Using ThermalInfrared Images”, 2002

[18] A. Nowakowski, M. Kaczmarek, J. Ruminski, “Syntetic Pictures in thermographic Diagnosis”, 2002

[19] J. F. Head, C.A. Lipari, R. L. Elliot, “Detection of Mean Temperatures of Normal Whole Breast and Breast quadrants by Infrared Imaging and Image Analysis”, 2001

[20] A. Nowakowski, M. Kaczmarek, J. Wtorek, J. Siebert, D. Jagilak, K. Roszak, J. Topolewics, W. Stojek, “Thermographic and Electrical Measurements for Cardiac Surgery Inspectation”, 2001

[21] Arcangelo Merla, Luigi Di Donato, Silvano Di Luzio, Gian Luca Romani, “Quantifying the Relevance and Stage of Disease with Tau Imagee Technique”, 2002

[22] J. F. Head, C.A. Lipary, R. L. Elliot, “Determination of Mean Temperatures of Normal Whole Breast and Breast Quadrants by Infrared Imaging and Image Analysis”, 2002

[23] Meditherm, “Indications for a Thermographic Evaluation”, http://www.meditherm.com/therm_page3.htm

[24] Thermal Camera Corp., “Infrared Thermography Applications”, http://www.thermalcamera.co.uk/thermography_applications.htm

[25] Academy of Infrared Training Inc., “Learn About Thermography”, http://www.infraredtraining.net/thermographer.htm

[26] Colbert Infrared Services Inc., “IR Info/ Facts”, http://www.colbert-infrared.com/ir_info_facts.htm

APPENDIX-A

VHDL Code for Filtering Block

```
--
-- VHDL Architecture fpga2.filter_controller.arch
--
-- Created:
--          by - kiziloz.YTSD_VHDL (kavak)
--          at - 17:20:41 12/10/04
--
-- using Mentor Graphics HDL Designer(TM) 2003.3 (Build 60)
--
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;

ENTITY filter_controller IS
  PORT(
    line_1_dout_read       : IN     std_logic_vector (17 DOWNTO 0);
    line_1_addr_read       : OUT    std_logic_VECTOR (9 DOWNTO 0);
    line_1_addr_write      : OUT    std_logic_VECTOR (9 DOWNTO 0);
    line_1_clk_read        : OUT    std_logic;
    line_1_clk_write       : OUT    std_logic;
    line_1_din_write       : OUT    std_logic_VECTOR (17 DOWNTO 0);
    line_1_en_write        : OUT    std_logic;
    line_1_en_read         : OUT    std_logic;
    line_1_we_write        : OUT    std_logic;
    line_1_we_read         : OUT    std_logic;
    line_2_dout_read       : IN     std_logic_vector (17 DOWNTO 0);
    line_2_addr_read       : OUT    std_logic_VECTOR (9 DOWNTO 0);
    line_2_addr_write      : OUT    std_logic_VECTOR (9 DOWNTO 0);
    line_2_clk_read        : OUT    std_logic;
    line_2_clk_write       : OUT    std_logic;
    line_2_din_write       : OUT    std_logic_VECTOR (17 DOWNTO 0);
    line_2_en_write        : OUT    std_logic;
    line_2_en_read         : OUT    std_logic;
    line_2_we_write        : OUT    std_logic;
    line_2_we_read         : OUT    std_logic;
    line_3_dout_read       : IN     std_logic_vector (17 DOWNTO 0);
    line_3_addr_read       : OUT    std_logic_VECTOR (9 DOWNTO 0);
    line_3_addr_write      : OUT    std_logic_VECTOR (9 DOWNTO 0);
    line_3_clk_read        : OUT    std_logic;
    line_3_clk_write       : OUT    std_logic;
    line_3_din_write       : OUT    std_logic_VECTOR (17 DOWNTO 0);
    line_3_en_write        : OUT    std_logic;
    line_3_en_read         : OUT    std_logic;
```

```
    line_3_we_write         : OUT    std_logic;
    line_3_we_read          : OUT    std_logic;
    line_4_dout_read        : IN     std_logic_vector (17 DOWNTO 0);
    line_4_addr_read        : OUT    std_logic_VECTOR (9 DOWNTO 0);
    line_4_addr_write       : OUT    std_logic_VECTOR (9 DOWNTO 0);
    line_4_clk_read         : OUT    std_logic;
    line_4_clk_write        : OUT    std_logic;
    line_4_din_write        : OUT    std_logic_VECTOR (17 DOWNTO 0);
    line_4_en_write         : OUT    std_logic;
    line_4_en_read          : OUT    std_logic;
    line_4_we_write         : OUT    std_logic;
    line_4_we_read          : OUT    std_logic;
    line_5_dout_read        : IN     std_logic_vector (17 DOWNTO 0);
    line_5_addr_read        : OUT    std_logic_VECTOR (9 DOWNTO 0);
    line_5_addr_write       : OUT    std_logic_VECTOR (9 DOWNTO 0);
    line_5_clk_read         : OUT    std_logic;
    line_5_clk_write        : OUT    std_logic;
    line_5_din_write        : OUT    std_logic_VECTOR (17 DOWNTO 0);
    line_5_en_write         : OUT    std_logic;
    line_5_en_read          : OUT    std_logic;
    line_5_we_write         : OUT    std_logic;
    line_5_we_read          : OUT    std_logic;
    hblank                  : IN     std_logic;
    vblank                  : IN     std_logic;
    filter_coefficient_dout : IN     std_logic_vector (17 DOWNTO 0);
    filter_coefficient_addr : OUT    std_logic_VECTOR (9 DOWNTO 0);
    filter_coefficient_clk  : OUT    std_logic;
    filter_coefficient_din  : OUT    std_logic_VECTOR (17 DOWNTO 0);
    filter_coefficient_en   : OUT    std_logic;
    filter_coefficient_we   : OUT    std_logic;
    image_data              : IN     std_logic_vector (15 DOWNTO 0);
    video_clk               : IN     std_logic;
    reset                   : IN     std_logic;
    sum                     : OUT    std_logic_vector (41 DOWNTO 0)
  );

-- Declarations

END filter_controller ;

--
ARCHITECTURE arch OF filter_controller IS



  signal line_1_dout_read_delay_1 : std_logic_vector(17 downto 0);
  signal line_1_dout_read_delay_2 : std_logic_vector(17 downto 0);
  signal line_1_dout_read_delay_3 : std_logic_vector(17 downto 0);
  signal line_1_dout_read_delay_4 : std_logic_vector(17 downto 0);
  signal line_2_dout_read_delay_1 : std_logic_vector(17 downto 0);
  signal line_2_dout_read_delay_2 : std_logic_vector(17 downto 0);
  signal line_2_dout_read_delay_3 : std_logic_vector(17 downto 0);
  signal line_2_dout_read_delay_4 : std_logic_vector(17 downto 0);
  signal line_3_dout_read_delay_1 : std_logic_vector(17 downto 0);
  signal line_3_dout_read_delay_2 : std_logic_vector(17 downto 0);
  signal line_3_dout_read_delay_3 : std_logic_vector(17 downto 0);
```

```
  signal line_3_dout_read_delay_4 : std_logic_vector(17 downto 0);
  signal line_4_dout_read_delay_1 : std_logic_vector(17 downto 0);
  signal line_4_dout_read_delay_2 : std_logic_vector(17 downto 0);
  signal line_4_dout_read_delay_3 : std_logic_vector(17 downto 0);
  signal line_4_dout_read_delay_4 : std_logic_vector(17 downto 0);
  signal line_5_dout_read_delay_1 : std_logic_vector(17 downto 0);
  signal line_5_dout_read_delay_2 : std_logic_vector(17 downto 0);
  signal line_5_dout_read_delay_3 : std_logic_vector(17 downto 0);
  signal line_5_dout_read_delay_4 : std_logic_vector(17 downto 0);


  signal multiplier_1_1_out  : std_logic_vector(35 downto 0);
  signal multiplier_1_2_out  : std_logic_vector(35 downto 0);
  signal multiplier_1_3_out  : std_logic_vector(35 downto 0);
  signal multiplier_1_4_out  : std_logic_vector(35 downto 0);
  signal multiplier_1_5_out  : std_logic_vector(35 downto 0);
  signal multiplier_2_1_out  : std_logic_vector(35 downto 0);
  signal multiplier_2_2_out  : std_logic_vector(35 downto 0);
  signal multiplier_2_3_out  : std_logic_vector(35 downto 0);
  signal multiplier_2_4_out  : std_logic_vector(35 downto 0);
  signal multiplier_2_5_out  : std_logic_vector(35 downto 0);
  signal multiplier_3_1_out  : std_logic_vector(35 downto 0);
  signal multiplier_3_2_out  : std_logic_vector(35 downto 0);
  signal multiplier_3_3_out  : std_logic_vector(35 downto 0);
  signal multiplier_3_4_out  : std_logic_vector(35 downto 0);
  signal multiplier_3_5_out  : std_logic_vector(35 downto 0);
  signal multiplier_4_1_out  : std_logic_vector(35 downto 0);
  signal multiplier_4_2_out  : std_logic_vector(35 downto 0);
  signal multiplier_4_3_out  : std_logic_vector(35 downto 0);
  signal multiplier_4_4_out  : std_logic_vector(35 downto 0);
  signal multiplier_4_5_out  : std_logic_vector(35 downto 0);
  signal multiplier_5_1_out  : std_logic_vector(35 downto 0);
  signal multiplier_5_2_out  : std_logic_vector(35 downto 0);
  signal multiplier_5_3_out  : std_logic_vector(35 downto 0);
  signal multiplier_5_4_out  : std_logic_vector(35 downto 0);
  signal multiplier_5_5_out  : std_logic_vector(35 downto 0);

-- coefficient_read signals

  signal coefficient_1_1 : std_logic_vector(17 downto 0);
  signal coefficient_1_2 : std_logic_vector(17 downto 0);
  signal coefficient_1_3 : std_logic_vector(17 downto 0);
  signal coefficient_1_4 : std_logic_vector(17 downto 0);
  signal coefficient_1_5 : std_logic_vector(17 downto 0);
  signal coefficient_2_1 : std_logic_vector(17 downto 0);
  signal coefficient_2_2 : std_logic_vector(17 downto 0);
  signal coefficient_2_3 : std_logic_vector(17 downto 0);
  signal coefficient_2_4 : std_logic_vector(17 downto 0);
  signal coefficient_2_5 : std_logic_vector(17 downto 0);
  signal coefficient_3_1 : std_logic_vector(17 downto 0);
  signal coefficient_3_2 : std_logic_vector(17 downto 0);
  signal coefficient_3_3 : std_logic_vector(17 downto 0);
  signal coefficient_3_4 : std_logic_vector(17 downto 0);
  signal coefficient_3_5 : std_logic_vector(17 downto 0);
  signal coefficient_4_1 : std_logic_vector(17 downto 0);
  signal coefficient_4_2 : std_logic_vector(17 downto 0);
  signal coefficient_4_3 : std_logic_vector(17 downto 0);
```

```
  signal coefficient_4_4 : std_logic_vector(17 downto 0);
  signal coefficient_4_5 : std_logic_vector(17 downto 0);
  signal coefficient_5_1 : std_logic_vector(17 downto 0);
  signal coefficient_5_2 : std_logic_vector(17 downto 0);
  signal coefficient_5_3 : std_logic_vector(17 downto 0);
  signal coefficient_5_4 : std_logic_vector(17 downto 0);
  signal coefficient_5_5 : std_logic_vector(17 downto 0);
  signal state_coef_read     : std_logic_vector(4 downto 0);


-- assign sync signals  ------------------------------------------------------


  constant wait_between_hblank_counter_limit : std_logic_vector(8 downto 0):="000011111";
  constant hblank_counter_limit      : std_logic_vector(8 downto 0):="111111111";
  constant wait_between_hblank       : std_logic_vector(2 downto 0):="000";
  constant wait_for_vblank           : std_logic_vector(2 downto 0):="001";
  constant wait_for_5line            : std_logic_vector(2 downto 0):="010";
  constant latch_hblank_vblank       : std_logic_vector(2 downto 0):="011";
  constant send_5line                : std_logic_vector(2 downto 0):="100";
  constant count_hblank_between_hblanks      : std_logic_vector(1 downto 0):="00";
  constant count_hblank_wait_for_hblank_high : std_logic_vector(1 downto 0):="01";
  constant count_hblank_wait_for_hblank_low  : std_logic_vector(1 downto 0):="10";

  signal vblank_5line_delay          : std_logic;
  signal hblank_5line_delay          : std_logic;
  signal hblank_5line_delay_delay1   : std_logic;
  signal hblank_counter              : std_logic_vector(8 downto 0);  -- max dec320=0x140
  signal number_hblank               : std_logic_vector(2 downto 0);  -- to count 5 hblank
  signal assign_sync_state           : std_logic_vector(2 downto 0);
  signal next_assign_sync_state      : std_logic_vector(2 downto 0);
  signal vblank_5line_delay_buf      : std_logic;
  signal next_number_hblank          : std_logic_vector(2 downto 0);
  signal number_hblank_state         : std_logic_vector(1 downto 0);

--------------------------------------------------------------------------------
-- add multiplication signals--------------------------------------------------
  signal line_1_sum                  : std_logic_vector(38 downto 0);
  signal line_2_sum                  : std_logic_vector(38 downto 0);
  signal line_3_sum                  : std_logic_vector(38 downto 0);
  signal line_4_sum                  : std_logic_vector(38 downto 0);
  signal line_5_sum                  : std_logic_vector(38 downto 0);
--  signal sum                        : std_logic_vector(41 downto 0);
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-- read_line_data signals------------------------------------------------------
  signal hblank_line_counter         : std_logic_vector(8 downto 0);  -- max dec320=0x140
  signal ram_1_dout_read             : std_logic_vector(17 downto 0);
  signal ram_2_dout_read             : std_logic_vector(17 downto 0);
  signal ram_3_dout_read             : std_logic_vector(17 downto 0);
  signal ram_4_dout_read             : std_logic_vector(17 downto 0);
  signal ram_5_dout_read             : std_logic_vector(17 downto 0);
  signal line_1_read                 : std_logic_vector(17 downto 0);
  signal line_2_read                 : std_logic_vector(17 downto 0);
  signal line_3_read                 : std_logic_vector(17 downto 0);
  signal line_4_read                 : std_logic_vector(17 downto 0);
  signal line_5_read                 : std_logic_vector(17 downto 0);
```

```
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-- multiplex_line_data signals-------------------------------------------------
--  signal line_1_read              : std_logic_vector(17 downto 0);
--  signal line_2_read              : std_logic_vector(17 downto 0);
--  signal line_3_read              : std_logic_vector(17 downto 0);
--  signal line_4_read              : std_logic_vector(17 downto 0);
--  signal line_5_read              : std_logic_vector(17 downto 0);
  signal number_hblank_delay1       : std_logic_vector(2 downto 0);
  signal number_hblank_delay2       : std_logic_vector(2 downto 0);
  signal number_hblank_delay3       : std_logic_vector(2 downto 0);
  signal number_hblank_delay4       : std_logic_vector(2 downto 0);
  signal number_hblank_delay5       : std_logic_vector(2 downto 0);
--------------------------------------------------------------------------------
-- assign_write_line_data_address signals --------------------------------------
  signal address_counter_write_data : std_logic_vector(8 downto 0);
  signal address_counter_write_data_delay1 : std_logic_vector(8 downto 0);
  signal address_counter_write_data_delay2 : std_logic_vector(8 downto 0);
  signal hblank_delay1              : std_logic;
  signal hblank_delay2              : std_logic;
  signal hblank_delay3              : std_logic;
  signal hblank_delay4              : std_logic;
  signal hblank_delay5              : std_logic;
  signal hblank_delay6              : std_logic;
  signal image_data_delay1          : std_logic_vector(15 downto 0);
  signal image_data_delay2          : std_logic_vector(15 downto 0);
  signal image_data_delay3          : std_logic_vector(15 downto 0);
  signal image_data_delay4          : std_logic_vector(15 downto 0);
  signal image_data_delay5          : std_logic_vector(15 downto 0);
  signal image_data_delay6          : std_logic_vector(15 downto 0);
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
  component multiplier18
            port (
            clk: IN std_logic;
            a: IN std_logic_VECTOR(17 downto 0);
      b: IN std_logic_VECTOR(17 downto 0);
      o: OUT std_logic_VECTOR(35 downto 0));
  end component;

BEGIN

delay_line_data:process(video_clk,reset)

begin

  if reset='0' then

    line_1_dout_read_delay_1    <= (others=>'0');
    line_1_dout_read_delay_2    <= (others=>'0');
    line_1_dout_read_delay_3    <= (others=>'0');
    line_1_dout_read_delay_4    <= (others=>'0');
    line_2_dout_read_delay_1    <= (others=>'0');
    line_2_dout_read_delay_2    <= (others=>'0');
    line_2_dout_read_delay_3    <= (others=>'0');
    line_2_dout_read_delay_4    <= (others=>'0');
    line_3_dout_read_delay_1    <= (others=>'0');
```

```
    line_3_dout_read_delay_2    <= (others=>'0');
    line_3_dout_read_delay_3    <= (others=>'0');
    line_3_dout_read_delay_4    <= (others=>'0');
    line_4_dout_read_delay_1    <= (others=>'0');
    line_4_dout_read_delay_2    <= (others=>'0');
    line_4_dout_read_delay_3    <= (others=>'0');
    line_4_dout_read_delay_4    <= (others=>'0');
    line_5_dout_read_delay_1    <= (others=>'0');
    line_5_dout_read_delay_2    <= (others=>'0');
    line_5_dout_read_delay_3    <= (others=>'0');
    line_5_dout_read_delay_4    <= (others=>'0');

  elsif Rising_edge(video_clk) then

    line_1_dout_read_delay_1    <= line_1_read;
    line_1_dout_read_delay_2    <= line_1_dout_read_delay_1;
    line_1_dout_read_delay_3    <= line_1_dout_read_delay_2;
    line_1_dout_read_delay_4    <= line_1_dout_read_delay_3;
    line_2_dout_read_delay_1    <= line_2_read;
    line_2_dout_read_delay_2    <= line_2_dout_read_delay_1;
    line_2_dout_read_delay_3    <= line_2_dout_read_delay_2;
    line_2_dout_read_delay_4    <= line_2_dout_read_delay_3;
    line_3_dout_read_delay_1    <= line_3_read;
    line_3_dout_read_delay_2    <= line_3_dout_read_delay_1;
    line_3_dout_read_delay_3    <= line_3_dout_read_delay_2;
    line_3_dout_read_delay_4    <= line_3_dout_read_delay_3;
    line_4_dout_read_delay_1    <= line_4_read;
    line_4_dout_read_delay_2    <= line_4_dout_read_delay_1;
    line_4_dout_read_delay_3    <= line_4_dout_read_delay_2;
    line_4_dout_read_delay_4    <= line_4_dout_read_delay_3;
    line_5_dout_read_delay_1    <= line_5_read;
    line_5_dout_read_delay_2    <= line_5_dout_read_delay_1;
    line_5_dout_read_delay_3    <= line_5_dout_read_delay_2;
    line_5_dout_read_delay_4    <= line_5_dout_read_delay_3;

  end if;
end process;


filter_coefficient_clk   <= video_clk;
read_coefficients : process(video_clk,reset)

begin
  if reset='0' then
--    coefficient_1_1        <= (others=>'1');
--    coefficient_1_2        <= (others=>'1');
--    coefficient_1_3        <= (others=>'1');
--    coefficient_1_4        <= (others=>'1');
--    coefficient_1_5        <= (others=>'1');
--    coefficient_2_1        <= (others=>'1');
--    coefficient_2_2        <= (others=>'1');
--    coefficient_2_3        <= (others=>'1');
--    coefficient_2_4        <= (others=>'1');
--    coefficient_2_5        <= (others=>'1');
--    coefficient_3_1        <= (others=>'1');
--    coefficient_3_2        <= (others=>'1');
--    coefficient_3_3        <= (others=>'1');
```

```
--    coefficient_3_4        <= (others=>'1');
--    coefficient_3_5        <= (others=>'1');
--    coefficient_4_1        <= (others=>'1');
--    coefficient_4_2        <= (others=>'1');
--    coefficient_4_3        <= (others=>'1');
--    coefficient_4_4        <= (others=>'1');
--    coefficient_4_5        <= (others=>'1');
--    coefficient_5_1        <= (others=>'1');
--    coefficient_5_2        <= (others=>'1');
--    coefficient_5_3        <= (others=>'1');
--    coefficient_5_4        <= (others=>'1');
--    coefficient_5_5        <= (others=>'1');
    coefficient_1_1        <= (others=>'0');
    coefficient_1_2        <= (others=>'0');
    coefficient_1_3        <= "0000000000000000001";
    coefficient_1_4        <= "0000000000000000001";
    coefficient_1_5        <= (others=>'0');
    coefficient_2_1        <= (others=>'0');
    coefficient_2_2        <= (others=>'0');
    coefficient_2_3        <= (others=>'0');
    coefficient_2_4        <= (others=>'0');
    coefficient_2_5        <= (others=>'0');
    coefficient_3_1        <= (others=>'0');
    coefficient_3_2        <= (others=>'0');
    coefficient_3_3        <= "0000000000000000001";
    coefficient_3_4        <= (others=>'0');
    coefficient_3_5        <= (others=>'0');
    coefficient_4_1        <= (others=>'0');
    coefficient_4_2        <= (others=>'0');
    coefficient_4_3        <= (others=>'0');
    coefficient_4_4        <= (others=>'0');
    coefficient_4_5        <= (others=>'0');
    coefficient_5_1        <= (others=>'0');
    coefficient_5_2        <= (others=>'0');
    coefficient_5_3        <= (others=>'0');
    coefficient_5_4        <= (others=>'0');
    coefficient_5_5        <= (others=>'0');
    state_coef_read            <= (others=>'0');
    filter_coefficient_addr <= (others=>'1');
    filter_coefficient_en   <= '1';
    filter_coefficient_we   <= '1';

  elsif rising_edge(video_clk) then
    case state_coef_read is
      when "00000" =>
        filter_coefficient_addr <= (others=>'1');
        filter_coefficient_en   <= '1';
        filter_coefficient_we   <= '1';
        if ((vblank_5line_delay='0')and(vblank='0')) then
          state_coef_read       <= state_coef_read + 1;
        end if;

      when "00001" =>
        filter_coefficient_addr <= ("00000"&state_coef_read);
        filter_coefficient_en   <= '0';
        filter_coefficient_we   <= '1';
        state_coef_read         <= state_coef_read + 1;
```

```
when "00010" =>
  filter_coefficient_addr<= "00000"&state_coef_read;
  state_coef_read       <= state_coef_read + 1;
  coefficient_1_1   <= filter_coefficient_dout;

when "00011" =>
  filter_coefficient_addr<= "00000"&state_coef_read;
  state_coef_read       <= state_coef_read + 1;
  coefficient_1_2   <= filter_coefficient_dout;

when "00100" =>
  filter_coefficient_addr<= "00000"&state_coef_read;
  state_coef_read       <= state_coef_read + 1;
  coefficient_1_3   <= filter_coefficient_dout;

when "00101" =>
  filter_coefficient_addr<= "00000"&state_coef_read;
  state_coef_read       <= state_coef_read + 1;
  coefficient_1_4   <= filter_coefficient_dout;

when "00110" =>
  filter_coefficient_addr<= "00000"&state_coef_read;
  state_coef_read       <= state_coef_read + 1;
  coefficient_1_5   <= filter_coefficient_dout;

when "00111" =>
  filter_coefficient_addr<= "00000"&state_coef_read;
  state_coef_read       <= state_coef_read + 1;
  coefficient_2_1   <= filter_coefficient_dout;

when "01000" =>
  filter_coefficient_addr<= "00000"&state_coef_read;
  state_coef_read       <= state_coef_read + 1;
  coefficient_2_2   <= filter_coefficient_dout;

when "01001" =>
  filter_coefficient_addr<= "00000"&state_coef_read;
  state_coef_read       <= state_coef_read + 1;
  coefficient_2_3   <= filter_coefficient_dout;

when "01010" =>
  filter_coefficient_addr<= "00000"&state_coef_read;
  state_coef_read       <= state_coef_read + 1;
  coefficient_2_4   <= filter_coefficient_dout;

when "01011" =>
  filter_coefficient_addr<= "00000"&state_coef_read;
  state_coef_read       <= state_coef_read + 1;
  coefficient_2_5   <= filter_coefficient_dout;

when "01100" =>
  filter_coefficient_addr<= "00000"&state_coef_read;
  state_coef_read       <= state_coef_read + 1;
  coefficient_3_1   <= filter_coefficient_dout;

when "01101" =>
```

```
  filter_coefficient_addr<= "00000"&state_coef_read;
  state_coef_read       <= state_coef_read + 1;
  coefficient_3_2   <= filter_coefficient_dout;

when "01110" =>
  filter_coefficient_addr<= "00000"&state_coef_read;
  state_coef_read       <= state_coef_read + 1;
  coefficient_3_3   <= filter_coefficient_dout;

when "01111" =>
  filter_coefficient_addr<= "00000"&state_coef_read;
  state_coef_read       <= state_coef_read + 1;
  coefficient_3_4   <= filter_coefficient_dout;

when "10000" =>
  filter_coefficient_addr<= "00000"&state_coef_read;
  state_coef_read       <= state_coef_read + 1;
  coefficient_3_5   <= filter_coefficient_dout;

when "10001" =>
  filter_coefficient_addr<= "00000"&state_coef_read;
  state_coef_read       <= state_coef_read + 1;
  coefficient_4_1   <= filter_coefficient_dout;

when "10010" =>
  filter_coefficient_addr<= "00000"&state_coef_read;
  state_coef_read       <= state_coef_read + 1;
  coefficient_4_2   <= filter_coefficient_dout;

when "10011" =>
  filter_coefficient_addr<= "00000"&state_coef_read;
  state_coef_read       <= state_coef_read + 1;
  coefficient_4_3   <= filter_coefficient_dout;

when "10100" =>
  filter_coefficient_addr<= "00000"&state_coef_read;
  state_coef_read       <= state_coef_read + 1;
  coefficient_4_4   <= filter_coefficient_dout;

when "10101" =>
  filter_coefficient_addr<= "00000"&state_coef_read;
  state_coef_read       <= state_coef_read + 1;
  coefficient_4_5   <= filter_coefficient_dout;

when "10110" =>
  filter_coefficient_addr<= "00000"&state_coef_read;
  state_coef_read       <= state_coef_read + 1;
  coefficient_5_1   <= filter_coefficient_dout;

when "10111" =>
  filter_coefficient_addr<= "00000"&state_coef_read;
  state_coef_read       <= state_coef_read + 1;
  coefficient_5_2   <= filter_coefficient_dout;

when "11000" =>
  filter_coefficient_addr<= "00000"&state_coef_read;
  state_coef_read       <= state_coef_read + 1;
```

```
        coefficient_5_3    <= filter_coefficient_dout;

      when "11001" =>
        filter_coefficient_addr<= "00000"&state_coef_read;
        state_coef_read        <= state_coef_read + 1;
        coefficient_5_4    <= filter_coefficient_dout;

      when "11010" =>
        filter_coefficient_addr<= (others=>'1');
        state_coef_read        <= state_coef_read + 1;
        coefficient_5_5    <= filter_coefficient_dout;
        filter_coefficient_en  <= '1';
        filter_coefficient_we  <= '1';
      when "11011" =>
        filter_coefficient_addr <= (others=>'1');
        filter_coefficient_en   <= '1';
        filter_coefficient_we   <= '1';
        if ((vblank='1')) then
          state_coef_read       <= (others=>'0');
        end if;
      when others =>
        filter_coefficient_addr <= (others=>'1');
        filter_coefficient_en   <= '1';
        filter_coefficient_we   <= '1';
        if ((vblank='1')) then
          state_coef_read       <= (others=>'0');
        end if;

    end case;
  end if;
end process;



assign_sync: process(video_clk,reset)
begin
  if reset='0' then
    vblank_5line_delay          <= '0';
    hblank_5line_delay          <= '0';
    assign_sync_state           <= wait_for_vblank;
    number_hblank               <= (others=>'0');
    next_assign_sync_state      <= wait_for_vblank;
    vblank_5line_delay_buf      <= '1';
    next_number_hblank          <= (others=>'0');
    number_hblank_state         <= count_hblank_between_hblanks;
    hblank_5line_delay_delay1   <= '0';
  elsif rising_edge(video_clk) then
    hblank_5line_delay_delay1   <= hblank_5line_delay;
    case assign_sync_state is
      when wait_for_vblank =>
        if ((vblank='1')and(hblank='1')) then
          number_hblank         <= number_hblank + 1;
          assign_sync_state     <= wait_for_5line;
        end if;
      when wait_for_5line =>
        if number_hblank="101" then
          next_assign_sync_state<= latch_hblank_vblank;
```

```
    next_number_hblank    <= (others=>'0');
    number_hblank_state   <= count_hblank_between_hblanks;
   else
    next_assign_sync_state<= wait_for_vblank;
    next_number_hblank    <= number_hblank;
   end if;
   if ((vblank='1')and(hblank='0')) then
    assign_sync_state     <= next_assign_sync_state;
    number_hblank         <= next_number_hblank;
   end if;
  when latch_hblank_vblank =>
   hblank_5line_delay      <= hblank;
   vblank_5line_delay      <= vblank;
   if vblank='0' then
    assign_sync_state     <= wait_between_hblank;
    hblank_counter        <= (others=>'0');
    number_hblank         <= (others=>'0');
    vblank_5line_delay    <= '1';
   end if;
   case number_hblank_state is
    when count_hblank_between_hblanks =>
     if number_hblank = "101" then
      number_hblank     <= "001";
     else
      number_hblank     <= number_hblank +1;
     end if;
     number_hblank_state <= count_hblank_wait_for_hblank_high;
    when count_hblank_wait_for_hblank_high =>
     if hblank='1' then
      number_hblank_state<= count_hblank_wait_for_hblank_low;
     end if;
    when count_hblank_wait_for_hblank_low =>
     if hblank='0' then
      number_hblank_state<= count_hblank_between_hblanks;
     end if;
    when others =>
     number_hblank_state<= count_hblank_between_hblanks;
   end case;
  when wait_between_hblank =>
   hblank_counter          <= hblank_counter +1;
   hblank_5line_delay      <= '0';
   if hblank_counter=wait_between_hblank_counter_limit then
    hblank_counter        <= (others=>'0');
    number_hblank         <= number_hblank +1;
    assign_sync_state     <= send_5line;
   end if;
  when send_5line =>
   hblank_counter          <= hblank_counter + 1;
   hblank_5line_delay      <= '1';
   if number_hblank="101" then
    next_assign_sync_state<= wait_for_vblank;
    vblank_5line_delay_buf<= '0';
    next_number_hblank    <= (others =>'0');
   else
    next_assign_sync_state<= wait_between_hblank;
    vblank_5line_delay_buf<= '1';
        next_number_hblank    <= number_hblank;
```

```
        end if;
        if hblank_counter=hblank_counter_limit then
          hblank_counter        <= (others=>'0');
          assign_sync_state     <= next_assign_sync_state;
          vblank_5line_delay    <= vblank_5line_delay_buf;
          hblank_5line_delay    <= vblank_5line_delay_buf;
               number_hblank          <= next_number_hblank;
        end if;
       when others =>
        assign_sync_state       <= wait_for_vblank;

    end case;
  end if;
end process;

line_1_clk_read    <= video_clk;
line_2_clk_read    <= video_clk;
line_3_clk_read    <= video_clk;
line_4_clk_read    <= video_clk;
line_5_clk_read    <= video_clk;

read_line_data: process(video_clk,reset)
begin
  if reset='0' then
      line_1_addr_read        <= ( others=>'0' );
      line_1_en_read          <= '1';
      line_1_we_read          <= '1';
      line_2_addr_read        <= ( others=>'0' );
      line_2_en_read          <= '1';
      line_2_we_read          <= '1';
      line_3_addr_read        <= ( others=>'0' );
      line_3_en_read          <= '1';
      line_3_we_read          <= '1';
      line_4_addr_read        <= ( others=>'0' );
      line_4_en_read          <= '1';
      line_4_we_read          <= '1';
      line_5_addr_read        <= ( others=>'0' );
      line_5_en_read          <= '1';
      line_5_we_read          <= '1';
      hblank_line_counter     <= (others=>'0');
  elsif rising_edge(video_clk) then  -- adress portlari birlestirilebilir, we_pini disarida birlenebilir
    if vblank_5line_delay='1'then
      if hblank_5line_delay = '1' then
        hblank_line_counter   <= hblank_line_counter + 1;
        line_1_en_read        <= '0';
        line_1_addr_read      <= '0'&hblank_line_counter;
        ram_1_dout_read       <= line_1_dout_read;
        line_2_en_read        <= '0';
        line_2_addr_read      <= '0'&hblank_line_counter;
        ram_2_dout_read       <= line_2_dout_read;
        line_3_en_read        <= '0';
        line_3_addr_read      <= '0'&hblank_line_counter;
        ram_3_dout_read       <= line_3_dout_read;
        line_4_en_read        <= '0';
        line_4_addr_read      <= '0'&hblank_line_counter;
        ram_4_dout_read       <= line_4_dout_read;
        line_5_en_read        <= '0';
```

```
      line_5_addr_read     <= '0'&hblank_line_counter;
      ram_5_dout_read      <= line_5_dout_read;
     else
      line_1_en_read       <= '1';
      line_2_en_read       <= '1';
      line_3_en_read       <= '1';
      line_4_en_read       <= '1';
      line_5_en_read       <= '1';
      hblank_line_counter  <= (others=>'0');
     end if;
    else
     line_1_addr_read      <= ( others=>'0' );
     line_1_en_read        <= '1';
     line_1_we_read        <= '1';
     line_2_addr_read      <= ( others=>'0' );
     line_2_en_read        <= '1';
     line_2_we_read        <= '1';
     line_3_addr_read      <= ( others=>'0' );
     line_3_en_read        <= '1';
     line_3_we_read        <= '1';
     line_4_addr_read      <= ( others=>'0' );
     line_4_en_read        <= '1';
     line_4_we_read        <= '1';
     line_5_addr_read      <= ( others=>'0' );
     line_5_en_read        <= '1';
     line_5_we_read        <= '1';
     hblank_line_counter   <= (others=>'0');
    end if;
  end if;
end process;

  line_1_clk_write         <= video_clk;
  line_2_clk_write         <= video_clk;
  line_3_clk_write         <= video_clk;
  line_4_clk_write         <= video_clk;
  line_5_clk_write         <= video_clk;

assign_write_line_data_addr: process(video_clk,reset)
begin
  if reset='0' then
    line_1_addr_write      <= (others=>'0');
    line_5_addr_write      <= (others=>'0');
    line_4_addr_write      <= (others=>'0');
    line_3_addr_write      <= (others=>'0');
    line_2_addr_write      <= (others=>'0');
    hblank_delay1          <= '0';
    hblank_delay2          <= '0';
    hblank_delay3          <= '0';
    hblank_delay4          <= '0';
    hblank_delay5          <= '0';
    hblank_delay6          <= '0';
    image_data_delay1      <= (others=>'0');
    image_data_delay2      <= (others=>'0');
    image_data_delay3      <= (others=>'0');
    image_data_delay4      <= (others=>'0');
    image_data_delay5      <= (others=>'0');
    image_data_delay6      <= (others=>'0');
```

```
    address_counter_write_data<= (others=>'0');
--    address_counter_write_data_delay1 <= (others=>'0');
--    address_counter_write_data_delay2 <= (others=>'0');
  elsif rising_edge(video_clk) then
--    address_counter_write_data_delay1 <= address_counter_write_data;
--    address_counter_write_data_delay2 <= address_counter_write_data_delay1;
    hblank_delay1          <= hblank;
    hblank_delay2          <= hblank_delay1;
    hblank_delay3          <= hblank_delay2;
    hblank_delay4          <= hblank_delay3;
    hblank_delay5          <= hblank_delay4;
    hblank_delay6          <= hblank_delay5;
    image_data_delay1      <= image_data;
    image_data_delay2      <= image_data_delay1;
    image_data_delay3      <= image_data_delay2;
    image_data_delay4      <= image_data_delay3;
    image_data_delay5      <= image_data_delay4;
    image_data_delay6      <= image_data_delay5;
    line_1_addr_write      <= '0'&address_counter_write_data;
    line_2_addr_write      <= '0'&address_counter_write_data;
    line_3_addr_write      <= '0'&address_counter_write_data;
    line_4_addr_write      <= '0'&address_counter_write_data;
    line_5_addr_write      <= '0'&address_counter_write_data;
    if hblank_delay4 = '1' then
      address_counter_write_data <= address_counter_write_data +1;
    else
      address_counter_write_data <= (others=>'1');
    end if;
  end if;
end process;

multiplex_line_data: process(video_clk,reset)
begin
  if reset='0' then
    line_1_read            <= (others=>'0');
    line_2_read            <= (others=>'0');
    line_3_read            <= (others=>'0');
    line_4_read            <= (others=>'0');
    line_5_read            <= (others=>'0');
    line_1_din_write       <= (others=>'0');
    line_2_din_write       <= (others=>'0');
    line_3_din_write       <= (others=>'0');
    line_4_din_write       <= (others=>'0');
    line_5_din_write       <= (others=>'0');
    line_1_we_write        <= '1';
    line_2_we_write        <= '1';
    line_3_we_write        <= '1';
    line_4_we_write        <= '1';
    line_5_we_write        <= '1';
    line_1_en_write        <= '1';
    line_2_en_write        <= '1';
    line_3_en_write        <= '1';
    line_4_en_write        <= '1';
    line_5_en_write        <= '1';
    number_hblank_delay1   <= (others=>'0');
    number_hblank_delay2   <= (others=>'0');
    number_hblank_delay3   <= (others=>'0');
```

```
  number_hblank_delay4     <= (others=>'0');
  number_hblank_delay5     <= (others=>'0');
elsif rising_edge(video_clk) then
  number_hblank_delay1     <= number_hblank;
  number_hblank_delay2     <= number_hblank_delay1;
  number_hblank_delay3     <= number_hblank_delay2;
  number_hblank_delay4     <= number_hblank_delay3;
  number_hblank_delay5     <= number_hblank_delay4;
  if hblank_delay5 ='1' then
    case number_hblank_delay4 is
      when "001" =>
        line_1_read         <= line_1_dout_read;
        line_2_read         <= line_2_dout_read;
        line_3_read         <= line_3_dout_read;
        line_4_read         <= line_4_dout_read;
        line_5_read         <= line_5_dout_read;
        line_1_din_write    <= "00"&image_data_delay5;
        line_1_we_write      <= '0';
        line_2_we_write      <= '1';
        line_3_we_write      <= '1';
        line_4_we_write      <= '1';
        line_5_we_write      <= '1';
        line_1_en_write     <= '0';
        line_2_en_write     <= '1';
        line_3_en_write     <= '1';
        line_4_en_write     <= '1';
        line_5_en_write     <= '1';
      when "010" =>
        line_1_read         <= line_2_dout_read;
        line_2_read         <= line_3_dout_read;
        line_3_read         <= line_4_dout_read;
        line_4_read         <= line_5_dout_read;
        line_5_read         <= line_1_dout_read;
        line_2_din_write    <= "00"&image_data_delay5;
        line_1_we_write      <= '1';
        line_2_we_write      <= '0';
        line_3_we_write      <= '1';
        line_4_we_write      <= '1';
        line_5_we_write      <= '1';
        line_1_en_write     <= '1';
        line_2_en_write     <= '0';
        line_3_en_write     <= '1';
        line_4_en_write     <= '1';
        line_5_en_write     <= '1';
      when "011" =>
        line_1_read         <= line_3_dout_read;
        line_2_read         <= line_4_dout_read;
        line_3_read         <= line_5_dout_read;
        line_4_read         <= line_1_dout_read;
        line_5_read         <= line_2_dout_read;
        line_3_din_write    <= "00"&image_data_delay5;
        line_1_we_write      <= '1';
        line_2_we_write      <= '1';
        line_3_we_write      <= '0';
        line_4_we_write      <= '1';
        line_5_we_write      <= '1';
        line_1_en_write     <= '1';
```

```
    line_2_en_write     <= '1';
    line_3_en_write     <= '0';
    line_4_en_write     <= '1';
    line_5_en_write     <= '1';
   when "100" =>
    line_1_read         <= line_4_dout_read;
    line_2_read         <= line_5_dout_read;
    line_3_read         <= line_1_dout_read;
    line_4_read         <= line_2_dout_read;
    line_5_read         <= line_3_dout_read;
    line_4_din_write    <= "00"&image_data_delay5;
    line_1_we_write     <= '1';
    line_2_we_write     <= '1';
    line_3_we_write     <= '1';
    line_4_we_write     <= '0';
    line_5_we_write     <= '1';
    line_1_en_write     <= '1';
    line_2_en_write     <= '1';
    line_3_en_write     <= '1';
    line_4_en_write     <= '0';
    line_5_en_write     <= '1';
   when "101" =>
    line_1_read         <= line_5_dout_read;
    line_2_read         <= line_1_dout_read;
    line_3_read         <= line_2_dout_read;
    line_4_read         <= line_3_dout_read;
    line_5_read         <= line_4_dout_read;
    line_5_din_write    <= "00"&image_data_delay5;
    line_1_we_write     <= '1';
    line_2_we_write     <= '1';
    line_3_we_write     <= '1';
    line_4_we_write     <= '1';
    line_5_we_write     <= '0';
    line_1_en_write     <= '1';
    line_2_en_write     <= '1';
    line_3_en_write     <= '1';
    line_4_en_write     <= '1';
    line_5_en_write     <= '0';
   when others =>
    line_1_read         <= line_1_dout_read;
    line_2_read         <= line_2_dout_read;
    line_3_read         <= line_3_dout_read;
    line_4_read         <= line_4_dout_read;
    line_5_read         <= line_5_dout_read;
    line_5_din_write    <= "00"&image_data_delay5;
    line_1_we_write     <= '0';
    line_2_we_write     <= '1';
    line_3_we_write     <= '1';
    line_4_we_write     <= '1';
    line_5_we_write     <= '1';
    line_1_en_write     <= '0';
    line_2_en_write     <= '1';
    line_3_en_write     <= '1';
    line_4_en_write     <= '1';
    line_5_en_write     <= '1';
  end case;
end if;
```

```
  end if;
end process;


add_multiplications: process(video_clk,reset)
begin
  if reset='0' then
    line_1_sum    <= (others=>'0');
    line_2_sum    <= (others=>'0');
    line_3_sum    <= (others=>'0');
    line_4_sum    <= (others=>'0');
    line_5_sum    <= (others=>'0');
    sum           <= (others=>'0');
  elsif rising_edge(video_clk) then
    line_1_sum                  <=    ("000"&multiplier_1_1_out)    +    ("000"&multiplier_1_2_out)    +
("000"&multiplier_1_3_out) + ("000"&multiplier_1_4_out) + ("000"&multiplier_1_5_out);
    line_2_sum                  <=    ("000"&multiplier_2_1_out)    +    ("000"&multiplier_2_2_out)    +
("000"&multiplier_2_3_out) + ("000"&multiplier_2_4_out) + ("000"&multiplier_2_5_out);
    line_3_sum                  <=    ("000"&multiplier_3_1_out)    +    ("000"&multiplier_3_2_out)    +
("000"&multiplier_3_3_out) + ("000"&multiplier_3_4_out) + ("000"&multiplier_3_5_out);
    line_4_sum                  <=    ("000"&multiplier_4_1_out)    +    ("000"&multiplier_4_2_out)    +
("000"&multiplier_4_3_out) + ("000"&multiplier_4_4_out) + ("000"&multiplier_4_5_out);
    line_5_sum                  <=    ("000"&multiplier_5_1_out)    +    ("000"&multiplier_5_2_out)    +
("000"&multiplier_5_3_out) + ("000"&multiplier_5_4_out) + ("000"&multiplier_5_5_out);
    sum               <=    ("000"&line_1_sum)    +    ("000"&line_2_sum)    +    ("000"&line_3_sum)    +
("000"&line_4_sum) + ("000"&line_5_sum);
  end if;
end process;


M_1_1 : multiplier18 port map
            (       clk       => video_clk,
                    a         => line_1_dout_read_delay_4,
                    b         => coefficient_1_1,
                    o         => multiplier_1_1_out);



M_1_2 : multiplier18 port map
            (       clk       => video_clk,
                    a         => line_1_dout_read_delay_3,
                    b         => coefficient_1_2,
                    o         => multiplier_1_2_out);



M_1_3 : multiplier18 port map
            (       clk       => video_clk,
                    a         => line_1_dout_read_delay_2,
                    b         => coefficient_1_3,
                    o         => multiplier_1_3_out);



M_1_4 : multiplier18 port map
            (       clk       => video_clk,
                    a         => line_1_dout_read_delay_1,
```

```
                b       => coefficient_1_4,
                o       => multiplier_1_4_out);


M_1_5 : multiplier18 port map
        (       clk     => video_clk,
                a       => line_1_read,
                b       => coefficient_1_5,
                o       => multiplier_1_5_out);


M_2_1 : multiplier18 port map
        (       clk     => video_clk,
                a       => line_2_dout_read_delay_4,
                b       => coefficient_2_1,
                o       => multiplier_2_1_out);


M_2_2 : multiplier18 port map
        (       clk     => video_clk,
                a       => line_2_dout_read_delay_3,
                b       => coefficient_2_2,
                o       => multiplier_2_2_out);


M_2_3 : multiplier18 port map
        (       clk     => video_clk,
                a       => line_2_dout_read_delay_2,
                b       => coefficient_2_3,
                o       => multiplier_2_3_out);


M_2_4 : multiplier18 port map
        (       clk     => video_clk,
                a       => line_2_dout_read_delay_1,
                b       => coefficient_2_4,
                o       => multiplier_2_4_out);


M_2_5 : multiplier18 port map
        (       clk     => video_clk,
                a       => line_2_read,
                b       => coefficient_2_5,
                o       => multiplier_2_5_out);


M_3_1 : multiplier18 port map
        (       clk     => video_clk,
                a       => line_3_dout_read_delay_4,
                b       => coefficient_3_1,
```

```
                o       => multiplier_3_1_out);


M_3_2 : multiplier18 port map
        (       clk     => video_clk,
                a       => line_3_dout_read_delay_3,
                b       => coefficient_3_2,
                o       => multiplier_3_2_out);


M_3_3 : multiplier18 port map
        (       clk     => video_clk,
                a       => line_3_dout_read_delay_2,
                b       => coefficient_3_3,
                o       => multiplier_3_3_out);


M_3_4 : multiplier18 port map
        (       clk     => video_clk,
                a       => line_3_dout_read_delay_1,
                b       => coefficient_3_4,
                o       => multiplier_3_4_out);


M_3_5 : multiplier18 port map
        (       clk     => video_clk,
                a       => line_3_read,
                b       => coefficient_3_5,
                o       => multiplier_3_5_out);


M_4_1 : multiplier18 port map
        (       clk     => video_clk,
                a       => line_4_dout_read_delay_4,
                b       => coefficient_4_1,
                o       => multiplier_4_1_out);


M_4_2 : multiplier18 port map
        (       clk     => video_clk,
                a       => line_4_dout_read_delay_3,
                b       => coefficient_4_2,
                o       => multiplier_4_2_out);


M_4_3 : multiplier18 port map
        (       clk     => video_clk,
                a       => line_4_dout_read_delay_2,
                b       => coefficient_4_3,
                o       => multiplier_4_3_out);
```

```
M_4_4 : multiplier18 port map
        (       clk     => video_clk,
                a       => line_4_dout_read_delay_1,
                b       => coefficient_4_4,
                o       => multiplier_4_4_out);


M_4_5 : multiplier18 port map
        (       clk     => video_clk,
                a       => line_4_read,
                b       => coefficient_4_5,
                o       => multiplier_4_5_out);


M_5_1 : multiplier18 port map
        (       clk     => video_clk,
                a       => line_5_dout_read_delay_4,
                b       => coefficient_5_1,
                o       => multiplier_5_1_out);


M_5_2 : multiplier18 port map
        (       clk     => video_clk,
                a       => line_5_dout_read_delay_3,
                b       => coefficient_5_2,
                o       => multiplier_5_2_out);


M_5_3 : multiplier18 port map
        (       clk     => video_clk,
                a       => line_5_dout_read_delay_2,
                b       => coefficient_5_3,
                o       => multiplier_5_3_out);


M_5_4 : multiplier18 port map
        (       clk     => video_clk,
                a       => line_5_dout_read_delay_1,
                b       => coefficient_5_4,
                o       => multiplier_5_4_out);


M_5_5 : multiplier18 port map
        (       clk     => video_clk,
                a       => line_5_read,
                b       => coefficient_5_5,
                o       => multiplier_5_5_out);
```

```
END arch;


-- VHDL Entity fpga2.filter.symbol
--
-- Created:
--          by - kiziloz.YTSD_VHDL (kavak)
--          at - 18:12:02 12/23/04
--
-- Generated by Mentor Graphics' HDL Designer(TM) 2003.3 (Build 60)
--
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;

ENTITY filter IS
  PORT(
    hblank     : IN     std_logic;
    image_data : IN     std_logic_vector (15 DOWNTO 0);
    reset      : IN     std_logic;
    vblank     : IN     std_logic;
    video_clk  : IN     std_logic;
    sum        : OUT    std_logic_vector (41 DOWNTO 0);
           coef_en    : IN     std_logic;
           coef_we    : IN     std_logic;
           coef_addr  : IN     std_logic_vector (9 DOWNTO 0);
           coef_din   : IN     std_logic_vector (17 DOWNTO 0);
           coef_clk   : IN     std_logic
  );

-- Declarations

END filter ;

--
-- VHDL Architecture fpga2.filter.struct
--
-- Created:
--          by - kiziloz.YTSD_VHDL (kavak)
--          at - 18:12:03 12/23/04
--
-- Generated by Mentor Graphics' HDL Designer(TM) 2003.3 (Build 60)
--
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;


ARCHITECTURE struct OF filter IS

  -- Architecture declarations

  -- Internal signal declarations
  SIGNAL addrb                 : std_logic_VECTOR(9 DOWNTO 0);
  SIGNAL clkb                  : std_logic;
```

```
SIGNAL dina                 : std_logic_VECTOR(17 DOWNTO 0);
SIGNAL dina1                : std_logic_VECTOR(17 DOWNTO 0);
SIGNAL dina2                : std_logic_VECTOR(17 DOWNTO 0);
SIGNAL dina3                : std_logic_VECTOR(17 DOWNTO 0);
SIGNAL dina4                : std_logic_VECTOR(17 DOWNTO 0);
SIGNAL dinb                 : std_logic_VECTOR(17 DOWNTO 0);
SIGNAL enb                  : std_logic;
SIGNAL filter_coefficient_addr : std_logic_VECTOR(9 DOWNTO 0);
SIGNAL filter_coefficient_clk  : std_logic;
SIGNAL filter_coefficient_din  : std_logic_VECTOR(17 DOWNTO 0);
SIGNAL filter_coefficient_dout : std_logic_vector(17 DOWNTO 0);
SIGNAL filter_coefficient_en   : std_logic;
SIGNAL filter_coefficient_we   : std_logic;
SIGNAL line_1_addr_read     : std_logic_VECTOR(9 DOWNTO 0);
SIGNAL line_1_addr_write    : std_logic_VECTOR(9 DOWNTO 0);
SIGNAL line_1_clk_read      : std_logic;
SIGNAL line_1_clk_write     : std_logic;
SIGNAL line_1_din_write     : std_logic_VECTOR(17 DOWNTO 0);
SIGNAL line_1_dout_read     : std_logic_vector(17 DOWNTO 0);
SIGNAL line_1_en_read       : std_logic;
SIGNAL line_1_en_write      : std_logic;
SIGNAL line_1_we_read       : std_logic;
SIGNAL line_1_we_write      : std_logic;
SIGNAL line_2_addr_read     : std_logic_VECTOR(9 DOWNTO 0);
SIGNAL line_2_addr_write    : std_logic_VECTOR(9 DOWNTO 0);
SIGNAL line_2_clk_read      : std_logic;
SIGNAL line_2_clk_write     : std_logic;
SIGNAL line_2_din_write     : std_logic_VECTOR(17 DOWNTO 0);
SIGNAL line_2_dout_read     : std_logic_vector(17 DOWNTO 0);
SIGNAL line_2_en_read       : std_logic;
SIGNAL line_2_en_write      : std_logic;
SIGNAL line_2_we_read       : std_logic;
SIGNAL line_2_we_write      : std_logic;
SIGNAL line_3_addr_read     : std_logic_VECTOR(9 DOWNTO 0);
SIGNAL line_3_addr_write    : std_logic_VECTOR(9 DOWNTO 0);
SIGNAL line_3_clk_read      : std_logic;
SIGNAL line_3_clk_write     : std_logic;
SIGNAL line_3_din_write     : std_logic_VECTOR(17 DOWNTO 0);
SIGNAL line_3_dout_read     : std_logic_vector(17 DOWNTO 0);
SIGNAL line_3_en_read       : std_logic;
SIGNAL line_3_en_write      : std_logic;
SIGNAL line_3_we_read       : std_logic;
SIGNAL line_3_we_write      : std_logic;
SIGNAL line_4_addr_read     : std_logic_VECTOR(9 DOWNTO 0);
SIGNAL line_4_addr_write    : std_logic_VECTOR(9 DOWNTO 0);
SIGNAL line_4_clk_read      : std_logic;
SIGNAL line_4_clk_write     : std_logic;
SIGNAL line_4_din_write     : std_logic_VECTOR(17 DOWNTO 0);
SIGNAL line_4_dout_read     : std_logic_vector(17 DOWNTO 0);
SIGNAL line_4_en_read       : std_logic;
SIGNAL line_4_en_write      : std_logic;
SIGNAL line_4_we_read       : std_logic;
SIGNAL line_4_we_write      : std_logic;
SIGNAL line_5_addr_read     : std_logic_VECTOR(9 DOWNTO 0);
SIGNAL line_5_addr_write    : std_logic_VECTOR(9 DOWNTO 0);
SIGNAL line_5_clk_read      : std_logic;
SIGNAL line_5_clk_write     : std_logic;
```

```
SIGNAL line_5_din_write        : std_logic_VECTOR(17 DOWNTO 0);
SIGNAL line_5_dout_read        : std_logic_vector(17 DOWNTO 0);
SIGNAL line_5_en_read          : std_logic;
SIGNAL line_5_en_write         : std_logic;
SIGNAL line_5_we_read          : std_logic;
SIGNAL line_5_we_write         : std_logic;
SIGNAL web                     : std_logic;


-- Component Declarations
COMPONENT filter_controller
PORT (
  line_1_dout_read       : IN     std_logic_vector (17 DOWNTO 0);
  line_1_addr_read       : OUT    std_logic_VECTOR (9 DOWNTO 0);
  line_1_addr_write      : OUT    std_logic_VECTOR (9 DOWNTO 0);
  line_1_clk_read        : OUT    std_logic ;
  line_1_clk_write       : OUT    std_logic ;
  line_1_din_write       : OUT    std_logic_VECTOR (17 DOWNTO 0);
  line_1_en_write        : OUT    std_logic ;
  line_1_en_read         : OUT    std_logic ;
  line_1_we_write        : OUT    std_logic ;
  line_1_we_read         : OUT    std_logic ;
  line_2_dout_read       : IN     std_logic_vector (17 DOWNTO 0);
  line_2_addr_read       : OUT    std_logic_VECTOR (9 DOWNTO 0);
  line_2_addr_write      : OUT    std_logic_VECTOR (9 DOWNTO 0);
  line_2_clk_read        : OUT    std_logic ;
  line_2_clk_write       : OUT    std_logic ;
  line_2_din_write       : OUT    std_logic_VECTOR (17 DOWNTO 0);
  line_2_en_write        : OUT    std_logic ;
  line_2_en_read         : OUT    std_logic ;
  line_2_we_write        : OUT    std_logic ;
  line_2_we_read         : OUT    std_logic ;
  line_3_dout_read       : IN     std_logic_vector (17 DOWNTO 0);
  line_3_addr_read       : OUT    std_logic_VECTOR (9 DOWNTO 0);
  line_3_addr_write      : OUT    std_logic_VECTOR (9 DOWNTO 0);
  line_3_clk_read        : OUT    std_logic ;
  line_3_clk_write       : OUT    std_logic ;
  line_3_din_write       : OUT    std_logic_VECTOR (17 DOWNTO 0);
  line_3_en_write        : OUT    std_logic ;
  line_3_en_read         : OUT    std_logic ;
  line_3_we_write        : OUT    std_logic ;
  line_3_we_read         : OUT    std_logic ;
  line_4_dout_read       : IN     std_logic_vector (17 DOWNTO 0);
  line_4_addr_read       : OUT    std_logic_VECTOR (9 DOWNTO 0);
  line_4_addr_write      : OUT    std_logic_VECTOR (9 DOWNTO 0);
  line_4_clk_read        : OUT    std_logic ;
  line_4_clk_write       : OUT    std_logic ;
  line_4_din_write       : OUT    std_logic_VECTOR (17 DOWNTO 0);
  line_4_en_write        : OUT    std_logic ;
  line_4_en_read         : OUT    std_logic ;
  line_4_we_write        : OUT    std_logic ;
  line_4_we_read         : OUT    std_logic ;
  line_5_dout_read       : IN     std_logic_vector (17 DOWNTO 0);
  line_5_addr_read       : OUT    std_logic_VECTOR (9 DOWNTO 0);
  line_5_addr_write      : OUT    std_logic_VECTOR (9 DOWNTO 0);
  line_5_clk_read        : OUT    std_logic ;
  line_5_clk_write       : OUT    std_logic ;
```

```
    line_5_din_write       : OUT    std_logic_VECTOR (17 DOWNTO 0);
    line_5_en_write        : OUT    std_logic ;
    line_5_en_read         : OUT    std_logic ;
    line_5_we_write        : OUT    std_logic ;
    line_5_we_read         : OUT    std_logic ;
    hblank                 : IN     std_logic ;
    vblank                 : IN     std_logic ;
    filter_coefficient_dout : IN     std_logic_vector (17 DOWNTO 0);
    filter_coefficient_addr : OUT    std_logic_VECTOR (9 DOWNTO 0);
    filter_coefficient_clk  : OUT    std_logic ;
    filter_coefficient_din  : OUT    std_logic_VECTOR (17 DOWNTO 0);
    filter_coefficient_en   : OUT    std_logic ;
    filter_coefficient_we   : OUT    std_logic ;
    image_data             : IN     std_logic_vector (15 DOWNTO 0);
    video_clk              : IN     std_logic ;
    reset                  : IN     std_logic ;
    sum                    : OUT    std_logic_vector (41 DOWNTO 0)
  );
  END COMPONENT;
  COMPONENT dpram_1kx18
  PORT (
    addra : IN     std_logic_VECTOR (9 DOWNTO 0);
    addrb : IN     std_logic_VECTOR (9 DOWNTO 0);
    clka  : IN     std_logic ;
    clkb  : IN     std_logic ;
    dina  : IN     std_logic_VECTOR (17 DOWNTO 0);
    dinb  : IN     std_logic_VECTOR (17 DOWNTO 0);
    douta : OUT    std_logic_VECTOR (17 DOWNTO 0);
    doutb : OUT    std_logic_VECTOR (17 DOWNTO 0);
    ena   : IN     std_logic ;
    enb   : IN     std_logic ;
    wea   : IN     std_logic ;
    web   : IN     std_logic
  );
  END COMPONENT;


BEGIN

  -- Instance port mappings.
  I5 : filter_controller
    PORT MAP (
      line_1_dout_read       => line_1_dout_read,
      line_1_addr_read       => line_1_addr_read,
      line_1_addr_write      => line_1_addr_write,
      line_1_clk_read        => line_1_clk_read,
      line_1_clk_write       => line_1_clk_write,
      line_1_din_write       => line_1_din_write,
      line_1_en_write        => line_1_en_write,
      line_1_en_read         => line_1_en_read,
      line_1_we_write        => line_1_we_write,
      line_1_we_read         => line_1_we_read,
      line_2_dout_read       => line_2_dout_read,
      line_2_addr_read       => line_2_addr_read,
      line_2_addr_write      => line_2_addr_write,
      line_2_clk_read        => line_2_clk_read,
      line_2_clk_write       => line_2_clk_write,
```

```
    line_2_din_write       => line_2_din_write,
    line_2_en_write        => line_2_en_write,
    line_2_en_read         => line_2_en_read,
    line_2_we_write        => line_2_we_write,
    line_2_we_read         => line_2_we_read,
    line_3_dout_read       => line_3_dout_read,
    line_3_addr_read       => line_3_addr_read,
    line_3_addr_write      => line_3_addr_write,
    line_3_clk_read        => line_3_clk_read,
    line_3_clk_write       => line_3_clk_write,
    line_3_din_write       => line_3_din_write,
    line_3_en_write        => line_3_en_write,
    line_3_en_read         => line_3_en_read,
    line_3_we_write        => line_3_we_write,
    line_3_we_read         => line_3_we_read,
    line_4_dout_read       => line_4_dout_read,
    line_4_addr_read       => line_4_addr_read,
    line_4_addr_write      => line_4_addr_write,
    line_4_clk_read        => line_4_clk_read,
    line_4_clk_write       => line_4_clk_write,
    line_4_din_write       => line_4_din_write,
    line_4_en_write        => line_4_en_write,
    line_4_en_read         => line_4_en_read,
    line_4_we_write        => line_4_we_write,
    line_4_we_read         => line_4_we_read,
    line_5_dout_read       => line_5_dout_read,
    line_5_addr_read       => line_5_addr_read,
    line_5_addr_write      => line_5_addr_write,
    line_5_clk_read        => line_5_clk_read,
    line_5_clk_write       => line_5_clk_write,
    line_5_din_write       => line_5_din_write,
    line_5_en_write        => line_5_en_write,
    line_5_en_read         => line_5_en_read,
    line_5_we_write        => line_5_we_write,
    line_5_we_read         => line_5_we_read,
    hblank                 => hblank,
    vblank                 => vblank,
    filter_coefficient_dout => filter_coefficient_dout,
    filter_coefficient_addr => filter_coefficient_addr,
    filter_coefficient_clk  => filter_coefficient_clk,
    filter_coefficient_din  => filter_coefficient_din,
    filter_coefficient_en   => filter_coefficient_en,
    filter_coefficient_we   => filter_coefficient_we,
    image_data             => image_data,
    video_clk              => video_clk,
    reset                  => reset,
    sum                    => sum
  );
I0 : dpram_1kx18
  PORT MAP (
    addra => line_1_addr_read,
    addrb => line_1_addr_write,
    clka  => line_1_clk_read,
    clkb  => line_1_clk_write,
    dina  => dina,
    dinb  => line_1_din_write,
    douta => line_1_dout_read,
```

```
      doutb => OPEN,
      ena   => line_1_en_read,
      enb   => line_1_en_write,
      wea   => line_1_we_read,
      web   => line_1_we_write
    );
  I1 : dpram_1kx18
    PORT MAP (
      addra => line_2_addr_read,
      addrb => line_2_addr_write,
      clka  => line_2_clk_read,
      clkb  => line_2_clk_write,
      dina  => dina1,
      dinb  => line_2_din_write,
      douta => line_2_dout_read,
      doutb => OPEN,
      ena   => line_2_en_read,
      enb   => line_2_en_write,
      wea   => line_2_we_read,
      web   => line_2_we_write
    );
  I2 : dpram_1kx18
    PORT MAP (
      addra => line_3_addr_read,
      addrb => line_3_addr_write,
      clka  => line_3_clk_read,
      clkb  => line_3_clk_write,
      dina  => dina2,
      dinb  => line_3_din_write,
      douta => line_3_dout_read,
      doutb => OPEN,
      ena   => line_3_en_read,
      enb   => line_3_en_write,
      wea   => line_3_we_read,
      web   => line_3_we_write
    );
  I3 : dpram_1kx18
    PORT MAP (
      addra => line_4_addr_read,
      addrb => line_4_addr_write,
      clka  => line_4_clk_read,
      clkb  => line_4_clk_write,
      dina  => dina3,
      dinb  => line_4_din_write,
      douta => line_4_dout_read,
      doutb => OPEN,
      ena   => line_4_en_read,
      enb   => line_4_en_write,
      wea   => line_4_we_read,
      web   => line_4_we_write
    );
  I4 : dpram_1kx18
    PORT MAP (
      addra => line_5_addr_read,
      addrb => line_5_addr_write,
      clka  => line_5_clk_read,
      clkb  => line_5_clk_write,
```

```
      dina  => dina4,
      dinb  => line_5_din_write,
      douta => line_5_dout_read,
      doutb => OPEN,
      ena   => line_5_en_read,
      enb   => line_5_en_write,
      wea   => line_5_we_read,
      web   => line_5_we_write
    );
  I6 : dpram_1kx18
    PORT MAP (
      addra => filter_coefficient_addr,
      addrb => coef_addr,
      clka  => filter_coefficient_clk,
      clkb  => coef_clk,
      dina  => filter_coefficient_din,
      dinb  => coef_din,
      douta => filter_coefficient_dout,
      doutb => OPEN,
      ena   => filter_coefficient_en,
      enb   => coef_en,
      wea   => filter_coefficient_we,
      web   => coef_we
    );

END struct;
```

APPENDIX B

FPGA Suppliers

Actel Corporation

Altera Corporation

AMI Semiconductor

Amphion Semiconductor, Inc.

Aptix Corporation

Atmel Corporation

Kawasaki LSI U.S.A., Inc.

Nallatech, Inc.

Pentek, Inc.

SiQUEST, Inc.

Tekmos, Inc.

Transtech Parallel Systems

Xilinx, Inc.

APPENDIX C

Communication Block VHDL Codes

RS232 Receiver Block VHDL Code

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;


ENTITY rs232_receiver IS

PORT(
     baudrate  : IN     std_logic_vector (7 DOWNTO 0);
     clk       : IN     std_logic;
     cs        : IN     std_logic;
     rst       : IN     std_logic;
     sin       : IN     std_logic;
     data_out  : OUT    std_logic_vector (7 DOWNTO 0);
     datavalid : OUT    std_logic;
     receiving : OUT    std_logic
   );
END  rs232_receiver ;

architecture arch_rs232_receiver of rs232_receiver is


constant wait_first : std_logic_vector(1 downto 0) := "00";
constant count_half : std_logic_vector(1 downto 0) := "01";
constant get_first  : std_logic_vector(1 downto 0) := "10";
constant get_last   : std_logic_vector(1 downto 0) := "11";

signal   counter    : std_logic_vector(7 downto 0);
```

```
signal   count_8    : std_logic_vector(2 downto 0);
signal   rcv_data   : std_logic_vector(7 downto 0);
signal   data_flag  : std_logic;
signal   rcv_state  : std_logic_vector(1 downto 0);

begin

 process(clk, rst)

 begin

  if rst = '0' then
    rcv_state          <= wait_first;
    counter            <= (others => '0');
    count_8            <= (others => '0');
    rcv_data           <= (others => '0');
    data_flag          <= '1';
    receiving          <= '0';
  elsif rising_edge(clk) then
    if cs = '0' then
      rcv_state        <= wait_first;
      counter          <= (others => '0');
      count_8          <= (others => '0');
      receiving        <= '0';
      data_flag        <= '1';
    else
     case rcv_state is
         when wait_first =>
           counter       <= (others => '0');
           data_flag     <= '1';
           if sin = '1' then
          rcv_state    <= wait_first;
        else
           rcv_state    <= count_half;
        end if;
       when count_half          =>
         counter       <= counter+1;
         if sin = '0' then
```

```
            if counter     = ('0'&baudrate(7 downto 1)) then
               counter    <= (others => '0');
               rcv_state  <= get_first;
               receiving  <= '1';
            else
               rcv_state  <= count_half;
            end if;
          else
             rcv_state     <= wait_first;
          end if;
        when get_first             =>
          counter         <= counter+1;
         if counter       = baudrate then
            counter      <= (others => '0');
            rcv_data     <= sin&rcv_data(7 downto 1);
            count_8      <= count_8+1;
          if count_8 = "111" then
              rcv_state <= get_last;
           end if;
          end if;
        when get_last      =>
          counter         <= counter+1;
          if counter = baudrate then
            counter      <= (others => '0');
            receiving    <= '0';
            if sin='1' then
              data_flag <= '0';
            end if;
            rcv_state    <=wait_first;
          end if;

        when others       =>
          null;
      end case;
    end if;
  end if;

end process;
```

```
 process(clk,rst)
 begin
  if rst='0' then
     data_out      <=(others=>'0');
     datavalid     <= '1';
  elsif rising_edge(clk) then
    datavalid      <= '1';
    if data_flag='0' then
       data_out    <= rcv_data;
       datavalid   <= '0';
    end if;
  end if;
 end process;

END ;
--*********************************************************************************
```

RS232 Transmitter Block VHDL Code

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;

ENTITY rs232_transmitter IS

PORT(
    baudrate     : IN     std_logic_vector (7 DOWNTO 0);
    clk          : IN     std_logic;
    cs           : IN     std_logic;
    data_in      : IN     std_logic_vector (7 DOWNTO 0);
    rst          : IN     std_logic;
    start_xmit   : IN     std_logic;
    sout         : OUT    std_logic;
    transmitting : OUT    std_logic
  );
```

```
END  rs232_transmitter ;

architecture  arch_rs232_transmitter of rs232_transmitter is

signal xmitdt        : std_logic_vector(7 downto 0);
signal xmit_stm      : std_logic_vector(1 downto 0);
signal cnt1          : std_logic_vector(7 downto 0);
signal cnt_xmit_stop : std_logic_vector(2 downto 0);

constant WAITSTATE   : std_logic_vector(1 downto 0) := "00";
constant STARTSTATE  : std_logic_vector(1 downto 0) := "01";
constant DATASTATE   : std_logic_vector(1 downto 0) := "10";
constant STOPSTATE   : std_logic_vector(1 downto 0) := "11";
begin

 process(clk, rst)
 begin
  if rst = '0' then
    xmit_stm        <= WAITSTATE;
    sout            <= '1';
    cnt1            <= (others => '0');
    cnt_xmit_stop   <= (others => '0');
    xmitdt          <= (others => '0');
    transmitting    <= '0';
  elsif falling_edge(clk) then
    if cs = '0' then
      xmit_stm      <= WAITSTATE;
      transmitting  <= '0';
      sout          <= '1';
      cnt1          <= (others => '0');
      cnt_xmit_stop <= (others => '0');
    else
     case xmit_stm is
      --ready to transmit data 1 start,1 stop bit
      when WAITSTATE =>
        if start_xmit = '1' then
          xmit_stm <= STARTSTATE;
          xmitdt   <= data_in;
```

```
          end if;
        when STARTSTATE          =>
          transmitting <= '1';
          sout        <= '0';
          cnt1        <= cnt1+1;
          if cnt1 = baudrate then
            cnt1      <= (others => '0');
            xmit_stm  <= DATASTATE;
          end if;
        when DATASTATE               =>
          sout              <= xmitdt(0);
          cnt1              <= cnt1+1;
          if cnt1 = baudrate then
            cnt_xmit_stop    <= cnt_xmit_stop+1;
            xmitdt           <= '0'&xmitdt(7 downto 1);
            if cnt_xmit_stop = "111" then
              sout          <= '1';
              cnt_xmit_stop <= (others => '0');
              xmit_stm      <= STOPSTATE;
            end if;
            cnt1            <= (others => '0');
          end if;
        when STOPSTATE =>
          cnt1 <=cnt1+1;
          if cnt1=baudrate then
            cnt1<=(others=>'0');
            xmit_stm<=WAITSTATE;
            transmitting<='0';
          end if;
        when others=>
          null;
       end case;
     end if;
   end if;
 end process;
END ;
```